Руководство по эксплуатации R-Vision SIEM

v2.7.2

Оглавление

Об R-Vision SIEM . . . 11
Архитектура системы . . . 14
Порты и протоколы, используемые системой . . . 18
Поддерживаемые внешние системы . . . 22
О версии 2.7 . . . 23
Развертывание системы . . . 28
Технические требования . . . 29
Развертывание кластера Kubernetes . . . 38
Установка кластера Kubernetes с помощью Kubespray . . . 38
Установка кластера Kubernetes вручную . . . 45
Установка СУБД . . . 52
Установка ClickHouse в выделенном кластере . . . 53
Настройка нового кластера ClickHouse . . . 54
Настройка существующего кластера ClickHouse . . . 64
Справка о параметрах ClickHouse . . . 80
Вынесенная установка PostgreSQL . . . 83
Установка PostgreSQL на выделенном сервере . . . 84
Установка PostgreSQL в выделенном кластере . . . 88
Настройка балансировки запросов к ClickHouse и PostgreSQL . . . 97
Подготовка к установке системы . . . 107
Установка системы . . . 113
Пакет установки . . . 114
Процесс установки системы . . . 119
Установка сателлитов . . . 146
Технические требования к сателлитам . . . 146
Подготовка к установке сателлита . . . 148
Процесс установки сателлита . . . 154
Обновление системы . . . 160
Ручное обновление компонентов системы . . . 177
Переход с агентов R-Vision Endpoint на системные агенты . . . 195
Развертывание агентов . . . 196
Технические требования к хосту агента . . . 197
Подготовка к установке агента из золотого образа . . . 199
Установка агента . . . 200
Настройка конфигурации агента . . . 203
Подключение агента к системе . . . 219
Подключение агента с шифрованием . . . 224
Добавление TLS-сертификата на агенте . . . 224
Настройка шифрования между агентом и шлюзом . . . 229

Работа с агентами на хосте 232
Обновление агента 240
Удаление агента 242
Удаление системы 244
Администрирование системы 247
Добавление SSL-сертификата 249
Добавление самоподписанного сертификата в список доверенных 256
Работа с кластером 257
Подключение к кластеру 258
Мониторинг кластера 261
Состояние кластера 262
Создание пользовательских дашбордов в Grafana 266
Анализ работоспособности сервисов кластера 268
Анализ работоспособности control plane кластера 270
Анализ состояния сателлита 273
Просмотр журнала событий кластера 273
Просмотр состояния подов кластера 275
Список селекторов системных подов 278
Настройка оповещений 283
Настройка оповещений в Telegram 290
Настройка оповещений по электронной почте 291
Работа с метриками Prometheus 296
Управление кластером 302
Управление узлами кластера 302
Работа с подами audit-manager и collector-audit 306
Работа с подом collector-manager 309
Работа с подом active-list-manager 311
Увеличение размера тома для подов 314
Получение идентификатора сервиса в NATS Service Discovery 316
Работа с объектным хранилищем NATS Object Store 317
Настройка стримов в менеджере пространств 319
Работа с паролями в кластере 323
Изменение доменного имени установленного экземпляра системы 332
Настройка отдельного диска под образы контейнеров 333
Ошибки при запуске пода ingress-nginx-controller 340
Поиск установленных экземпляров системы 343
Настройка системы 345
Настройка быстрого доступа к продуктам R-Vision 345
Настройка формата даты 346
Настройка времени хранения оповещений 350
Изменение пула адресов в MetalLB 351
Настройка времени хранения запросов 353

Настройка ограничений размера файла драйвера БД . . . 354
Настройка ограничений размера загружаемых файлов . . . 355
Настройка получения событий по произвольному порту . . . 356
Резервное копирование и восстановление компонентов системы . . . 360
Резервное копирование хранилища etcd . . . 361
Резервное копирование без Backup Wrapper . . . 364
Резервное копирование PostgreSQL без Backup Wrapper . . . 365
Резервное копирование ClickHouse без Backup Wrapper . . . 369
Резервное копирование сервисов платформы . . . 373
Резервное копирование внутренних баз данных . . . 377
Работа с Backup Wrapper . . . 386
Настройка PostgreSQL для работы с Backup Wrapper . . . 387
Настройка ClickHouse для работы с Backup Wrapper . . . 393
Настройка многоуровневого хранилища ClickHouse . . . 395
Установка и настройка Backup Wrapper . . . 402
Удаление Backup Wrapper . . . 408
Резервное копирование внешних баз данных . . . 409
Настройка резервного копирования из кластера системы . . . 409
Настройка резервного копирования вне кластера системы . . . 413
Восстановление из резервных копий . . . 416
Восстановление внутренних баз данных . . . 416
Восстановление внешних баз данных . . . 421
Восстановление базы данных PostgreSQL . . . 421
Восстановление базы данных ClickHouse . . . 427
Устранение проблем при восстановлении внешних баз данных . . . 431
Восстановление сервисов платформы . . . 434
Архивирование и восстановление исторических событий . . . 440
Настройка утилиты для архивирования и восстановления исторических событий . . . 440
Работа с утилитой для архивирования исторических событий . . . 448
Управление лицензией . . . 453
Управление пространствами . . . 456
Работа с пространствами . . . 457
Работа с сервисами . . . 461
Управление доступом . . . 468
Тенанты . . . 468
Пользователи . . . 472
Группы пользователей . . . 480
Роли . . . 488
Стандартные роли . . . 493
Политики входа и пароля . . . 497
Настройка времени жизни для токенов доступа . . . 500
LDAP-соединения . . . 502

Настройка сервисной учетной записи ALD Pro 507
Атрибуты Distinguished Name 508
Политики управления доступом к событиям 510
Настройка агентов 514
Настройки менеджера агентов 514
Сбор инвентарной информации 515
Сбор событий ИБ на агентах 532
Аудит 535
Аудит источников событий 535
Политики аудита источников событий 536
Настройка конвейера для аудита коллектора 552
Аудит системных событий 562
Модель события аудита 568
Создание оповещений о критических событиях 571
Отслеживание действий суперадминистраторов 576
Отслеживание действий над объектами 578
Мониторинг RQL-запросов и управление ресурсами 582
Авторизация 588
Интерфейс 589
Документация 592
Настройки профиля 592
Настройки интерфейса 593
Смена пароля 595
Работа с таблицами 595
Работа с объектами 600
Изменение объектов 600
Удаление объектов 603
Клавиши и комбинации клавиш 605
Начало работы с системой 608
Подключение источников 610
Обработка событий на одном конвейере 611
Обработка событий на нескольких конвейерах 645
Обработка событий на нескольких коллекторах 653
Обработка событий с нескольких IP-адресов 660
Сбор событий из вынесенного коллектора 664
Сбор событий с агентов 671
Аналитика 680
Работа с дашбордами 680
Работа с виджетами 686
Рекомендации по работе с дашбордами 708
Работа с оповещениями 710
Работа с отчетами 722

Работа с разделом Поиск . . . 731
Ресурсы . . . 768
Работа с коллекторами . . . 769
Работа с конвейерами . . . 782
Работа с точками входа . . . 796
Работа с VRL-трансформациями . . . 846
Работа с фильтрами . . . 848
Работа с маршрутизаторами . . . 849
Работа с коррелляторами . . . 852
Работа с нормализаторами . . . 856
Работа с агрегаторами . . . 860
Работа с сервисами оповещений . . . 863
Работа со сбором метрик . . . 867
Работа с конечными точками . . . 868
Работа с шинами . . . 880
Работа с обогащениями . . . 883
Работа с моделями событий . . . 886
Универсальная модель события . . . 894
Универсальная модель события 2.0 . . . 907
Типы данных . . . 921
Типы операций в событии . . . 926
Работа с базами данных событий . . . 931
Работа с хранилищами событий . . . 936
Работа с активными списками . . . 942
Работа с интеграциями . . . 951
Интеграция с R-Vision SOAR . . . 954
Почтовые интеграции . . . 963
Работа с драйверами БД . . . 969
Работа с секретами . . . 972
Агенты . . . 977
Работа с агентами . . . 977
Работа с группами агентов . . . 987
Настройка конфигурации группы агентов . . . 991
Синтаксис языка XPath . . . 998
Работа со шлюзами . . . 1001
Действия на агентах . . . 1006
Ручное снятие самоизоляции узла . . . 1009
Экспертиза . . . 1011
Работа с элементами экспертизы . . . 1011
Правила корреляции . . . 1028
Построение правила корреляции в конструкторе . . . 1032
Императивные правила . . . 1049

Декларативные правила 1075
Правила нормализации 1121
Правила агрегации 1134
Правила сегментации 1155
Схемы активных списков 1168
Схемы дашбордов 1180
Схемы шаблонов отчетов 1199
Таблицы обогащения 1220
Шаблоны поиска 1232
Глобальные функции 1243
Тестирование правил экспертизы с помощью языка VRL 1258
Оценка покрытия техник MITRE ATT&CK правилами экспертизы 1263
Справочник по языку запросов RQL 1265
Работа с RQL-запросами 1265
Функции 1292
Агрегатные функции 1293
Условные функции 1298
Функции сравнения 1299
Логические функции 1300
Функции преобразования типов 1301
Функции для работы с Nullable-аргументами 1309
Математические функции 1310
Арифметические функции 1312
Битовые функции 1314
Функции битовых карт 1315
Функции округления 1316
Функции для работы с датой и временем 1317
Функции для работы со строками 1319
Функции разбиения и слияния строк и массивов 1323
Функции для поиска в строках 1324
Функции для замены строк 1326
Функции для работы с массивами 1327
Географические функции 1331
Функции временных рядов 1335
Функции для работы с IP-адресами 1336
Функции для работы с URL 1337
Функции для работы с UUID 1339
Справочник по языку преобразования данных VRL 1341
О языке VRL 1342
Выражения 1346
Пустое значение 1349
Логическое значение 1350

Целое число 1350
Число с плавающей точкой 1351
Метка времени 1353
Строка 1354
Регулярное выражение 1358
Массив 1368
Объект 1370
Переменная 1371
Путь 1372
Индекс 1376
Присваивание 1377
Арифметическое выражение 1381
Сравнение 1383
Логическое выражение 1386
Вызов функции 1388
Объединение 1391
Условие 1392
Блок 1395
Прерывание 1396
Функции 1397
Функции разбора 1398
Функции кодирования и декодирования 1479
Функции событий и активных списков 1502
Функции коллекций 1514
Функции строк 1549
Функции типов и форматирования 1580
Хэш-функции и криптографические функции 1622
Функции генерации случайных значений и ID 1636
Системные функции 1644
Функции отладки 1646
Справочник ошибок 1649
Советы по написанию кода на языке VRL 1667
Оптимизация кода 1667
Оформление кода 1673
Работа с VRL-сниппетами 1674
Работа с VRL-песочницей 1677
GraphQL API 1685
Открытие доступа к GraphQL-песочнице 1688
API-токены 1689
Работа с GraphQL 1691
Обработка ошибок API 1714
Устранение проблем 1718

Устранение проблем при установке или обновлении системы . . . 1718
Устранение проблем при установке или обновлении агентов . . . 1721
Устранение проблем при эксплуатации системы . . . 1726
Глоссарий . . . 1773

Документация по продукту "Система управления событиями информационной безопасности R-Vision Security Information and Event Management" (далее по тексту — "R-Vision SIEM" или "система") содержит описание функциональных характеристик системы и сведения, необходимые для ее эксплуатации, включая пошаговые инструкции и рекомендации по настройке.

Об R-Vision SIEM

Система управления событиями информационной безопасности R-Vision Security Information and Event Management (R-Vision SIEM) — это комплексное программное решение, предназначенное для сбора, обработки, хранения, анализа и визуализации данных, относящихся к событиям в IT-инфраструктуре организации.

Основные функциональные возможности R-Vision SIEM

Управление модулями в распределенной инфраструктуре

- Размещение модулей комплекса в распределенной инфраструктуре.
- Управление пространствами и размещенными в них сервисами.

Управление доступом к функциям системы

- Поддержка мультитенантности — возможности создать в рамках одного экземпляра системы несколько логических копий системы (тенантов), каждый из которых конфигурируется независимо.
- Гибкое управление пользователями: настройка прав доступа на основе ролевой модели, задание политик входа и пароля.

Обработка событий ИБ

- Сбор, обработка и хранение событий, полученных от поддерживаемых источников.
- Глубокий аналитический разбор событий, включающий их корреляцию, нормализацию, агрегацию и сегментацию.
- Создание хранилищ событий и пользовательских моделей событий.
- Выполнение аналитических запросов к событиям в хранилищах.
- Тестирование работоспособности объектов системы.
- Мониторинг и анализ системных событий и источников.
- Формирование виджетов и аналитических дашбордов для отслеживания показателей по событиям, а также метрик работы компонентов системы.
- Построение отчетов на основе аналитических данных и отправка их во внешние системы с помощью интеграций.

Управление агентами на конечных устройствах

- Установка агентов на конечные устройства для выявления угроз и осуществления реагирования на эти угрозы.
- Сбор инвентарной информации, сведений об уязвимом ПО, телеметрии и событий на конечных устройствах.

Сертификаты совместимости R-Vision SIEM с внешними системами

Совместимость и корректность работы R-Vision SIEM начиная с версии 2.4.1 и выше в окружении операционной системы **Альт СП Сервер 10.2.1** подтверждена сертификатом.

R-Vision SIEM позволяет производить сбор и нормализацию событий, поступающих из нижеперечисленных систем. Совместимость R-Vision SIEM с этими системами подтверждена сертификатами, выданными в результате испытаний.

- Начиная с версии 2.0.0 и выше:
  - Система централизованного управления ИТ-инфраструктурой **РЕД АДМ 1.2.2**.
  - Система двухфакторной аутентификации и контроля доступа **MULTIFACTOR**.
  - Сетевой сенсор обнаружения сетевых атак и вредоносного программного обеспечения **ViPNet IDS NS 3.10.0**.
  - Программно-аппаратный комплекс автоматического выявления инцидентов **ViPNet TIAS 3.10.0**.
- Начиная с версии 2.4.1 и выше:
  - Операционная система **РЕД ОС 8**.

Технические характеристики

- Функционирование системы на основе технологии Kubernetes с поддержкой Linux-контейнеров версии не ниже 1.24.
- База данных хранения событий на основе технологии распределенной СУБД ClickHouse.
- Операционная база данных на основе технологии СУБД PostgreSQL.
- Хранение данных в горячем или холодном хранилище.
- Создание пользовательских элементов экспертизы в рамках единой схемы описания объектов RObject, с использованием языка VRL.
- Управление оповещениями и просмотр информации об обнаруженной потенциальной вредоносной активности.
- Предустановленные виджеты и дашборды для визуального анализа данных.
- Отчеты для анализа и представления данных, собранных системой.
- Поддержка сетевого протокола SMTP.
- Интеграция с системой R-Vision SOAR.
- Поддерживаемые среды функционирования:
  - Альт Сервер 11;
  - Альт СП Сервер релиз 10;
  - Astra Linux SE 1.7.2 и выше;
  - РЕД ОС 7.3 и выше.



Поддерживаются как сертифицированные, так и несертифицированные варианты ОС.

Например, для РЕД ОС поддерживаются следующие версии:

- несертифицированные: 7.3 и выше;
- сертифицированные: 7.3c и выше.

- Поддерживаемые версии браузеров:
  - Google Chrome 110 и выше;
  - Microsoft Edge 110 и выше;
  - Mozilla Firefox 115 и выше.

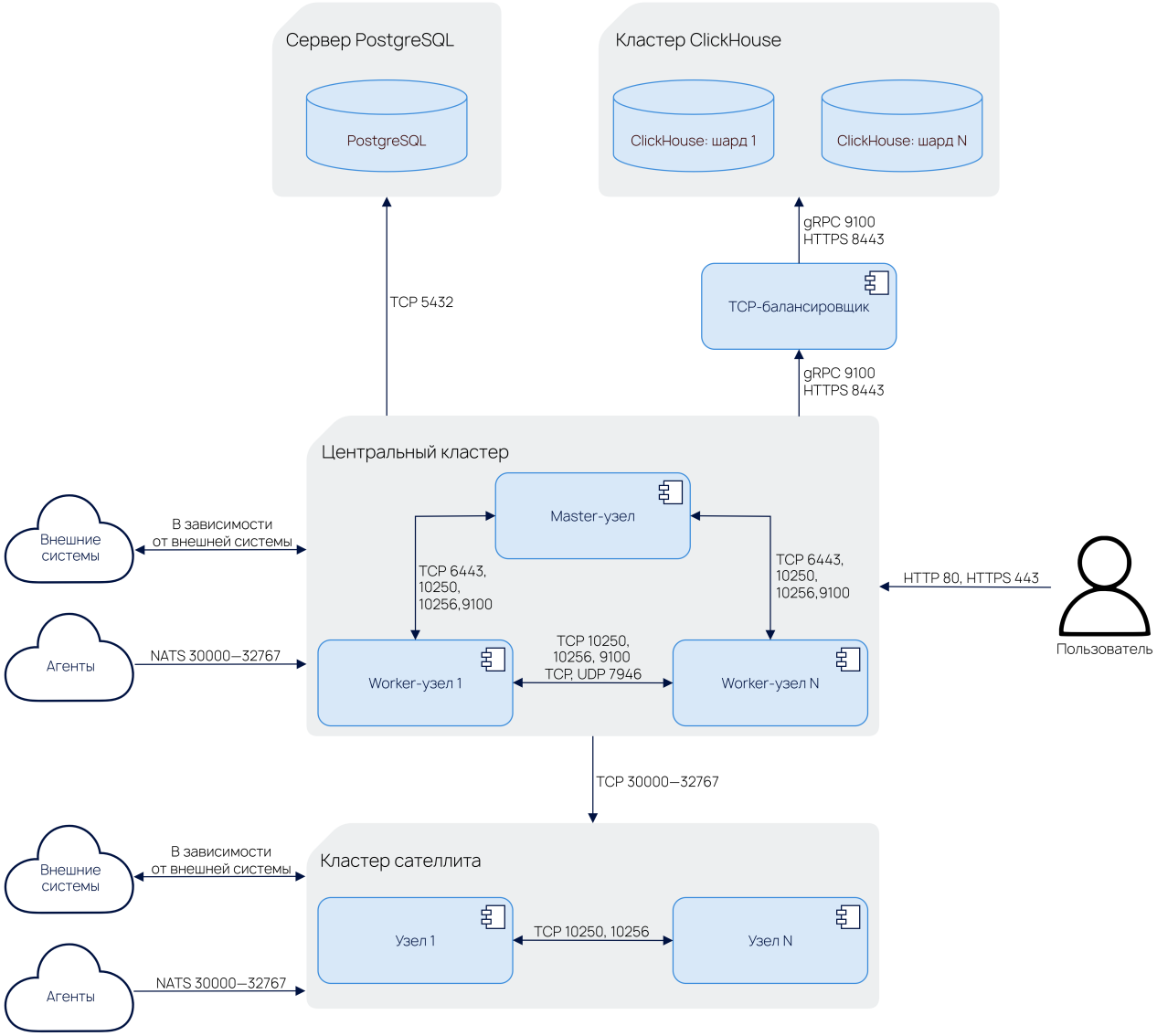
  Рекомендуется использовать наиболее актуальные версии браузеров.
- Поддерживаемые среды виртуализации — любые, которые удовлетворяют техническим требованиям и обеспечивают строгое резервирование ресурсов, а также прямой проброс аппаратных ресурсов в виртуальные машины.
- Для промышленной эксплуатации рекомендуется использовать гипервизоры bare-metal (Type 1), работающие непосредственно на аппаратном уровне. Использование hosted-гипервизоров (Type 2), функционирующих поверх операционной системы, не рекомендуется.
- Во избежание проблем в работе системы на виртуальных машинах (ВМ) под управлением гипервизора Hyper-V убедитесь, что в настройках всех ВМ кластера отключен режим совместимости. Для этого выполните следующие действия над каждой из ВМ:
  a. Перейдите в настройки ВМ.
  b. В дереве настроек выберите пункт **Processor → Совместимость**.
  c. Проверьте состояние флажка **Выполнить перенос на физический компьютер с другой версией процессора**. Если он установлен, выполните следующие действия:
     i. Если ВМ включена, выключите ее.
     ii. Снимите флажок **Выполнить перенос на физический компьютер с другой версией процессора**.
     iii. Сохраните настройки ВМ.
     iv. Включите ВМ.

Архитектура системы

В данном разделе представлены графические схемы архитектуры системы в стандартной и отказоустойчивой конфигурациях и их описание.

Символ **N** на схемах обозначает не определенное число, а "2 и более". Число, обозначаемое символом **N**, может быть различным для каждого компонента системы. Например, при развертывании системы в стандартной конфигурации количество шардов кластера ClickHouse не обязательно должно быть равно количеству worker-узлов центрального кластера.

Стандартная конфигурация системы



Стандартная конфигурация системы включает следующие компоненты:

- **Центральный кластер** — кластер Kubernetes, в котором разворачивается основной экземпляр системы. Основной экземпляр системы содержит все компоненты продукта, включая графический интерфейс.
    - Инструкция по установке центрального кластера Kubernetes приведена в разделе

Развертывание кластера Kubernetes.

- Инструкция по установке основного экземпляра системы приведена в разделе Процесс установки системы.

- **Кластер сателлита** — дополнительный удаленный кластер Kubernetes, в котором разворачивается сателлит. Сателлит — это экземпляр системы, который может работать только с запущенными в нем компонентами. Сателлит не обладает полным спектром функциональности продукта, в том числе для него не предусмотрен графический интерфейс.
  - Инструкция по установке кластера Kubernetes для сателлита аналогична инструкции для центрального кластера. Она приведена в разделе Развертывание кластера Kubernetes.
  - Инструкция по установке сателлита приведена в разделе Процесс установки сателлита.
- **Сервер PostgreSQL** используется для хранения конфигураций сервисов системы.



Система поддерживает установку PostgreSQL как на выделенном сервере, так и в кластере Kubernetes.

Вынесенная установка PostgreSQL не является обязательной, однако рекомендуется для повышения стабильности работы системы и удобства обслуживания СУБД.

- **Кластер ClickHouse** является основным хранилищем данных в системе.
  - Инструкция по установке ClickHouse в выделенном кластере приведена в разделе Установка ClickHouse в выделенном кластере. Можно как развернуть новый кластер ClickHouse, так и использовать существующий кластер.
  - При установке ClickHouse в выделенном кластере можно настроить балансировку запросов к узлам кластера с помощью DNS-сервера или выделенного узла балансировщика.

Также возможна установка ClickHouse в кластере Kubernetes.



Установка ClickHouse в кластере Kubernetes допускается только для минимальной инсталляции системы в демонстрационных целях. Для обеспечения стабильной работы системы рекомендуется устанавливать ClickHouse в выделенном кластере.

- **Внешние системы** позволяют обмениваться событиями ИБ с системой с помощью конвейеров обработки событий.
  - Сбор событий из внешних систем осуществляется с помощью точек входа. Перечень поддерживаемых внешних систем для сбора данных, а также инструкции по настройке соответствующих точек входа приведены в разделе Работа с точками входа.
  - Передача событий во внешние системы осуществляется с помощью конечных точек. Перечень поддерживаемых внешних систем для передачи данных, а также инструкции по настройке соответствующих конечных точек приведены в разделе Работа с конечными точками.



Подробности о поддерживаемых внешних системах приведены в разделе Поддерживаемые внешние системы.

- **Агенты** — бинарные исполняемые файлы, устанавливаемые на конечных устройствах для сбора информации об уязвимостях и угрозах, а также для выполнения действий по реагированию на них.

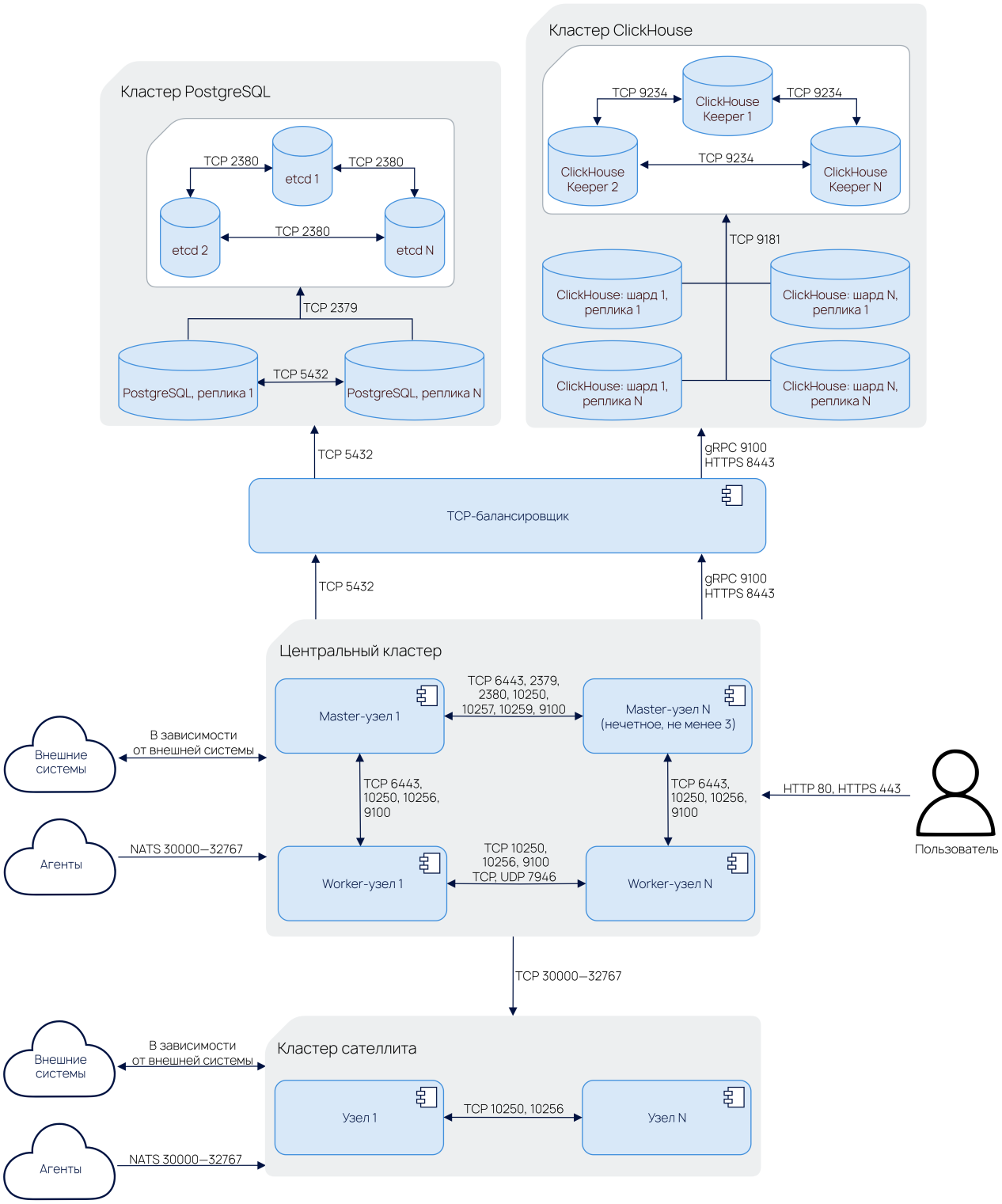
  Агенты передают данные в систему через шлюзы. Для подключения к каждому шлюзу используется порт из диапазона 30 000—32 767. Каждый шлюз могут использовать один или несколько агентов.

  Инструкция по установке агентов на конечные устройства приведена в разделе Развертывание агентов.

  Инструкция по настройке сбора событий на агентах приведена в разделе Настройка агентов.

  Процесс работы с агентами в веб-интерфейсе системы, в том числе выполнения действий по реагированию на угрозы, описан в разделе Работа с агентами.

Отказоустойчивая конфигурация системы



Ключевые отличия отказоустойчивой конфигурации системы от стандартной конфигурации:

- PostgreSQL разворачивается не на одном выделенном сервере, а в отказоустойчивом кластере с использованием выделенных узлов хранилища конфигураций etcd.
- В кластере ClickHouse присутствуют выделенные узлы с ролью Keeper, а также настроена репликация.

  Настройка узлов Keeper и репликации производится с помощью конфигурационных файлов при

установке нового или существующего кластера ClickHouse.

- Балансировка запросов осуществляется как для кластера ClickHouse, так и для кластера PostgreSQL.
- В центральном кластере развернуто несколько master-узлов.



При развертывании кластера в отказоустойчивой конфигурации необходимо использовать нечетное количество master-узлов, не менее трех.

Развертывание кластера с несколькими master-узлами поддерживается при установке Kubernetes как с помощью Kubespray, так и вручную.

Порты и протоколы, используемые системой

Для корректной работы системы нужно убедиться, что ее компоненты могут взаимодействовать по сети через все необходимые порты и протоколы. Эти порты и протоколы приведены в таблице ниже.

Протокол	Порт	Направление	Назначение подключения
HTTP	80	Направление подключения зависит от расположения системы: • При установке системы **на один сервер**: от машин пользователей к серверу системы. • При установке системы **в распределенный кластер**: от машин пользователей к worker-узлам центрального кластера.	Подключение пользователей к системе через веб-интерфейс.
HTTPS	443		
TCP	2379	Между master-узлами центрального кластера.  Этот порт следует открывать, только если вы разворачиваете систему в отказоустойчивой конфигурации.	Чтение и запись в распределенное хранилище etcd.
		Между узлами реплик и выделенными узлами etcd кластера PostgreSQL.  Этот порт следует открывать, только если вы разворачиваете PostgreSQL в выделенном кластере.	

Протокол	Порт	Направление	Назначение подключения
TCP	2380	Между master-узлами центрального кластера.  Этот порт следует открывать, только если вы разворачиваете систему в отказоустойчивой конфигурации. Между выделенными узлами etcd кластера PostgreSQL.  Этот порт следует открывать, только если вы разворачиваете PostgreSQL в выделенном кластере.	Обмен сообщениями между узлами кластера etcd для поддержания консистентности данных.
UDP	4789	Между всеми узлами кластера.  Этот порт следует открывать, только если вы используете сетевой плагин Calico в режиме VXLAN.	VXLAN-туннель для маршрутизации данных между подами.
TCP	5432	Между центральным кластером и вынесенной БД PostgreSQL.  Этот порт следует открывать, только если вы разворачиваете PostgreSQL на выделенном сервере или в выделенном кластере. Между узлами реплик кластера PostgreSQL.  Этот порт следует открывать, только если вы разворачиваете PostgreSQL в выделенном кластере.	Выполнение запросов к БД PostgreSQL на чтение, запись и репликацию данных.
TCP	6443	Между master- и worker-узлами центрального кластера.	Выполнение запросов к Kubernetes API.

Протокол	Порт	Направление	Назначение подключения
TCP, UDP	7946	Между всеми worker-узлами центрального кластера.	Выбор ведущего узла спикером MetalLB в режиме L2.
HTTPS	8443	Между master-узлами центрального кластера и вынесенным ClickHouse.	Взаимодействие системы и ClickHouse.
TCP	9090	От каждого worker-узла к хосту, выделенному для мониторинга состояния кластера.  Этот порт следует открывать, только если вы используете выделенный хост для мониторинга.	Сбор метрик Prometheus.
TCP	9100	Между всеми узлами центрального кластера.	Сбор метрик Prometheus Node Exporter.
		Между узлами центрального кластера и вынесенным ClickHouse.	Выполнение запросов к ClickHouse по API.
TCP	9101	Между всеми узлами центрального кластера.  Этот порт следует открывать, только если Prometheus Node Exporter использует порт 9101 вместо 9100.	Сбор метрик Prometheus Node Exporter.
TCP	9181	Между узлами шардов и реплик ClickHouse и выделенными узлами ClickHouse Keeper.  Этот порт следует открывать, только если в кластере ClickHouse настроены выделенные узлы ClickHouse Keeper.	Взаимодействие узлов ClickHouse Keeper с остальными узлами кластера ClickHouse.
TCP	9234	Между выделенными узлами ClickHouse Keeper.  Этот порт следует открывать, только если в кластере ClickHouse настроены выделенные узлы ClickHouse Keeper.	Поддержание консенсуса в кластере ClickHouse по алгоритму Raft.

Протокол	Порт	Направление	Назначение подключения
TCP	10250	Между всеми узлами центрального кластера и кластера сателлита.	Выполнение запросов и мониторинг демона Kubelet.
TCP	10256	Между всеми узлами центрального кластера и кластера сателлита.	Сбор метрик kube-proxy.
TCP	10257	Между master-узлами центрального кластера.  Этот порт следует открывать, только если вы разворачиваете систему в отказоустойчивой конфигурации.	Управление распределением подов по узлам с помощью компонента kube-scheduler.
TCP	10259	Между master-узлами центрального кластера.  Этот порт следует открывать, только если вы разворачиваете систему в отказоустойчивой конфигурации.	Мониторинг состояния кластера с помощью компонента kube-controller-manager.
TCP	30000—32767	Между master-узлами центрального кластера и узлами кластера сателлита.  Номер порта для подключения к сателлиту указывается на этапе **NATS satellite setup** установки сателлита.	Управление сателлитом и запуск сервисов на нем.
		От агентов к worker-узлам центрального кластера или кластера сателлита.  Номер порта для подключения агентов — это порт шлюза, к которому подключаются агенты.	Получение данных от агентов на конечных устройствах.



Для подключения внешних систем через точки входа и конечные точки может понадобиться открытие дополнительных портов.

Порты и протоколы, используемые внешними системами, перечислены в разделах

Работа с точками входа и Работа с конечными точками.

Поддерживаемые внешние системы

R-Vision SIEM поддерживает следующие виды интеграции с внешними системами:

- Источники событий ИБ
- Интеграции для обработки событий ИБ

Источники событий ИБ

Список поддерживаемых внешних систем, которые можно использовать в точках входа и в конечных точках, приведен в разделе Поддерживаемые источники событий руководства по настройке источников на справочном портале R-Vision.



- Подробности о подключении и настройке источников событий приведены в разделе *Подключение источников*.
- Подробности о работе с точками входа и конечными точками приведены в разделах Работа с точками входа и Работа с конечными точками соответственно.

Интеграции для обработки событий ИБ

Система поддерживает интеграцию для обработки событий ИБ со следующими системами:

- R-Vision SOAR;
- почтовый сервер SMTP.



Подробности об интеграциях для обработки событий ИБ приведены в разделе Работа с интеграциями.

О версии 2.7

Технические особенности

Информация в данном разделе поможет предотвратить возможные ошибки при обновлении и эксплуатации продукта.

Особенности обновления

При обновлении системы до версии 2.7.1 необходимо выполнить ряд ручных действий. Подробности приведены в разделе Обновление системы.

Перевод правил корреляции в декларативный формат

В ближайших версиях продукта из функционала будут убраны императивные правила. Рекомендуется перевести все правила корреляции в декларативный формат. Подробности о данном формате приведены в разделе Декларативные правила.

Известные проблемы

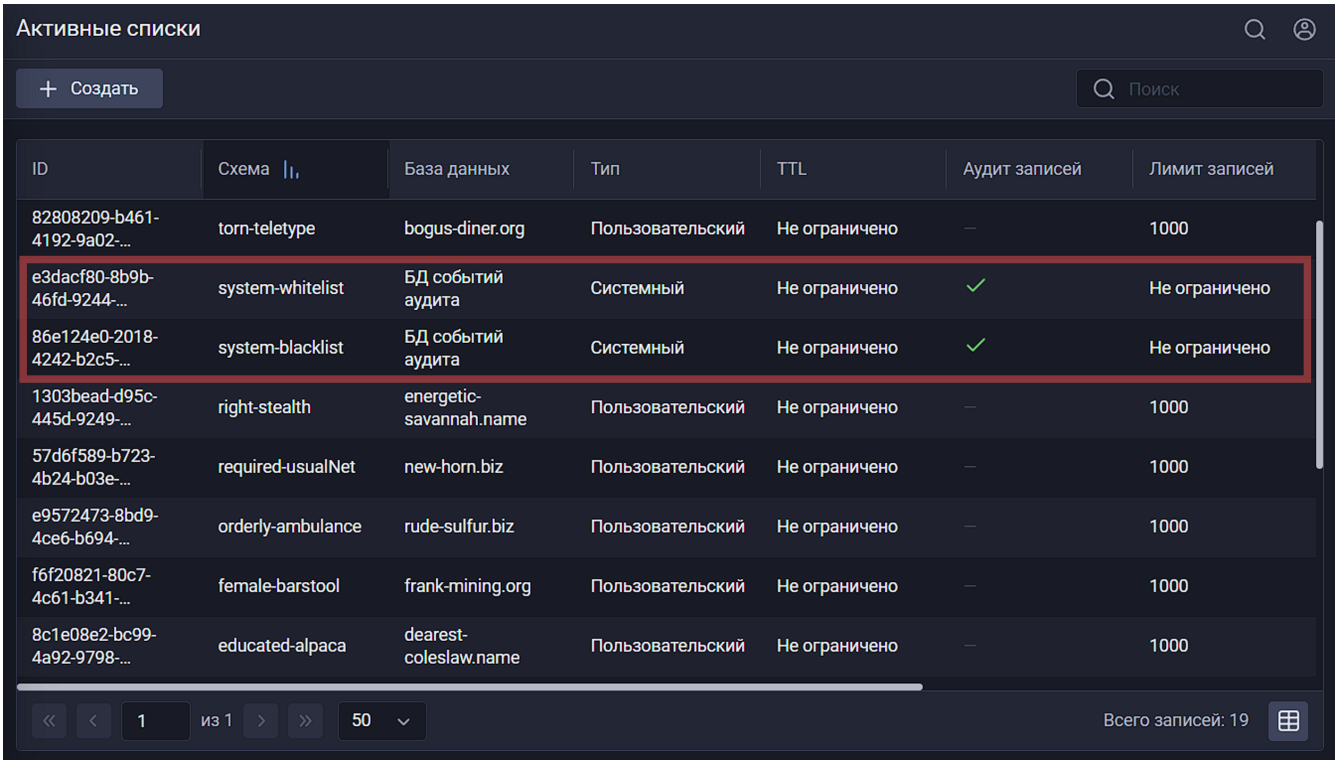
- В версиях 2.7.2 и ниже возможно возникновение ошибки при поиске данных с группировкой, если группа содержит поля типа `IPv4` или `IPv6` с пустыми значениями. Данная проблема будет исправлена в ближайших версиях продукта.
- Если изменить срок хранения в настройках томов хранилища событий, которое содержит большие объемы данных, окно изменения хранилища может перестать отвечать на запросы в течение некоторого времени, пока не истечет таймаут синхронного ожидания обработки. Таймаут может быть связан как с большим количеством данных, так и с проблемами в обработке очереди DDL-операций. Работа с очередью, а также с асинхронными и длительными операциями будет переработана в будущем.

Новые возможности

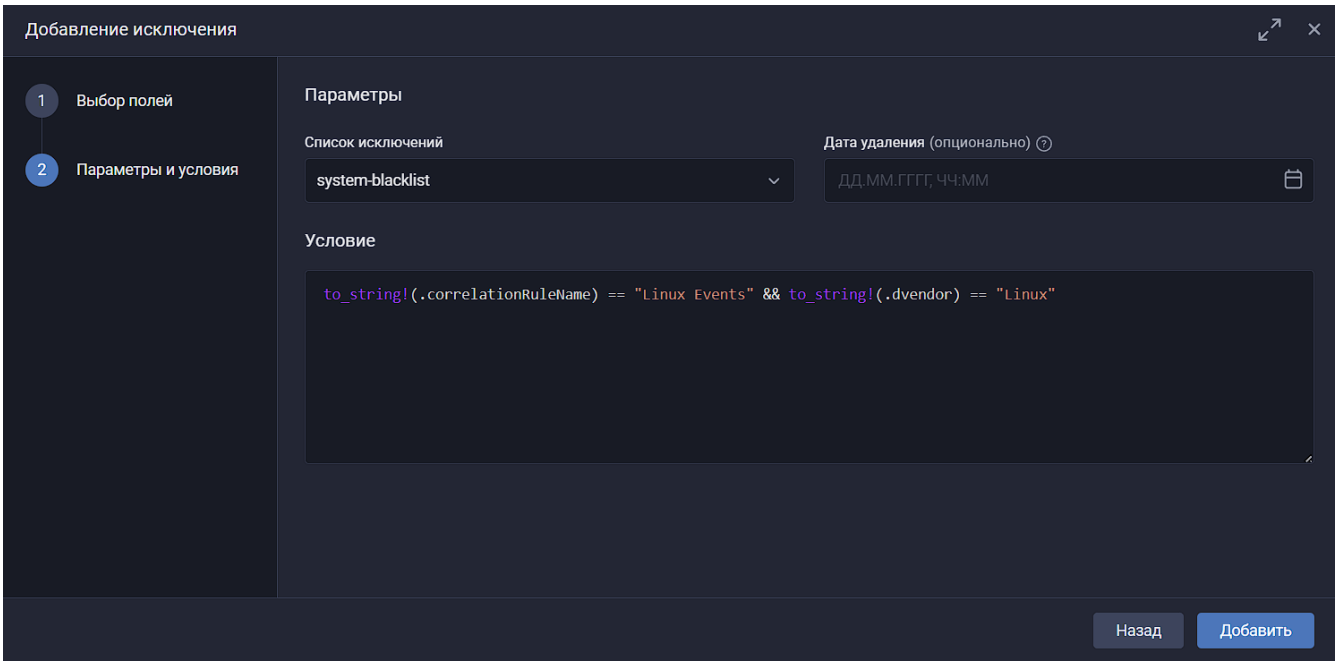
Исключения в правилах корреляции

В коррелляторах реализован механизм исключений с помощью системных активных списков:

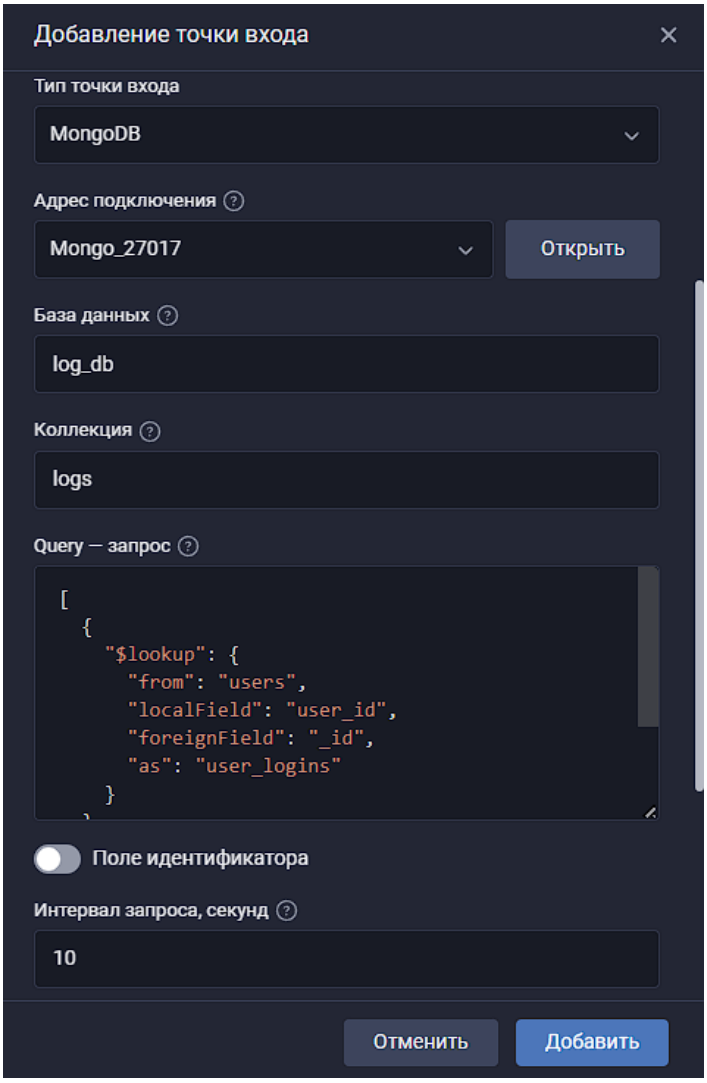
- **Черный список** — для определения подозрительных сущностей, которые необходимо проанализировать.
- **Белый список** — для указания доверенных сущностей, дальнейшая обработка которых не требуется.



Добавление записей в черный и белый списки осуществляется через раздел **Поиск** на основе корреляционных событий. Также записи могут быть добавлены вручную через раздел **Ресурсы** → **Активные списки**.



Точка входа MongoDB



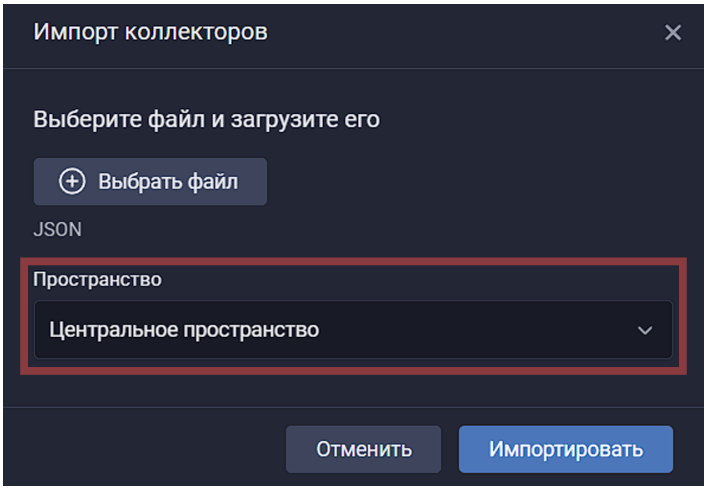
Добавлена точка входа MongoDB, позволяющая собирать данные из одноименной базы данных.

Данная точка входа может быть полезна, если требуется получить логи приложений, которые записываются в MongoDB, для их дальнейшей обработки в системе.

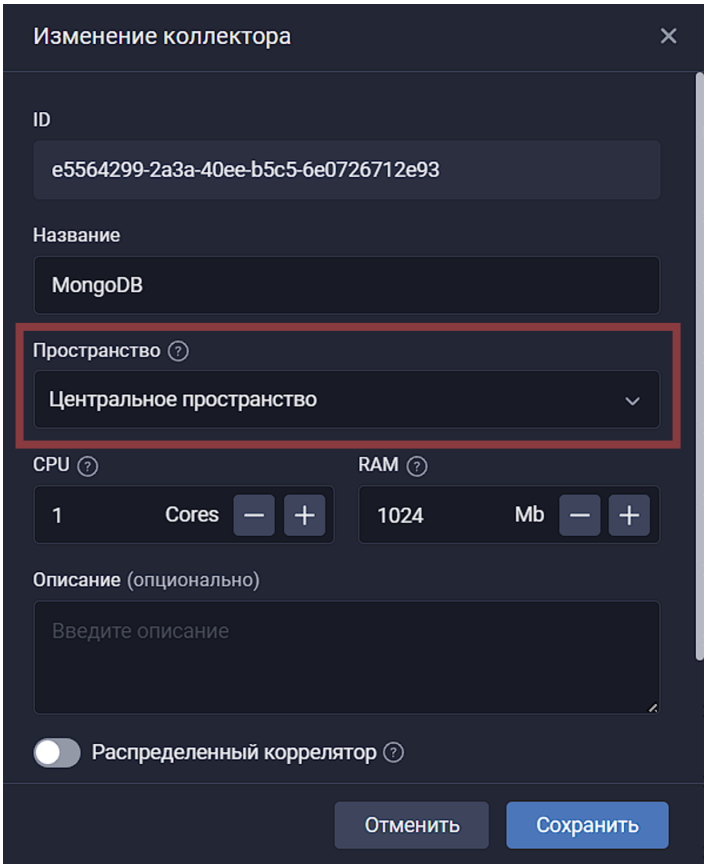
Выбор и изменение пространства у коллектора

Расширены возможности работы с пространством у коллектора. Теперь доступны следующие операции:

- Выбор подключения к пространству при импорте коллектора.



- Изменение выбранного подключения к пространству для выключенного коллектора.



Улучшения и доработки

Пользовательский интерфейс и опыт

- **Переработка ошибок хранилищ и БД**: теперь, когда получение данных о хранилищах и БД событий завершается с ошибками, в их карточках отображаются соответствующие сообщения.

Изменения в версии 2.7.1

- **Срок жизни для конкретных записей в активных списках**: добавлена возможность задавать срок жизни в записях активных списков с помощью поля `ttl` типа `integer`. Если поле принимает значение `0`, то запись становится бессрочной.

- **VRL в активных списках**: для полей активных списков добавлен тип данных `vrl_string`, поддерживающий задание и валидацию строк VRL-кода.
- **Разделение запросов по метрикам в коллекторе**: все метрики для коллектора и конвейера теперь запрашиваются системой независимо друг от друга. Это позволяет сохранить работоспособность коллекторов и конвейеров в случае ошибок при получении метрик.
- **Оригинальный IP-адрес источника**: для точек входа Socket и Syslog добавлена возможность получения оригинального IP-адреса источника событий. По умолчанию в событии сохраняется IP-адрес узла кластера Kubernetes.

Исправление проблем

- Исправлена проблема, при которой после удаления активного списка из системы в бакетах NATS Key/Value оставались его записи.
- Исправлена проблема, при которой для активных списков наблюдалось увеличение объема бакетов NATS Key/Value.
- Исправления в версии 2.7.1:
  - Исправлена проблема, при которой в коррелляторе происходила утечка памяти, если перед ним стояла точка входа R-Vision Endpoint.
  - Исправлена проблема, при которой в точках входа R-Vision Endpoint игнорировалась настройка **Максимум сообщений в обработке**.
  - Исправлена проблема, из-за которой для точек входа с активным сбором увеличивался размер бакета NATS Object Store, что вызывало повышенную нагрузку на NATS.
  - Исправлена проблема, из-за которой коллектор аудита не запускался при наличии символа `$` в имени пользователя, пароле пользователя или имени хоста ClickHouse.
  - Исправлена проблема, из-за которой в секретах типа **Контейнер PKCS** и **Контейнер PKCS#12** не сохранялся пароль приватного ключа.
- Исправления в версии 2.7.2:
  - Исправлена проблема, при которой система Prometheus могла потреблять все ресурсы на worker-узле, что приводило к его сбою. Теперь для Prometheus установлены лимиты по потреблению памяти.
  - Исправлена проблема, при которой для коллекторов, созданных в сателлитах, отсутствовали метрики по CPU и RAM.
  - Исправлена проблема, при которой получение больших объемов данных с помощью точки входа Database приводило к ошибке работы коллектора.
  - Исправлена проблема, при которой для точки входа Socket сбрасывались метрики, если между событиями проходило более 5 минут реального времени.
  - Исправлена проблема, при которой в коллекторах разных пространств нельзя было создать точки входа с одинаковым портом.
  - Исправлена проблема, при которой допускалась установка конфигурации конвейера, где элементы использовали правила экспертизы несуществующих версий.
  - Исправлена проблема, из-за которой поиск данных с группировкой по меткам времени, например `timestamp`, приводил к ошибке и найденные данные не отображались.

Развертывание системы

В данном разделе приведены инструкции по настройке окружения, установке компонентов системы и их первоначальной настройке, а также аппаратные и программные требования к серверам.

Чтобы установить систему, выполните следующие действия:

1. Убедитесь, что окружение, в котором будет установлена система, соответствует техническим требованиям.
2. Разверните кластер Kubernetes, в котором будет работать система.
   - Если установка производится в окружении РЕД ОС или Debian, вы можете воспользоваться автономным установщиком на базе Kubespray.
   - Для развертывания кластера в окружении Astra Linux, а также РЕД ОС с подключением к Интернету, обратитесь к разделу Установка кластера Kubernetes вручную.
   - Для развертывания кластера в окружении Альт Сервер или Альт СП Сервер обратитесь к официальной документации Альт Сервер.
3. Установите СУБД ClickHouse и PostgreSQL.



Установка ClickHouse в кластере Kubernetes допускается только для минимальной инсталляции системы в демонстрационных целях. Для обеспечения стабильной работы системы рекомендуется устанавливать ClickHouse в выделенном кластере.



Опционально выполните вынесенную установку PostgreSQL. Установка PostgreSQL на выделенном сервере или в выделенном кластере не является обязательной, однако рекомендуется для повышения стабильности работы системы и удобства обслуживания СУБД.

4. Выполните подготовительные действия перед установкой системы: настройте сетевой доступ к системе и кластеру, в котором она развернута, и скачайте дистрибутив.
5. Установите систему.
6. Если требуется организовать распределенный сбор и корреляцию событий в удаленных сегментах сети, вы можете установить сателлиты — вспомогательные экземпляры системы, развернутые в дополнительных удаленных кластерах.

Чтобы обновить систему, обратитесь к разделу Обновление системы.

Если в процессе установки или обновления системы возникнут неполадки, обратитесь к разделу Устранение проблем при установке или обновлении системы.

В разделе Развертывание агентов приведены инструкции по установке и настройке агентов, собирающих события на рабочих станциях.

Чтобы удалить систему, обратитесь к разделу Удаление системы.

Технические требования

В данном разделе представлены минимальные технические требования для установки и функционирования системы, в том числе для ее основного экземпляра и сателлитов.

- Программные требования
- Аппаратные требования

Программные требования

Поддерживаемые среды виртуализации

Система поддерживает любые среды виртуализации, которые удовлетворяют техническим требованиям и обеспечивают строгое резервирование ресурсов, а также прямой проброс аппаратных ресурсов в виртуальные машины.



- Для промышленной эксплуатации рекомендуется использовать гипервизоры bare-metal (Type 1), работающие непосредственно на аппаратном уровне. Использование hosted-гипервизоров (Type 2), функционирующих поверх операционной системы, не рекомендуется.
- Во избежание проблем в работе системы на виртуальных машинах (ВМ) под управлением гипервизора Hyper-V убедитесь, что в настройках всех ВМ кластера отключен режим совместимости. Для этого выполните следующие действия над каждой из ВМ:
  a. Перейдите в настройки ВМ.
  b. В дереве настроек выберите пункт **Processor** → **Совместимость**.
  c. Проверьте состояние флажка **Выполнить перенос на физический компьютер с другой версией процессора**. Если он установлен, выполните следующие действия:
     i. Если ВМ включена, выключите ее.
     ii. Снимите флажок **Выполнить перенос на физический компьютер с другой версией процессора**.
     iii. Сохраните настройки ВМ.
     iv. Включите ВМ.

Поддерживаемые операционные системы

Для основного экземпляра системы:

- Альт Сервер 11;
- Альт СП Сервер релиз 10;
- Astra Linux SE 1.7.2 и выше;
- РЕД ОС 7.3 и выше.



Поддерживаются как сертифицированные, так и несертифицированные варианты

ОС.

Например, для РЕД ОС поддерживаются следующие версии:

- несертифицированные: 7.3 и выше;
- сертифицированные: 7.3c и выше.

В случае сателлитов требования к операционной системе узлов кластера не предъявляются.

Требования к Kubernetes

- Минимальная версия Kubernetes: 1.24.
- Зарегистрированное полное доменное имя (FQDN) на каждом узле кластера.



Убедитесь, что FQDN узла разрешается в IP-адрес. Это можно проверить, например, с помощью утилиты nslookup:

```
nslookup <FQDN>
```

Здесь:

- `<FQDN>` — FQDN текущего узла.

Установка Kubernetes на узлы с адресами localhost.localdomain, localhost и 127.0.0.1 не предусмотрена.

- Поддерживаемая среда выполнения контейнеров (Container Runtime Interface, CRI): containerd, CRI-O.
- Установка сателлитов допускается на базе K0s.

Выбор среды выполнения имеет значение для автономной установки без Docker-registry в инфраструктуре.

Требования к СУБД

PostgreSQL

Для работы системы требуется реляционная СУБД:

- рекомендуется PostgreSQL версии 15 или выше;
- допустимо использование совместимых аналогов: Postgres Pro или Jatoba J4.

PostgreSQL может быть развернута **в кластере Kubernetes**, а также **на выделенном сервере или в выделенном кластере**. Настройка подключения к установленному экземпляру PostgreSQL производится на этапе **PostgreSQL setup** установки системы.



Вынесенная установка PostgreSQL не является обязательной, однако рекомендуется для повышения стабильности работы системы и удобства

обслуживания СУБД.

В случае установки PostgreSQL на одном выделенном сервере требования к операционной системе этого сервера не предъявляются.



Инструкция описывает процесс установки PostgreSQL 16 в выделенном кластере в ОС Debian 11 и РЕД ОС 7.3.

Для установки PostgreSQL согласно этой инструкции требуется:

- Для PostgreSQL — не менее двух выделенных узлов.
- Для хранилища конфигураций etcd — нечетное количество выделенных узлов, не менее трех. Рекомендуется использовать три узла.

ClickHouse

Для работы системы требуется СУБД ClickHouse.

Поддерживаемые версии ClickHouse:

- **До R-Vision SIEM 1.7.0**: ClickHouse 23.1.
- **R-Vision SIEM 1.7.0—1.8.0**: ClickHouse 23.1—23.2.
- **R-Vision SIEM 1.9.0—2.1.1**: ClickHouse 23.1—24.7.
- **R-Vision SIEM 2.2.0—2.4.1**: ClickHouse 24.7.
- **R-Vision SIEM 2.5.0—2.6.3**: ClickHouse 24.8.
- **Начиная с R-Vision SIEM 2.7.0**: ClickHouse 25.8.



Для корректной работы системы необходимо использовать ClickHouse поддерживаемой версии. Так, например, при обновлении системы с версии 2.1.1 до 2.5.0 требуется обновить ClickHouse до версии 24.8.

При использовании версии ClickHouse, отличной от указанной, корректная работа системы не гарантируется.

ClickHouse может быть развернута **в кластере Kubernetes** или **в выделенном кластере**. Настройка подключения к установленному экземпляру ClickHouse производится на этапе **ClickHouse setup** установки системы.



Установка ClickHouse в кластере Kubernetes допускается только для минимальной инсталляции системы в демонстрационных целях. Для обеспечения стабильной работы системы рекомендуется устанавливать ClickHouse в выделенном кластере.

На узлах, предназначенных для развертывания нового кластера ClickHouse, должна быть установлена одна из следующих операционных систем:

- Debian 11, 12;
- РЕД ОС 7.3, 8.0;

- Rocky Linux 8, 9.

Системные утилиты

Утилита	Минимальная версия	Наличие на узлах Kubernetes и АРМ администратора
ansible	2.9	Требуется.
bash	4.0	Требуется.
curl	Любая.	Требуется только на АРМ администратора.
diff	Любая.	Требуется только на АРМ администратора.
gunzip	Любая.	Требуется.
kubectl	Та же, что и версия Kubernetes, с отличием не более чем на один минор. Например, для Kubernetes 1.24 допустимы kubectl 1.23, 1.24 и 1.25.	Требуется только на АРМ администратора.
openssl	Любая.	Требуется только на АРМ администратора.
python	3.8	Требуется.
sed	Любая.	Требуется только на АРМ администратора.
tar	Любая.	Требуется только на АРМ администратора.

АРМ администратора — автоматизированное рабочее место администратора.

Поддерживаемые версии браузеров на рабочих станциях

- Google Chrome 110 и выше;
- Microsoft Edge 110 и выше;
- Mozilla Firefox 115 и выше.

Рекомендуется использовать наиболее актуальные версии браузеров.

Аппаратные требования

Требования для работы установщика

Для основного экземпляра системы: от 20 ГБ свободного места на диске.

Для сателлитов: от 10 ГБ свободного места на диске.

Требования для узлов кластера Kubernetes

Планирование ресурсов для развертывания кластера Kubernetes должно выполняться исходя из ожидаемой нагрузки и требуемого срока хранения данных в системе.



При развертывании кластера в отказоустойчивой конфигурации необходимо использовать нечетное количество master-узлов, не менее трех.

Необходимо использовать следующее аппаратное обеспечение:

- Процессоры:
  - Тактовая частота — не менее 2,5 ГГц в нормальном режиме работы и не менее 3,5 ГГц в режиме Boost.
  - Все ядра процессора должны быть только производительные.
  - Поддержка инструкций AVX и AVX2.



Чтобы проверить, поддерживает ли процессор инструкции AVX и AVX2, выполните следующую команду в терминале узла кластера:

```
grep -m1 '^flags' /proc/cpuinfo | awk '{
  f="sse3 ssse3 sse4_1 sse4_2 popcnt cx16 avx avx2";
  n=split(f, a, " ");
  for(i=1;i<=n;i++) if(index($0,a[i])==0){print "FAIL: missing "
a[i]; exit 1}
  print "CPU is OK: x86-64 v2 + AVX/AVX2"
}'
```

Если все необходимые инструкции поддерживаются, в результате выполнения команды будет выведена следующая строка:

```
CPU is OK: x86-64 v2 + AVX/AVX2
```

Иначе будет выведен список инструкций, поддержка которых отсутствует или не включена. В этом случае убедитесь, что процессор хоста виртуализации поддерживает указанные инструкции. Если процессор их поддерживает, то обратитесь за руководством по их включению к документации системы виртуализации, которую вы используете.

- Оперативная память: DDR4 с тактовой частотой не менее 3200 МГц в двухканальном или четырехканальном режиме.



При распределенной установке необходимо обеспечить сетевое взаимодействие между узлами кластера с пропускной способностью не менее 10 Гбит/с.

На узлах кластера следует использовать исключительно **твердотельные накопители (SSD) со скоростью чтения не менее 550 Мбит/с**. Рекомендуется

объединить SSD в массив RAID 1.

Требования для основного экземпляра системы

Минимальная конфигурация



Устанавливать систему в минимальной конфигурации следует только при ограниченности доступных вычислительных ресурсов. При такой установке возможны задержки в работе интерфейса и поиска по событиям, а также сбои системы.

При установке минимальной конфигурации допустимо использовать жесткие диски (HDD) со скоростью 7200 об./мин каждый и скоростью чтения и записи не менее 50 Мбит/с на узлах кластера, за исключением узла ClickHouse.

EPS (событий в секунду)	Конфигурация	Сетевое соединение
До 1000	Один сервер: 12vCPU, 20 ГБ ОЗУ, 500 ГБ на SSD	1 Гбит/с
1000—5000	Один сервер: 16vCPU, 24 ГБ ОЗУ, 500 ГБ на SSD	1 Гбит/с
5000—10 000	Один сервер: 18vCPU, 24 ГБ ОЗУ, 500 ГБ на SSD	1 Гбит/с

Рекомендуемая конфигурация

Рекомендуемая конфигурация — это основная конфигурация для стабильной работы системы.



Разделение требований к worker-узлам на роли **Коллектор** и **Другие приложения** носит логический характер и отражает распределение нагрузки. Это не означает необходимость выделения отдельных физических серверов для каждой из ролей. При достаточных ресурсах допускается размещение ролей на одном узле.

При размещении на одном узле необходимо предусмотреть ресурсы, достаточные для одновременной работы обеих ролей.

EPS (событий в секунду)	Конфигурация	Сетевое соединение
До 1000	Один сервер: 14vCPU, 28 ГБ ОЗУ, 500 ГБ на SSD	1 Гбит/с
1000—5000	Master: 4vCPU, 8 ГБ ОЗУ, 150 ГБ на SSD Worker (коллектор): 6vCPU, 8 ГБ ОЗУ, 150 ГБ на SSD Worker (другие приложения): 8vCPU, 12 ГБ ОЗУ, 500 ГБ на SSD ClickHouse: 1 узел, 24vCPU, 64 ГБ ОЗУ	1 Гбит/с

EPS (событий в секунду)	Конфигурация	Сетевое соединение
5000—10 000	Master: 4vCPU, 8 ГБ ОЗУ, 150 ГБ на SSD Worker (коллектор): 12vCPU, 16 ГБ ОЗУ, 150 ГБ на SSD Worker (другие приложения): 8vCPU, 12 ГБ ОЗУ, 500 ГБ на SSD ClickHouse: 1 узел, 24vCPU, 64 ГБ ОЗУ	1 Гбит/с
10 000—20 000	Master: 4vCPU, 8 ГБ ОЗУ, 150 ГБ на SSD Worker (коллектор): 16vCPU, 18 ГБ ОЗУ, 150 ГБ на SSD Worker (другие приложения): 8vCPU, 12 ГБ ОЗУ, 500 ГБ на SSD ClickHouse: 1 узел, 24vCPU, 64 ГБ ОЗУ	1 Гбит/с
20 000—30 000	Master: 4vCPU, 8 ГБ ОЗУ, 150 ГБ на SSD Worker (коллектор): 22vCPU, 18 ГБ ОЗУ, 150 ГБ на SSD Worker (другие приложения): 8vCPU, 12 ГБ ОЗУ, 500 ГБ на SSD ClickHouse: 1 узел, 24vCPU, 64 ГБ ОЗУ	1 Гбит/с
30 000—40 000	Master: 4vCPU, 8 ГБ ОЗУ, 150 ГБ на SSD Worker (коллектор): 28vCPU, 20 ГБ ОЗУ, 150 ГБ на SSD Worker (другие приложения): 8vCPU, 12 ГБ ОЗУ, 500 ГБ на SSD ClickHouse: 2 узла, 24vCPU, 64 ГБ ОЗУ	1 Гбит/с



При эксплуатации системы во избежание сбоев в ее работе избегайте заполнения дискового пространства на worker-узлах более чем на 80%.



Чтобы получить рекомендации по аппаратным требованиям к системе при EPS более 40 000 и сроке хранения данных более 12 месяцев, обратитесь в службу поддержки по адресу support@rvision.ru.

Рекомендации по разметке дискового пространства для основного экземпляра системы

Разметка диска при установке на один сервер

Раздел	Назначение раздела	Рекомендуемое свободное пространство
`/boot`	Ядро Linux, начальная файловая система initramfs, загрузчик.	Не менее 1 ГБ.
`/`	Системный раздел ОС.	Не менее 80 ГБ.
`/opt`	Хранение прикладного ПО, PVC и провайдера хранилища local-path-provisioner. Здесь находятся данные СУБД ClickHouse, PostgreSQL и других stateful-сервисов.	Не менее 220 ГБ.
`/var`	Хранение логов системы, собираемых демоном journald, runtime-файлов, кэша пакетов и временных данных сервисов.	Не менее 50 ГБ.
`/var/lib/containerd`	Хранение образов контейнеров.	Не менее 80 ГБ.
`/var/lib/kubelet`	Хранение данных Kubernetes: pod volumes, projected volumes, emptyDir, subPath, секретов, конфигураций.	Не менее 60 ГБ.
`/var/lib/etcd`	База данных control plane (плоскости управления) Kubernetes, в том числе все метаданные кластера.	Не менее 10 ГБ.
Всего		Не менее 500 ГБ.

Разметка диска для master-узла

Раздел	Назначение раздела	Рекомендуемое свободное пространство
`/boot`	Ядро Linux, начальная файловая система initramfs, загрузчик.	Не менее 1 ГБ.
`/`	Системный раздел ОС.	Не менее 50 ГБ.
`/opt`	Служебные файлы приложений, инсталляторы, вспомогательные компоненты.	Не менее 20 ГБ.
`/var`	Хранение логов системы и среды выполнения контейнеров (runtime) Kubernetes.	Не менее 30 ГБ.
`/var/lib/containerd`	Хранение образов control plane и системных контейнеров.	Не менее 40 ГБ.
`/var/lib/kubelet`	Хранение данных в томах подов control plane.	Не менее 5 ГБ.
`/var/lib/etcd`	Хранение информации о состоянии кластера.	Не менее 5 ГБ.

Раздел	Назначение раздела	Рекомендуемое свободное пространство
Всего		Не менее 150 ГБ.

Разметка диска для worker-узла

Раздел	Назначение раздела	Рекомендуемое свободное пространство
`/boot`	Ядро Linux, начальная файловая система initramfs, загрузчик.	Не менее 1 ГБ.
`/`	Системный раздел ОС.	Не менее 80 ГБ.
`/opt`	Хранение служебных данных приложений, PVC и провайдера хранилища local-path-provisioner.	Не менее 200 ГБ.
`/var`	Хранение логов системы и среды выполнения контейнеров (runtime) Kubernetes.	Не менее 20 ГБ.
`/var/lib/containerd`	Хранение образов приложений и слоев контейнеров сервисов (workloads).	Не менее 60 ГБ.
`/var/lib/kubelet`	Хранение данных в томах подов, в том числе emptyDir, projected volumes и subPath.	Не менее 40 ГБ.
Всего		Не менее 400 ГБ.

Разметка диска для выделенного узла ClickHouse

Раздел	Назначение раздела	Рекомендуемое свободное пространство
`/boot`	Ядро Linux, начальная файловая система initramfs, загрузчик.	Не менее 1 ГБ.
`/`	Системный раздел ОС.	Не менее 150 ГБ.
`/var/lib/clickhouse`	Основные данные ClickHouse: таблицы, партиции, таблица слияний (merges), метаданные, временные файлы.	В зависимости от поступающего в систему потока событий и желаемого срока их хранения.

Требования для сателлитов



Приведены требования для минимальной инсталляции сателлита, при которой возможна его нестабильная работа. Чтобы получить рекомендации по конфигурации сателлита для вашей инсталляции, обратитесь в службу поддержки по адресу support@rvision.ru.

Возможны следующие варианты инсталляции сателлита:

- **Моно** — кластер Kubernetes из одного узла, выполняющего роль master-узла и worker-узла одновременно.
- **Распределенный** — кластер из нескольких узлов, содержащий как минимум один master-узел и один worker-узел.



Если при распределенной установке сателлита в кластере имеется только один worker-узел, то к нему применяются требования, аналогичные моно-инсталляции.

Ресурс	Моно	Распределенный, master-узел	Распределенный, worker-узел
CPU, ядер	8	4	6
RAM, ГБ	8	4	6
Место на диске, ГБ	100	100	100

Развертывание кластера Kubernetes

До установки системы необходимо развернуть кластер Kubernetes.

- Если установка производится в окружении РЕД ОС или Debian, вы можете воспользоваться автономным установщиком на базе Kubespray.
- Для развертывания кластера в окружении Astra Linux, а также РЕД ОС с подключением к Интернету, обратитесь к разделу Установка кластера Kubernetes вручную.
- Для развертывания кластера в окружении Альт Сервер или Альт СП Сервер обратитесь к официальной документации Альт Сервер.

Установка кластера Kubernetes с помощью Kubespray



Это руководство описывает процесс офлайн-установки кластера Kubernetes с использованием автономного установщика на базе Kubespray. Офлайн-установка Kubernetes для основного экземпляра системы поддерживается для следующих операционных систем:

- РЕД ОС 7.3, 8.0;
- Debian 11, 12;
- Rocky Linux 9.

Для сателлитов поддерживается офлайн-установка кластера K0s на любых операционных системах семейства Linux.

Для установки в окружении других ОС с использованием подключения к Интернету обратитесь к разделу Установка кластера Kubernetes вручную.

Установку кластера можно производить как на master-узле кластера, так и на АРМ администратора — отдельном узле, имеющем сетевой доступ к узлам кластера.

Подготовка файлов установщика

Подготовка файлов установщика состоит из следующих этапов:

- Скачивание архива установщика
- Распаковка архива

Скачивание архива установщика

Для установки необходимо скачать архив установщика Kubernetes из файлового хранилища R-Vision:

- для РЕД ОС — `kubespray-installer-redos.tar.gz`;
- для Debian — `kubespray-installer-debian.tar.gz`;
- для Rocky Linux — `kubespray-installer-rocky.tar.gz`;
- для установки K0s в кластере сателлита — `k0s-installer.run`.



Если у вас есть скачанный установщик системы, архивы можно загрузить из его меню. Для этого:

1. Запустите установщик системы:

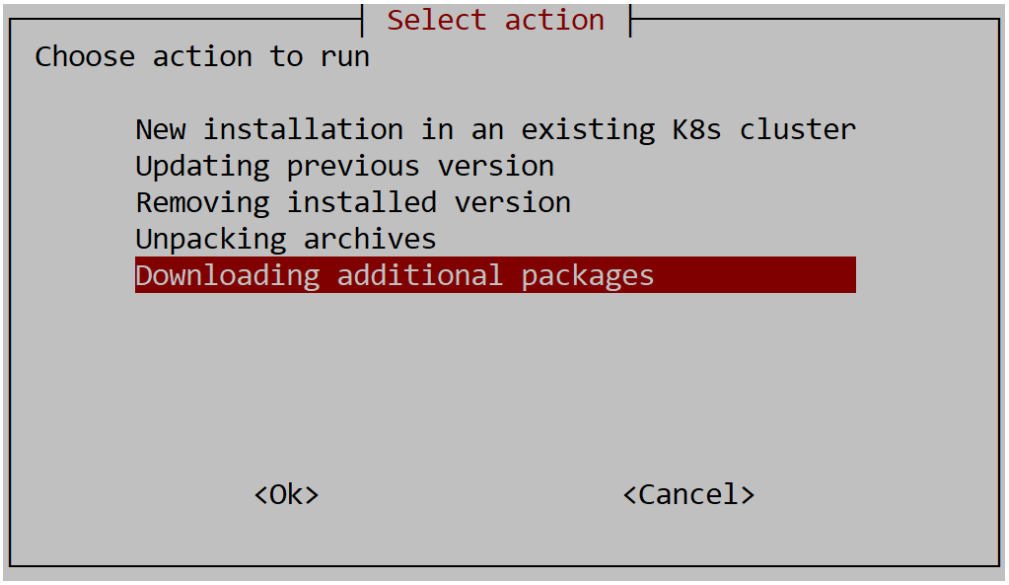
   ```
   bash ./evo.platform_<version>.run
   ```

   Требуемая версия bash: 4.0 или выше.

   Или сделайте файл установщика исполняемым и запустите непосредственно:

   ```
   chmod +x ./evo.platform_<version>.run
   ./evo.platform_<version>.run
   ```

2. Выберите режим работы **Downloading additional packages**.



3. Загрузите дополнительный пакет, например, `kubespray-installer-redos.tar.gz` или `k0s-installer.run`.

Архив с установочным файлом необходимо загрузить во временную директорию `/tmp`.

Распаковка архива

Выполните следующие шаги:

1. Загрузите архив установщика Kubernetes на узел, на котором будет производиться установка.
   - Для Windows:

     ```
     pscp <archive_path> user@server:<installation_dir>
     ```

   - Для Linux или macOS:

     ```
     scp <archive_path> user@server:<installation_dir>
     ```

   Здесь:

   - `<archive_path>` — путь к архиву установщика Kubernetes.
   - `<installation_dir>` — директория, в которой будет производиться установка (будет размещен файл дистрибутива `evo.platform_<version>.run`).

   Все дальнейшие действия выполняются на узле, на котором будет производиться установка.

2. Если вы используете архив `kubespray-installer-redos.tar.gz`, `kubespray-installer-debian.tar.gz` или `kubespray-installer-rocky.tar.gz`, распакуйте архив с установочными файлами с помощью следующих команд:

   ```
   cd <installation_dir>
   tar xvfz <archive_name>
   ```

   Здесь:

   - `<installation_dir>` — директория, в которой будет производиться установка (будет размещен файл дистрибутива `evo.platform_<version>.run`).
   - `<archive_name>` — имя архива, например, `kubespray-installer-redos.tar.gz`.

Установка кластера Kubernetes

Установка кластера Kubernetes состоит из следующих этапов:

- Отключение использования IPv6
- Настройка доменных имен узлов кластера

- Настройка SSH-доступа
- Установка утилит
- Синхронизация времени с NTP-сервером
- Запуск скрипта установки

Отключение использования IPv6

Чтобы избежать ошибок разрешения имен, перед развертыванием кластера отключите использование IPv6, выполнив следующие действия на каждом узле кластера:

1. В каталоге `/etc/sysctl.d` создайте файл `99-disable-ipv6.conf` со следующим содержимым:

   ```
   net.ipv6.conf.all.disable_ipv6 = 1
   net.ipv6.conf.default.disable_ipv6 = 1
   net.ipv6.conf.lo.disable_ipv6 = 1
   ```

2. Сохраните файл.
3. Примените изменения с помощью следующей команды:

   ```
   sudo sysctl --system
   ```

Настройка доменных имен узлов кластера

В процессе установки Kubespray автоматически изменяет доменную зону worker-узлов (часть доменного имени без имени хоста) на доменную зону первого master-узла в файле инвентаризации. Если имя хоста worker-узла совпадает с таковым у master-узла, после изменения доменной зоны полные доменные имена (FQDN) master- и worker-узла совпадут и установка завершится некорректно.



Пример:

- FQDN master-узла — `siem.master.local`;
- FQDN worker-узла — `siem.worker.local`.

В процессе установки FQDN worker-узла изменится на `siem.master.local`, что приведет к неработоспособности кластера.

Для корректной установки узлам требуется присвоить FQDN, имеющие единую доменную зону, но различные имена хостов. В примере выше:

- FQDN master-узла — `master.siem.local`;
- FQDN worker-узла — `worker.siem.local`.

Чтобы проверить и при необходимости изменить доменные имена узлов кластера, выполните следующие действия для каждого из них:

1. Получите доменное имя узла кластера, выполнив следующую команду:

```
hostname -f
```

2. Если имя хоста worker-узла совпадает с именем хоста master-узла, измените FQDN worker-узла, измените FQDN:
   - Если на узле установлена утилита `hostnamectl`:
     i. Измените FQDN с помощью `hostnamectl`:

     ```
     sudo hostnamectl set-hostname <new_fqdn>
     ```

     Здесь:

     - `<new_fqdn>` — новое FQDN узла.

     ii. Перезапустите узел.
   - Если на узле не установлена утилита `hostnamectl`:
     i. Измените FQDN с помощью `hostname`:

     ```
     sudo hostname <new_fqdn>
     ```

     Здесь:

     - `<new_fqdn>` — новое FQDN узла.

     Данная команда изменит FQDN узла только до перезагрузки, поэтому также необходимо выполнить последующие шаги.

     ii. Измените FQDN в файле `/etc/hostname`:

     ```
     sudo nano /etc/hostname
     ```

     iii. Измените FQDN в файле `/etc/hosts`:

     ```
     sudo nano /etc/hosts
     ```

Настройка SSH-доступа

Проверьте возможность подключения по SSH-ключу к каждому узлу кластера:

```
ssh <FQDN>
```

Здесь:

- <FQDN> — FQDN текущего узла.



Чтобы узнать FQDN текущего узла, выполните следующую команду:

```
hostname -f
```



Убедитесь, что FQDN узла разрешается в IP-адрес. Это можно проверить, например, с помощью утилиты nslookup:

```
nslookup <FQDN>
```

Здесь:

- <FQDN> — FQDN текущего узла.

Установка Kubernetes на узлы с адресами localhost.localdomain, localhost и 127.0.0.1 не предусмотрена.

Если подключение по SSH-ключу не настроено, сгенерируйте ключи и добавьте открытый ключ на каждый узел:

```
ssh-keygen
ssh-copy-id <user>@<FQDN>
```

Установка утилит

На каждом узле установите утилиты, требуемые для подключения к кластеру:

```
sudo dnf install bash-completion vim telnet chrony jq htop tcpdump zip unzip rsync
ebtables ethtool conntrack -y
```

Синхронизация времени с NTP-сервером

Настройте синхронизацию времени с корпоративным NTP-сервером, выполнив следующие команды на каждом узле кластера:

```
sudo timedatectl set-ntp true
sudo sed -i "/^server /d" /etc/chrony.conf
sudo bash -c 'echo "server <time_server>" >> /etc/chrony.conf'
sudo systemctl enable --now chronyd
sudo systemctl restart chronyd
sudo systemctl status chronyd
chronyc sources
```

Здесь:

- `<time_server>` — адрес корпоративного сервера времени (например, `ntp.rvision.local`).

Запуск скрипта установки

Скрипт принимает следующие переменные окружения:

Переменная	Описание
`ANSIBLE_USER`	Пользователь для подключения к узлам по SSH. Обязательная переменная.
`ANSIBLE_PASSWORD`	Пароль для подключения к узлам по SSH. Обязательная переменная.
`ASK_PASS`	Запрос пароля для подключения к узлам по SSH. Если необходимо запросить пароль, укажите `ASK_PASS=yes`.
`ASK_BECOME_PASS`	Запрос пароля для повышения привилегий до sudo. Если необходимо запросить пароль, укажите `ASK_BECOME_PASS=yes`.
`VERBOSE`	Уровень детализации логирования. Допустимые значения: целые числа `0`–`4`. По умолчанию: `0` (минимальный уровень детализации). Если передано любое недопустимое значение, устанавливается значение `2`.

Пример запуска только с обязательными переменными окружения

```
ANSIBLE_USER=username ANSIBLE_PASSWORD=password ./install.sh
```

Пример запуска с опциональными переменными окружения

```
ANSIBLE_USER=username ANSIBLE_PASSWORD=password ASK_PASS=yes ASK_BECOME_PASS=yes
VERBOSE=4 ./install.sh
```

По умолчанию скрипт установки запускается в интерактивном режиме, при котором отдельно запрашиваются FQDN для каждого узла кластера и путь до директории с данными containerd. Чтобы отключить интерактивность скрипта установки, необходимо сразу передать эти данные при его запуске с помощью следующих переменных окружения:

Переменная	Описание
`MASTER_NODES`	Список FQDN master-узлов через пробел.
`WORKER_NODES`	Список FQDN worker-узлов через пробел.

Переменная	Описание
`CONTAINERD_ROOT`	Путь к директории с данными containerd. По умолчанию: `/var/lib/containerd`.

Пример запуска в неинтерактивном режиме:

```
ANSIBLE_USER=username ANSIBLE_PASSWORD=password MASTER_NODES="m01.example.com"
WORKER_NODES="w01.example.com w02.example.com" CONTAINERD_ROOT="/var/lib/containerd"
./install.sh
```

Если пользователь не имеет права записи в корневой каталог, установщик необходимо запускать через `sudo`:

```
sudo ANSIBLE_USER=username ANSIBLE_PASSWORD=password MASTER_NODES="m01.example.com"
WORKER_NODES="w01.example.com w02.example.com" CONTAINERD_ROOT="/var/lib/containerd"
./install.sh
```

Проверка работоспособности

После завершения установки проверьте статус узлов кластера:

```
kubectl get nodes
```

Если все узлы кластера имеют статус `Ready`, значит, кластер успешно развернут.

Установка кластера Kubernetes вручную



Данное руководство описывает процесс установки кластера Kubernetes в окружении РЕД ОС и Astra Linux с использованием подключения к Интернету.

Для офлайн-установки в окружении РЕД ОС или Debian обратитесь к разделу Установка кластера Kubernetes с помощью Kubespray.

Для развертывания кластера в окружении Альт Сервер или Альт СП Сервер обратитесь к официальной документации Альт Сервер.

В данном руководстве установка кластера Kubernetes производится с использованием среды выполнения контейнеров containerd и сетевого плагина Calico.

- Можно использовать любую поддерживаемую среду выполнения из списка, приведенного в разделе технических требований.
- Использование альтернативных сетевых плагинов, таких как Flannel, **не поддерживается**.



Нижеприведенные действия выполняются на каждом узле кластера. Все узлы кластера должны быть подключены к Интернету.

Чтобы установить Kubernetes на узле кластера, выполните следующие действия:

1. Для корректной работы агентов kubelet отключите использование файлов подкачки и исключите их из файла `/etc/fstab`, чтобы их использование не возобновилось после перезагрузки узла:

```
swapoff -a && sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

2. Для автозагрузки модулей ядра, необходимых для работы сервиса containerd, в каталоге `/etc/modules-load.d` создайте файл `containerd.conf` со следующим содержимым:

```
overlay
br_netfilter
```

3. Загрузите модули ядра overlay и br_netfilter:

```
modprobe overlay
modprobe br_netfilter
```

4. Убедитесь, что модули загружены и работают, с помощью следующей команды:

```
lsmod | egrep "br_netfilter|overlay"
```

5. Для обеспечения работы внутренней сети Kubernetes в каталоге `/etc/sysctl.d/` создайте файл `99-kubernetes-cri.conf` со следующим содержимым:

```
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
```

6. Чтобы избежать ошибок разрешения имен, отключите использование IPv6. Для этого в каталоге `/etc/sysctl.d/` создайте файл `99-disable-ipv6.conf` со следующим содержимым:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

7. Примените изменения параметров sysctl, выполнив следующую команду:

```
sysctl --system
```

8. Установите пакеты, необходимые для работы Kubernetes. Список пакетов зависит от ОС, в окружении которой будет развернут кластер.

- Установка пакетов для РЕД ОС:

```
dnf install kubernetes kubernetes-kubeadm cri-tools tc ipvsadm ebtables socat
conntrack git curl wget runc containerd
```

- Установка пакетов для Astra Linux:

```
apt-get install kubelet kubeadm kubectl containerd cri-tools socat conntrack
```



Если некоторые из требуемых пакетов недоступны в репозитории производителя ОС, вы можете установить их из репозитория Kubernetes. Подробности приведены в официальной документации Kubernetes.

9. Установите утилиту nerdctl:
   a. Скачайте архив утилиты, выполнив следующую команду:

```
curl -O https://github.com/containerd/nerdctl/releases/download/v2.1.6/nerdctl-
2.1.6-linux-amd64.tar.gz
```

   b. Распакуйте архив в каталог `/usr/local/bin`, выполнив следующую команду:

```
tar -xzvf nerdctl-2.1.6-linux-amd64.tar.gz -C /usr/local/bin
```

10. Если на узле не установлена утилита iptables, установите ее согласно документации производителя ОС.
11. Разрешите проброс портов, выполнив следующую команду:

```
iptables -P FORWARD ACCEPT
```

12. Установите настройки по умолчанию для конфигурации контейнеров, выполнив следующую команду:

```
containerd config default | sudo tee /etc/containerd/config.toml
```

13. Разрешите использование контрольных групп (cgroup) Kubernetes:

```
sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g'
/etc/containerd/config.toml
```

14. Чтобы обеспечить работу сокета containerd, в каталоге `/etc` создайте файл `crictl.yaml`, если он не существует, и добавьте в него следующие строки:

```
runtime-endpoint: unix:///var/run/containerd/containerd.sock
image-endpoint: unix:///var/run/containerd/containerd.sock
timeout: 30
debug: false
```

15. Запустите службу containerd и добавьте ее в автозагрузку:

```
systemctl enable --now containerd
```

16. Добавьте службу kubelet в автозагрузку:

```
systemctl enable kubelet.service
```

17. На master-узлах кластера выполните следующие действия:

    a. Загрузите образы контейнеров, необходимых инструменту kubeadm для инициализации узла кластера:

```
kubeadm config images pull
```



Если доступ узла в Интернет осуществляется через прокси-сервер, необходимо скопировать его конфигурацию в файл конфигурации прокси containerd:

1. В каталоге `/etc/systemd/system/containerd.service.d` создайте файл `http-proxy.conf`, если он не существует.
2. Скопируйте в `http-proxy.conf` содержимое файла `/etc/environment`:

```
cat /etc/environment >
/etc/systemd/system/containerd.service.d/http-proxy.conf
```

После копирования добавьте исключения в список `NO_PROXY` файла `/etc/environment`.

Приведенный список исключений указан в качестве примера, для вашей сети он может отличаться. Ознакомьтесь с рекомендациями по настройке списка `NO_PROXY`.

```
10.96.0.0/12,10.233.0.0/18,10.244.0.0/16,.svc,.cluster.local
```

    b. Запустите инициализацию master-узла в одноранговом кластере.

```
kubeadm init --pod-network-cidr=<internal_cluster_net>
```

Здесь:

- `<internal_cluster_net>` — адрес внутренней подсети для кластера, например, `10.244.0.0/16`.

▼ *Пример вывода команды после успешной инициализации master-узла*

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each
as root:

kubeadm join 192.168.186.86:6443 --token kb0ejg.qtsdbljcnfcc3acd \
    --discovery-token-ca-cert-hash
sha256:92d275184e764076086f4cc86bca47f6ecedd46526568cc494d97dc27958bec7
```

c. Настройте параметры управления кластером:

- Для управления кластером от имени локального пользователя выполните следующие команды:

```
mkdir /home/$USER/.kube
cp -i /etc/kubernetes/admin.conf /home/$USER/.kube/config
chown $USER. /home/$USER/.kube /home/$USER/.kube/config
```

- Для управления кластером от имени суперпользователя (root) выполните следующие команды:

```
echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> /root/.bashrc
source .bashrc
export KUBECONFIG=/etc/kubernetes/admin.conf
```

d. Если настройка управления кластером производилась для локального пользователя, выйдите из сеанса пользователя root командой `exit` и дальнейшие действия по настройке кластера выполняйте от имени локального пользователя.

e. Установите внутреннюю конфигурацию сети в кластере, выполнив следующую команду:

```
kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.
yaml
```

f. Проверьте список и статус подов control plane (плоскости управления) кластера, выполнив следующую команду:

```
kubectl get pod -n kube-system
```

▼ *Пример вывода списка подов control plane кластера*

```
NAME                                      READY   STATUS    RESTARTS
AGE
calico-kube-controllers-6c99c8747f-cxxdr  1/1     Running   0
1m
calico-node-cxtrl                         1/1     Running   0
1m
coredns-5d78c9869d-phl6k                  1/1     Running   0
1m
coredns-5d78c9869d-w27xw                  1/1     Running   0
1m
etcd.test.rvision.local                   1/1     Running   0
1m
kube-apiserver.test.rvision.local         1/1     Running   0
1m
kube-controller-manager.test.rvision.local  1/1   Running   0
1m
kube-proxy-sw9qt                          1/1     Running   0
1m
kube-scheduler.test.rvision.local         1/1     Running   0
1m
```

g. Если установка выполняется на единственный узел, удалите все свойства taint с узла:

i. Получите список свойств taint узла, выполнив следующую команду:

```
kubectl get nodes <node_name> -o jsonpath='{.spec.taints}'
```

Здесь:

- `<node_name>` — имя узла.

▼ *Пример вывода списка свойств taint для узла*

```
[{"effect":"NoSchedule","key":"node-
role.kubernetes.io/master"},{"effect":"NoSchedule","key":"node-
role.kubernetes.io/control-plane"}]
```

ii. Удалите свойства taint с узла, выполнив для каждого свойства следующую команду:

```
kubectl taint node <node_name> <taint_key>-
```

Здесь:

- `<node_name>` — имя узла.
- `<taint_key>` — имя свойства, находящееся в поле `key` сведений о нем.

h. Установите дополнительные пакеты для инсталляции системы.

- Установка пакетов для РЕД ОС:

```
dnf install ansible yq jq
```

- Установка пакетов для Astra Linux:

```
apt-get install helm ansible yq jq
```



Если какие-либо из указанных пакетов недоступны в репозитории производителя ОС, установите их согласно официальной документации производителя ОС.

i. Для успешной установки менеджера логов Loki в составе системы измените имя сервиса kube-dns на coredns:

i. Создайте копию файла конфигурации сервиса kube-dns:

```
kubectl -n kube-system get svc kube-dns -o yaml > core.yaml
```

ii. Измените имя сервиса kube-dns на coredns:

```
sed -i 's/name: kube-dns/name: coredns/g' core.yaml
```

iii. Удалите существующий сервис kube-dns:

```
kubectl -n kube-system delete svc kube-dns
```

iv. Создайте сервис coredns, применив его конфигурацию:

```
kubectl apply -f core.yaml
```

j. Выведите команду для присоединения worker-узла, выполнив следующую команду:

```
kubeadm token create --print-join-command
```

▼ *Пример команды для присоединения worker-узла*

```
kubeadm join 192.168.186.86:6443 --token kb0ejg.qtsdbljcnfcc3acd --discovery
-token-ca-cert-hash
sha256:92d275184e764076086f4cc86bca47f6ecedd46526568cc494d97dc27958bec7
```

18. На каждом из worker-узлов выполните вышеуказанную команду для присоединения worker-узла к кластеру.

Установка СУБД

В данном разделе описаны действия по установке СУБД, необходимых для работы системы: ClickHouse и PostgreSQL.

Варианты установки СУБД

ClickHouse и PostgreSQL могут быть развернуты **в кластере Kubernetes**, а также **на выделенных серверах или в выделенных кластерах**.



Установка ClickHouse в кластере Kubernetes допускается только для минимальной инсталляции системы в демонстрационных целях. Для обеспечения стабильной работы системы рекомендуется устанавливать ClickHouse в выделенном кластере.



Вынесенная установка PostgreSQL не является обязательной, однако рекомендуется для повышения стабильности работы системы и удобства обслуживания СУБД.

Шифрование трафика между системой и кластером ClickHouse

Поддерживается шифрование трафика между системой и кластером ClickHouse (режим SSL). Для его работы необходимо сгенерировать TLS-сертификаты.

Система позволяет сгенерировать TLS-сертификаты автоматически в процессе установки или предоставить заранее созданные сертификаты.

Если вы планируете установку ClickHouse в кластере Kubernetes и использование заранее созданных сертификатов, разместите следующие файлы **на машине, с которой выполняется запуск установщика системы**:

Имя файла	Назначение
`ca.crt`	Корневой сертификат доверенного центра сертификации.
`tls.crt`	TLS-сертификат сервера.
`tls.key`	Приватный ключ TLS-сертификата сервера.



В поле **SAN (Subject Alternative Name)** генерируемых TLS-сертификатов необходимо указать следующие имена:

- `DNS: *.clickhouse.<namespace>.svc.cluster.local`;
- `DNS: clickhouse.<namespace>.svc.cluster.local`;
- `DNS: *.clickhouse-headless.<namespace>.svc.cluster.local`;
- `DNS: clickhouse-headless.<namespace>.svc.cluster.local`;
- `DNS: clickhouse`.

Здесь:

- `<namespace>` — название пространства имен, которое будет указано на этапе **Kubernetes cluster namespace to install** установки системы.

Абсолютные пути к вышеуказанным файлам потребуется ввести на этапе **ClickHouse setup** установки системы.

Если вы планируете вынесенную установку ClickHouse, за инструкциями по настройке TLS-сертификатов обратитесь к разделу Установка ClickHouse в выделенном кластере.

Вынесенная установка СУБД

Если планируется вынесенная установка ClickHouse и PostgreSQL, ознакомьтесь со следующими разделами:

- Установка ClickHouse в выделенном кластере
- Вынесенная установка PostgreSQL



Если планируется развертывание ClickHouse и PostgreSQL в распределенных кластерах, после их установки рекомендуется настроить балансировку запросов к ним. Методы балансировки запросов к СУБД описаны в разделе Настройка балансировки запросов к ClickHouse и PostgreSQL.

Иначе, если планируется установка ClickHouse и PostgreSQL в кластер Kubernetes, перейдите к разделу Подготовка к установке системы. Кластерная установка СУБД будет произведена автоматически в процессе установки системы.

Установка ClickHouse в выделенном кластере

В этом разделе описан процесс установки СУБД ClickHouse в выделенном кластере.

Поддерживаемые версии ClickHouse:

- **До R-Vision SIEM 1.7.0**: ClickHouse 23.1.
- **R-Vision SIEM 1.7.0—1.8.0**: ClickHouse 23.1—23.2.
- **R-Vision SIEM 1.9.0—2.1.1**: ClickHouse 23.1—24.7.
- **R-Vision SIEM 2.2.0—2.4.1**: ClickHouse 24.7.
- **R-Vision SIEM 2.5.0—2.6.3**: ClickHouse 24.8.
- **Начиная с R-Vision SIEM 2.7.0**: ClickHouse 25.8.



Для корректной работы системы необходимо использовать ClickHouse поддерживаемой версии. Так, например, при обновлении системы с версии 2.1.1 до 2.5.0 требуется обновить ClickHouse до версии 24.8.

При использовании версии ClickHouse, отличной от указанной, корректная работа системы не гарантируется.

Если у вас еще не развернут кластер ClickHouse, воспользуйтесь Ansible-плейбуком для автономной установки и настройки кластера с нуля.

Иначе, если у вас уже есть установленный кластер ClickHouse, настройте его конфигурацию для работы с системой.



Развернутый кластер ClickHouse включает следующие компоненты:

- **Шарды** — подмножества данных, расположенные на разных узлах кластера.
- **Реплики** — копии данных, позволяющие обеспечить их избыточность и тем самым повысить отказоустойчивость кластера.
- **ClickHouse Keeper** — служба, управляющая шардированием и репликацией данных в кластере.

Схема типовой архитектуры кластера ClickHouse приведена в составе схемы отказоустойчивой конфигурации системы.

Настройка нового кластера ClickHouse



Данная инструкция описывает установку и настройку кластера ClickHouse с нуля. При необходимости настроить уже существующий кластер ClickHouse, обратитесь к разделу Настройка существующего кластера ClickHouse.

Инструкция предлагает автоматизированный способ установки ClickHouse с помощью Ansible-плейбука.

На узлах, предназначенных для установки нового кластера ClickHouse, должна быть установлена одна из следующих операционных систем:

- Debian 11, 12;
- РЕД ОС 7.3, 8.0;
- Rocky Linux 8, 9.

Установку ClickHouse можно осуществлять как на одном из узлов кластера, так и на АРМ администратора — отдельном узле, имеющем сетевой доступ к узлам кластера. На АРМ администратора может быть установлена любая операционная система семейства Linux.



Вы можете установить ClickHouse самостоятельно, пользуясь инструкцией из официальной документации ClickHouse.

Если вы установили ClickHouse самостоятельно, перейдите к разделу Установка PostgreSQL на выделенном сервере.

Установка ClickHouse

Чтобы запустить установку:

1. Скачайте следующие архивы:
   - Архив с Ansible-плейбуком `clickhouse-cluster.tar.gz`.
   - Архив пакетов для подготовки к установке ClickHouse:

ОС узлов кластера	Ссылка на архив
Debian	files.rvision.ru/common/clickhouse/sys-prep-clickhouse-debian.tar.gz
РЕД ОС	files.rvision.ru/common/clickhouse/sys-prep-clickhouse-redos.tar.gz
Rocky Linux	files.rvision.ru/common/clickhouse/sys-prep-clickhouse-rocky.tar.gz



Если у вас есть скачанный установщик системы, архивы можно загрузить из его меню. Для этого:

1. Запустите установщик системы:

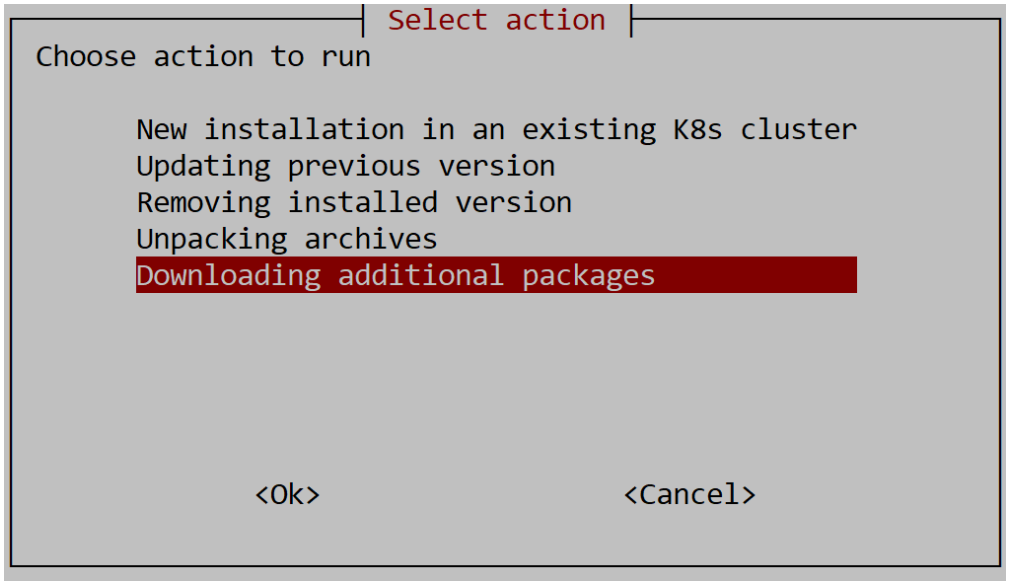
   ```
   bash ./evo.platform_<version>.run
   ```

   Требуемая версия bash: 4.0 или выше.

   Или сделайте файл установщика исполняемым и запустите непосредственно:

   ```
   chmod +x ./evo.platform_<version>.run
   ./evo.platform_<version>.run
   ```

2. Выберите режим работы **Downloading additional packages**.



3. Загрузите необходимый дополнительный пакет.

2. Распакуйте архив подготовительных пакетов:

```
mkdir -p ./sysprep
tar xvf ./<sysprep_archive> -C ./sysprep
```

Здесь:

- `<sysprep_archive>` — архив подготовительных пакетов для вашей ОС: `sys-prep-clickhouse-debian.tar.gz`, `sys-prep-clickhouse-redos.tar.gz` или `sys-prep-clickhouse-rocky.tar.gz`.

3. Запустите скрипт установки Ansible:

```
cd ./sysprep
./update-system.sh ansible
```

Скрипт установит Ansible и коллекции, требуемые для работы плейбука установки кластера ClickHouse.

4. Вернитесь в директорию, в которую был загружен архив с Ansible-плейбуком, и распакуйте его:

```
cd ..
tar xvf clickhouse-cluster.tar.gz
```



В дальнейшем полный путь к директории, в которую распакован архив плейбука, будет упоминаться как `<clickhouse_playbook_dir>`.

5. Перейдите в директорию `<clickhouse_playbook_dir>` и измените файл `inventory/hosts.ini`:

   a. В разделе `[clickhouse]` укажите полные доменные имена (FQDN) узлов, на которых будет развернут кластер ClickHouse.

b. В разделе `[clickhouse_keeper]` укажите FQDN узлов кластера, которым будет присвоена роль **Keeper**. Эта роль требуется для координации работы узлов.



Рекомендуется использовать нечетное количество узлов с ролью **Keeper**, не менее трех, поскольку для консенсуса используется алгоритм Raft, который принимает решения по правилу большинства.

c. В разделе `[clickhouse_shard<shard_id>]` укажите FQDN узлов, на которых будут размещены реплики шарда с номером `<shard_id>`. Каждый узел в списке соответствует одной реплике. Номера шардов должны быть неповторяющимися целыми числами, начиная с 0. Порядок номеров не важен.

▼ *Пример содержимого файла* `hosts.ini`

В данном примере настроен кластер ClickHouse из 5 узлов, 3 из которых имеют роль **Keeper**. Кластер состоит из 2 шардов, по 2 реплики для каждого.

```
[clickhouse]
node1.my.lan
node2.my.lan
node3.my.lan
node4.my.lan
node5.my.lan

[clickhouse_keeper]
node1.my.lan
node2.my.lan
node3.my.lan

[clickhouse_shard1]
node1.my.lan
node2.my.lan

[clickhouse_shard2]
node4.my.lan
node5.my.lan
```

6. Настройте подключение к кластеру и параметры сервера ClickHouse в файле `inventory/group_vars/clickhouse.yml`:

a. Настройте учетные данные для подключения системных пользователей к ClickHouse.



К паролям пользователей предъявляются следующие требования:

- Пароль может содержать только строчные латинские буквы (`a–z`), заглавные латинские буквы (`A–Z`) и цифры (`0–9`).
- Минимальная длина пароля: 8 знаков.

Эти пароли необходимо использовать на этапах настройки общесистемного экземпляра ClickHouse и ClickHouse для событий ИБ при установке системы.

В целях повышения безопасности при подключении к ClickHouse и управлении базой данных плейбук создает специального пользователя. Учетные данные этого пользователя задаются в следующих параметрах:

- `dcc_clickhouse_user` — имя пользователя.
- `dcc_clickhouse_password` — пароль пользователя.

Для работы системы необходимо, чтобы в ClickHouse были настроены три пользователя с разным набором прав:

- **owner** — пользователь с расширенным набором прав. Используется для создания, обновления и удаления таблиц (хранилищ событий), а также получения служебной информации.
- **writer** — пользователь с правами на запись. Используется в коллекторах для записи событий в ClickHouse.
- **reader** — пользователь с правами только на чтение. Используется для выполнения поисковых запросов.

▼ *Пример настроек данных пользователей в* `clickhouse.yml`

```
dcc_clickhouse_user: evo-user
dcc_clickhouse_password: EvoUserPassword
dcc_owner_password: OwnerPassword
dcc_writer_password: WriterPassword
dcc_reader_password: ReaderPassword
```

b. Если необходимо включить шифрование трафика между системой и кластером ClickHouse (режим SSL), добавьте в файл `inventory/group_vars/clickhouse.yml` следующие строки:

```
dcc_clickhouse_ssl_enabled: true
dcc_https_port: <https_port>
```

Здесь:

- `<https_port>` — порт для взаимодействия с ClickHouse по протоколу HTTPS. По умолчанию 8443.

Опция `dcc_clickhouse_ssl_enabled` включает прослушивание HTTPS-трафика по порту `dcc_https_port`. Также включается шифрование gRPC-трафика по порту `dcc_grpc_port`. При этом HTTP-порт `dcc_http_port` остается открытым, однако порт `dcc_grpc_port` принимает только зашифрованный трафик.

Подробное описание назначения портов `dcc_https_port`, `dcc_http_port` и `dcc_grpc_port` приведено в разделе Порты для взаимодействия системы и ClickHouse.

c. При необходимости настройте параметры кластера. Подробное описание параметров приведено в разделе Параметры кластера ClickHouse.



- Имена параметров следует записывать с префиксом `dcc`: например, `dcc_background_buffer_flush_schedule_pool_size`.
- Значения параметров по умолчанию можно просмотреть в файле `roles/clickhouse_cluster/defaults/main.yml`. Редактировать этот файл не рекомендуется.

▼ *Пример настроек в* `clickhouse.yml`

```
# Учетные данные пользователей ClickHouse.
dcc_clickhouse_user: evo-user
dcc_clickhouse_password: EvoUserPassword
dcc_owner_password: OwnerPassword
dcc_writer_password: WriterPassword
dcc_reader_password: ReaderPassword

# API-порт для взаимодействия системы и ClickHouse.
dcc_grpc_port: 9100
# HTTP-порт для взаимодействия системы и ClickHouse. По умолчанию порт 8123,
можно скорректировать (будет запрошен установщиком системы).
dcc_http_port: 8123
# HTTPS-порт для взаимодействия системы и ClickHouse. По умолчанию порт 8443,
можно скорректировать (будет запрошен установщиком системы).
dcc_https_port: 8443

# Параметры сервера ClickHouse.
dcc_background_buffer_flush_schedule_pool_size: 6
dcc_background_common_pool_size: 4
dcc_background_distributed_schedule_pool_size: 8
dcc_background_fetches_pool_size: 4
dcc_background_message_broker_schedule_pool_size: 8
dcc_background_move_pool_size: 4
dcc_background_pool_size: 8
dcc_background_merges_mutations_concurrency_ratio: 4
dcc_background_schedule_pool_size: 64
dcc_mark_cache_size: 5368709120
dcc_max_concurrent_queries: 0
dcc_max_server_memory_usage_to_ram_ratio: 0.9
dcc_max_server_memory_usage: 0
dcc_input_format_allow_errors_num: 0
dcc_input_format_allow_errors_ratio: 1
dcc_max_execution_time: 10
dcc_queue_max_wait_ms: 1000
dcc_connect_timeout_with_failover_ms: 750
dcc_connections_with_failover_max_tries: 2
dcc_skip_unavailable_shards: 1
dcc_insert_distributed_sync: 0
dcc_prefer_localhost_replica: true
dcc_allow_suspicious_ttl_expressions: 1

# Параметры таблиц MergeTree.
```

```
dcc_max_bytes_to_merge_at_max_space_in_pool: 2147483648
dcc_merge_max_block_size: 2048
dcc_number_of_free_entries_in_pool_to_lower_max_size_of_merge: 4
dcc_max_suspicious_broken_parts: 500

# Директории хранения данных.
dcc_hot_volume_path: /var/lib/clickhouse/hot-1/
dcc_cold_volume_path: /var/lib/clickhouse/cold-1/
```

7. Вернитесь в директорию, в которую загружены архивы подготовительных пакетов и плейбука.
8. Скопируйте архив подготовительных пакетов в директорию `files` распакованного архива плейбука:

   ```
   cp <sysprep_archive> <clickhouse_playbook_dir>/files
   ```

   Здесь:

   - `<sysprep_archive>` — архив подготовительных пакетов для вашей ОС: `sys-prep-clickhouse-debian.tar.gz`, `sys-prep-clickhouse-redos.tar.gz` или `sys-prep-clickhouse-rocky.tar.gz`.
9. Если вы включили опцию `dcc_clickhouse_ssl_enabled`, настройте TLS-сертификаты для работы ClickHouse в режиме SSL.

   По умолчанию роль `generate_certificates`, встроенная в плейбук, производит генерацию самоподписанных сертификатов. Все генерируемые сертификаты и ключи сохраняются в директории `<clickhouse_playbook_dir>/files/certs` на хосте, с которого производится запуск плейбука.

   Если у вас есть заранее подготовленные TLS-сертификаты, которые вы хотите использовать для ClickHouse, выполните следующие действия:

   a. В директории `<clickhouse_playbook_dir>/files` создайте директорию `certs`.
   b. Добавьте в директорию `certs` файлы сертификатов и ключей, соблюдая следующие правила именования:

Имя файла	Назначение
`ca-certificate.crt`	Сертификат доверенного центра сертификации.
`ca-certificate.key`	Ключ доверенного центра сертификации.  Данный ключ используется для выпуска и подписания сертификатов, которые генерируются ролью `generate_certificates` при работе плейбука. При использовании своих сертификатов создайте пустой файл с таким именем.

Имя файла	Назначение
`<node_fqdn>_server.crt`	Сертификат сервера, выпущенный доверенным центром сертификации. Сертификаты должны быть сгенерированы для каждого узла кластера ClickHouse.
`<node_fqdn>_server.key`	Приватный ключ для сертификата `<node_fqdn>_server.crt`. Приватные ключи должны быть сгенерированы для каждого узла кластера ClickHouse.

Здесь:

- `<node_fqdn>` — FQDN узла, указанное в файле `hosts.ini`.

При запуске плейбука с заранее подготовленными сертификатами генерация сертификатов производиться не будет. Произойдет копирование подготовленных сертификатов на узлы кластера.



Если установка системы будет производиться **не с того же хоста**, с которого выполняется развертывание ClickHouse, скопируйте сертификат доверенного центра сертификации `<clickhouse_playbook_dir>/files/certs/ca-certificate.crt` на этот хост.

Путь к сертификату центра сертификации на хосте, с которого производится установка системы, потребуется указать на этапе **ClickHouse setup** установки системы.

10. Перейдите в директорию распакованного архива и запустите плейбук установки:

```
cd <clickhouse_playbook_dir>
ansible-playbook -i inventory/hosts.ini playbooks/deploy_clickhouse_cluster.yml
```



Опциональные параметры запуска плейбука:

- `-u <user>` — имя пользователя, от имени которого будет создана SSH-сессия для развертывания ClickHouse.
- `-K` — запрос пароля, если для выполнения операций от имени пользователя, который производит развертывание, требуется повышение привилегий.
- `-vvv` — расширенное логирование.



После установки кластера ClickHouse рекомендуется настроить балансировку запросов к его узлам. Методы балансировки запросов к СУБД описаны в разделе Настройка балансировки запросов к ClickHouse и PostgreSQL.

Параметры кластера ClickHouse

Порты для взаимодействия системы и ClickHouse

API-порт для взаимодействия системы и ClickHouse задается в параметре `dcc_grpc_port`. По умолчанию установлен порт 9100.

Для взаимодействия системы и ClickHouse по протоколу HTTP по умолчанию используется порт 8123. Его можно изменить в параметре `dcc_http_port`.

Для взаимодействия системы и ClickHouse по протоколу HTTPS по умолчанию используется порт 8443. Его можно изменить в параметре `dcc_https_port`.

Параметры ClickHouse

В зависимости от планируемой нагрузки на систему и доступных вычислительных ресурсов вы можете настроить параметры сервера ClickHouse и таблиц MergeTree, перечисленные в разделе Справка о параметрах ClickHouse.

▼ *Пример настройки параметров ClickHouse*

```
dcc_background_buffer_flush_schedule_pool_size: 6
dcc_background_common_pool_size: 4
dcc_background_distributed_schedule_pool_size: 8
dcc_background_fetches_pool_size: 4
dcc_background_message_broker_schedule_pool_size: 8
dcc_background_move_pool_size: 4
dcc_background_pool_size: 8
dcc_background_merges_mutations_concurrency_ratio: 4
dcc_background_schedule_pool_size: 64
dcc_mark_cache_size: 5368709120
dcc_max_concurrent_queries: 0
dcc_max_server_memory_usage_to_ram_ratio: 0.9
dcc_max_server_memory_usage: 0
dcc_input_format_allow_errors_num: 0
dcc_input_format_allow_errors_ratio: 1
dcc_max_execution_time: 10
dcc_queue_max_wait_ms: 1000
dcc_connect_timeout_with_failover_ms: 750
dcc_connections_with_failover_max_tries: 2
dcc_skip_unavailable_shards: 1
dcc_insert_distributed_sync: 0
dcc_prefer_localhost_replica: true
dcc_allow_suspicious_ttl_expressions: 1
```

▼ *Пример настройки параметров таблиц MergeTree*

```
dcc_max_bytes_to_merge_at_max_space_in_pool: 2147483648
dcc_merge_max_block_size: 2048
dcc_number_of_free_entries_in_pool_to_lower_max_size_of_merge: 4
dcc_max_suspicious_broken_parts: 500
```

Директории хранения данных

Директории хранения данных ClickHouse задаются в следующих параметрах:

- `dcc_hot_volume_path` — директория для хранения часто используемых данных (горячее хранение);
- `dcc_cold_volume_path` — директории для хранения редко используемых данных (холодное хранение).

Если хранение событий планируется в корневую директорию ClickHouse, то оставьте настройки по умолчанию.

Пример настройки директорий хранения

```
dcc_hot_volume_path: /var/lib/clickhouse/hot-1/
dcc_cold_volume_path: /var/lib/clickhouse/cold-1/
```

Если планируется хранить события в отдельной директории, то:

1. Создайте необходимые директории под горячее и холодное хранение данных с помощью команды:

   ```
   mkdir <dir_path>
   ```

   Здесь:

   - `<dir_path>` — путь до директории горячего или холодного хранения.
2. Выдайте права на созданные директории группе пользователей clickhouse с помощью команды:

   ```
   chown -R clickhouse:clickhouse <dir_path>
   ```

   Здесь:

   - `<dir_path>` — путь до директории горячего или холодного хранения.
3. Укажите пути до созданных директорий в параметрах `dcc_hot_volume_path` и `dcc_cold_volume_path` в файле `clickhouse.yml`.

Пример настройки отдельных директорий

Например, необходимо хранить события ClickHouse в директориях `/storage/hot` (горячее хранение) и `/storage/cold` (холодное хранение).

Чтобы настроить хранение в этих директориях:

1. Создайте директории с помощью команд:

   ```
   mkdir /storage/hot
   mkdir /storage/cold
   ```

2. Выдайте права на директории группе пользователей clickhouse с помощью команд:

```
chown -R clickhouse:clickhouse /storage/hot
chown -R clickhouse:clickhouse /storage/cold
```



Если в директории `/storage` планируется хранение только данных событий ClickHouse, то, вместо выдачи прав на каждую поддиректорию по отдельности, вы можете выдать права сразу на всю директорию `/storage`:

```
chown -R clickhouse:clickhouse /storage
```

3. Укажите пути до директорий в параметрах `dcc_hot_volume_path` и `dcc_cold_volume_path` в файле `clickhouse.yml`:

```
dcc_hot_volume_path: /storage/hot/
dcc_cold_volume_path: /storage/cold/
```

Настройка существующего кластера ClickHouse



Данная инструкция описывает настройку уже существующего кластера ClickHouse. Для установки и настройки кластера с нуля обратитесь к разделу Настройка нового кластера ClickHouse.

Настройка существующего кластера ClickHouse состоит из следующих этапов:

1. Настройка конфигурации кластера ClickHouse
2. Запуск кластера ClickHouse
3. Создание пользователей ClickHouse

Настройка конфигурации кластера ClickHouse



Действия выполняются на каждом узле кластера.

Чтобы настроить конфигурацию кластера ClickHouse, на каждом узле кластера перейдите в директорию `config.d` в директории установки сервера ClickHouse и настройте параметры кластера в конфигурационном файле `config.xml`.

▼ *Пример содержимого файла* `config.xml`

```
<clickhouse>
    <access_control_improvements>

<users_without_row_policies_can_read_rows>true</users_without_row_policies_can_read_
rows>

<on_cluster_queries_require_cluster_grant>false</on_cluster_queries_require_cluster_
```

```
grant>

<select_from_system_db_requires_grant>false</select_from_system_db_requires_grant>

<select_from_information_schema_requires_grant>false</select_from_information_schema
_requires_grant>

<settings_constraints_replace_previous>false</settings_constraints_replace_previous>
        <role_cache_expiration_time_seconds>600</role_cache_expiration_time_seconds>
    </access_control_improvements>
    <jdbc_bridge>
        <host>127.0.0.1</host>
        <port>9019</port>
    </jdbc_bridge>
    <!-- API-порт для взаимодействия системы и ClickHouse. -->
    <grpc_port>9100</grpc_port>
    <!-- HTTP-порт для взаимодействия системы и ClickHouse. -->
    <http_port>8123</http_port>
    <!-- HTTPS-порт для взаимодействия системы и ClickHouse. -->
    <https_port>8443</http_port>
    <!-- Настройки сертификатов для шифрования gRPC-трафика между системой и
ClickHouse. -->
    <grpc>
        <!-- Включение режима SSL. Измените на true, если необходимо шифрование
трафика между системой и кластером ClickHouse. -->
        <enable_ssl>false</enable_ssl>
        <!-- Абсолютный путь к сертификату сервера, выпущенному доверенным центром
сертификации. -->
        <ssl_cert_file>/etc/clickhouse-server/certs/server.crt</ssl_cert_file>
        <!-- Абсолютный путь к приватному ключу сертификата сервера. -->
        <ssl_key_file>/etc/clickhouse-server/certs/server.key</ssl_key_file>
        <!-- Требование аутентификации клиента по протоколу gRPC. Обязательно false.
-->
        <ssl_require_client_auth>false</ssl_require_client_auth>
        <!-- Абсолютный путь к сертификату доверенного центра сертификации. -->
        <ssl_ca_cert_file>/etc/clickhouse-server/certs/ca-
certificate.crt</ssl_ca_cert_file>
    </grpc>
    <!-- Настройки сертификатов для шифрования HTTPS-трафика между системой и
ClickHouse. -->
    <openSSL>
        <server>
            <verificationMode>none</verificationMode>
            <loadDefaultCAFile>true</loadDefaultCAFile>
            <cacheSessions>true</cacheSessions>
            <disableProtocols>sslv2,sslv3</disableProtocols>
            <preferServerCiphers>true</preferServerCiphers>
            <!-- Абсолютный путь к сертификату сервера, выпущенному доверенным
центром сертификации. -->
            <certificateFile>/etc/clickhouse-
server/certs/server.crt</certificateFile>
```

```
            <!-- Абсолютный путь к приватному ключу сертификата сервера. -->
            <privateKeyFile>/etc/clickhouse-server/certs/server.key</privateKeyFile>
            <!-- Абсолютный путь к сертификату доверенного центра сертификации. -->
            <caConfig>/etc/clickhouse-server/certs/ca-certificate.crt</caConfig>
            <invalidCertificateHandler>
                <name>RejectCertificateHandler</name>
            </invalidCertificateHandler>
        </server>

        <client>
            <!-- Абсолютный путь к сертификату сервера, выпущенному доверенным
центром сертификации. -->
            <certificateFile>/etc/clickhouse-
server/certs/server.crt</certificateFile>
            <!-- Абсолютный путь к приватному ключу сертификата сервера. -->
            <privateKeyFile>/etc/clickhouse-server/certs/server.key</privateKeyFile>
            <!-- Абсолютный путь к сертификату доверенного центра сертификации. -->
            <caConfig>/etc/clickhouse-server/certs/ca-certificate.crt</caConfig>
            <loadDefaultCAFile>true</loadDefaultCAFile>
            <cacheSessions>true</cacheSessions>
            <disableProtocols>sslv2,sslv3</disableProtocols>
            <preferServerCiphers>true</preferServerCiphers>
            <invalidCertificateHandler>
                <name>RejectCertificateHandler</name>
            </invalidCertificateHandler>
        </client>
    </openSSL>
    <allow_plaintext_password>1</allow_plaintext_password>
    <allow_no_password>1</allow_no_password>
    <allow_implicit_no_password>1</allow_implicit_no_password>
    <timezone>UTC</timezone>
    <interserver_http_host>localhost</interserver_http_host>

<database_atomic_delay_before_drop_table_sec>10</database_atomic_delay_before_drop_t
able_sec>
    <!-- Шардирование и репликация узлов кластера. -->
    <remote_servers>
        <default>
            <shard>
                <internal_replication>true</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>IP или FQDN узла 1</host>
                    <port>9000</port>
                    <user>writer</user>
                    <password>writer</password>
                </replica>
            </shard>
            <shard>
                <internal_replication>true</internal_replication>
                <weight>1</weight>
```

```
                <replica>
                    <host>IP или FQDN узла 2</host>
                    <port>9000</port>
                    <user>writer</user>
                    <password>writer</password>
                </replica>
            </shard>
        </default>
    </remote_servers>
    <macros>
        <cluster>default</cluster>
        <replica>IP или FQDN текущего узла</replica>
        <shard>Имя текущего узла</shard>
        <layer>clickhouse</layer>
    </macros>
    <prometheus>
        <endpoint>/metrics</endpoint>
        <port>9363</port>
        <metrics>true</metrics>
        <events>true</events>
        <asynchronous_metrics>true</asynchronous_metrics>
    </prometheus>
    <listen_host>::</listen_host>
    <!-- Параметры сервера ClickHouse. -->

<background_buffer_flush_schedule_pool_size>6</background_buffer_flush_schedule_pool
_size>
    <background_common_pool_size>4</background_common_pool_size>

<background_distributed_schedule_pool_size>8</background_distributed_schedule_pool_s
ize>
    <background_fetches_pool_size>4</background_fetches_pool_size>

<background_message_broker_schedule_pool_size>8</background_message_broker_schedule_
pool_size>
    <background_move_pool_size>4</background_move_pool_size>
    <background_pool_size>8</background_pool_size>

<background_merges_mutations_concurrency_ratio>4</background_merges_mutations_concur
rency_ratio>
    <background_schedule_pool_size>64</background_schedule_pool_size>
    <mark_cache_size>5368709120</mark_cache_size>
    <max_concurrent_queries>0</max_concurrent_queries>
    <max_server_memory_usage_to_ram_ratio>0.9</max_server_memory_usage_to_ram_ratio>
    <max_server_memory_usage>0</max_server_memory_usage>
    <!-- Параметры таблиц MergeTree. -->
    <merge_tree>

<max_bytes_to_merge_at_max_space_in_pool>2147483648</max_bytes_to_merge_at_max_space
_in_pool>
        <merge_max_block_size>2048</merge_max_block_size>
```

```
<number_of_free_entries_in_pool_to_lower_max_size_of_merge>4</number_of_free_entries
_in_pool_to_lower_max_size_of_merge>
        <max_suspicious_broken_parts>500</max_suspicious_broken_parts>
    </merge_tree>
    <!-- Директории хранения данных. -->
    <storage_configuration>
        <disks>
            <hot_disk_1>
                <path>/var/lib/clickhouse/hot-1/</path>
            </hot_disk_1>
            <cold_disk_1>
                <path>/var/lib/clickhouse/cold-1/</path>
            </cold_disk_1>
        </disks>
        <policies>
            <streams_storage_policy>
                <volumes>
                    <hot_volume>
                        <disk>hot_disk_1</disk>
                    </hot_volume>
                    <cold_volume>
                        <disk>cold_disk_1</disk>
                    </cold_volume>
                </volumes>
            </streams_storage_policy>
        </policies>
    </storage_configuration>
    <users>
        <root>
            <access_management>1</access_management>
        </root>
    </users>
    <!-- Настройка ClickHouse Keeper. -->
    <keeper_server>
        <tcp_port>9181</tcp_port>
        <server_id>ID Текущего узла</server_id>
        <log_storage_path>/var/lib/clickhouse/coordination/log</log_storage_path>

<snapshot_storage_path>/var/lib/clickhouse/coordination/snapshots</snapshot_storage_
path>
        <coordination_settings>
            <operation_timeout_ms>10000</operation_timeout_ms>
            <session_timeout_ms>30000</session_timeout_ms>
            <raft_logs_level>trace</raft_logs_level>
        </coordination_settings>
        <raft_configuration>
            <server>
                <id>ID текущего узла</id>
                <hostname>FQDN текущего узла</hostname>
                <port>9234</port>
```

```
            </server>
        </raft_configuration>
    </keeper_server>
    <distributed_ddl>
        <path>/var/lib/clickhouse/task_queue/ddl</path>
    </distributed_ddl>
    <profiles>
        <default>
            <input_format_allow_errors_num>0</input_format_allow_errors_num>
            <input_format_allow_errors_ratio>1</input_format_allow_errors_ratio>
            <max_execution_time>10</max_execution_time>
            <queue_max_wait_ms>1000</queue_max_wait_ms>
            <connect_timeout_with_failover_ms>750</connect_timeout_with_failover_ms>

<connections_with_failover_max_tries>2</connections_with_failover_max_tries>
            <skip_unavailable_shards>1</skip_unavailable_shards>
            <insert_distributed_sync>0</insert_distributed_sync>
            <prefer_localhost_replica>1</prefer_localhost_replica>
            <allow_suspicious_ttl_expressions>1</allow_suspicious_ttl_expressions>
        </default>
        <readonly>
            <readonly>1</readonly>
        </readonly>
    </profiles>
</clickhouse>
```

Порты для взаимодействия системы и ClickHouse

API-порт для взаимодействия системы и ClickHouse задается в параметре `grpc_port`. По умолчанию установлен порт 9100.

Для взаимодействия системы и ClickHouse по протоколу HTTP по умолчанию используется порт 8123. Его можно изменить в параметре `http_port`.

Для взаимодействия системы и ClickHouse по протоколу HTTPS по умолчанию используется порт 8443. Его можно изменить в параметре `https_port`.

Настройка шифрования трафика между системой и кластером ClickHouse

Если необходимо включить шифрование трафика между системой и кластером ClickHouse (режим SSL), выполните следующие действия:

1. Сгенерируйте следующие файлы:
   - Сертификат доверенного центра сертификации с расширением `.crt`.
   - Сертификат сервера ClickHouse, выпущенный доверенным центром сертификации, с расширением `.crt`.
   - Приватный ключ сертификата сервера ClickHouse с расширением `.key`.
2. Поместите вышеуказанные файлы в директорию `/etc/clickhouse-server/certs` на сервере ClickHouse.

3. Добавьте параметры в файл `config.xml`:
   - `<https_port>` — HTTPS-порт для взаимодействия системы и ClickHouse. По умолчанию 8443.
   - `<grpc>` — блок параметров для шифрования gRPC-трафика между системой и ClickHouse:
     - `<enable_ssl>` — включение режима SSL. Установите значение `true`.
     - `<ssl_cert_file>` — абсолютный путь к сертификату сервера, выпущенному доверенным центром сертификации.
     - `<ssl_key_file>` — абсолютный путь к приватному ключу сертификации.
     - `<ssl_require_client_auth>` — требование аутентификации клиента. Установите значение `false`.

       Система не требует аутентификации клиента по протоколу gRPC.

     - `<ssl_ca_cert_file>` — абсолютный путь к сертификату доверенного центра сертификации.
   - `<openSSL>` — блок параметров для шифрования HTTPS-трафика между системой и ClickHouse:
     - `<certificateFile>` — абсолютный путь к сертификату сервера, выпущенному доверенным центром сертификации.
     - `<privateKeyFile>` — абсолютный путь к приватному ключу сертификации.
     - `<caConfig>` — абсолютный путь к сертификату доверенного центра сертификации.

       Остальные параметры в блоке должны иметь те же значения, что и в примере ниже.

▼ *Пример настроек режима SSL в файле* `config.xml`

```
<https_port>8443</http_port>
<grpc>
    <enable_ssl>true</enable_ssl>
    <ssl_cert_file>/etc/clickhouse-server/certs/server.crt</ssl_cert_file>
    <ssl_key_file>/etc/clickhouse-server/certs/server.key</ssl_key_file>
    <ssl_require_client_auth>false</ssl_require_client_auth>
    <ssl_ca_cert_file>/etc/clickhouse-server/certs/ca-
certificate.crt</ssl_ca_cert_file>
</grpc>
<openSSL>
    <server>
        <verificationMode>none</verificationMode>
        <loadDefaultCAFile>true</loadDefaultCAFile>
        <cacheSessions>true</cacheSessions>
        <disableProtocols>sslv2,sslv3</disableProtocols>
        <preferServerCiphers>true</preferServerCiphers>
        <certificateFile>/etc/clickhouse-server/certs/server.crt</certificateFile>
        <privateKeyFile>/etc/clickhouse-server/certs/server.key</privateKeyFile>
        <caConfig>/etc/clickhouse-server/certs/ca-certificate.crt</caConfig>
        <invalidCertificateHandler>
            <name>RejectCertificateHandler</name>
```

```
        </invalidCertificateHandler>
    </server>

    <client>
        <certificateFile>/etc/clickhouse-server/certs/server.crt</certificateFile>
        <privateKeyFile>/etc/clickhouse-server/certs/server.key</privateKeyFile>
        <caConfig>/etc/clickhouse-server/certs/ca-certificate.crt</caConfig>
        <loadDefaultCAFile>true</loadDefaultCAFile>
        <cacheSessions>true</cacheSessions>
        <disableProtocols>sslv2,sslv3</disableProtocols>
        <preferServerCiphers>true</preferServerCiphers>
        <invalidCertificateHandler>
            <name>RejectCertificateHandler</name>
        </invalidCertificateHandler>
    </client>
</openSSL>
```



Скопируйте сертификат доверенного центра сертификации (в примере `/etc/clickhouse-server/certs/ca-certificate.crt`) на хост, с которого будет производиться установка системы.

Путь к каталогу сертификатов на хосте, с которого производится установка системы, потребуется указать на этапе **ClickHouse setup** установки системы.

Шардирование и репликация узлов кластера

Шардирование позволяет разбивать данные БД на части и размещать их на разных узлах кластера (шардах). Репликация предназначена для управления копиями данных (репликами) в БД и их синхронизации. Эти механизмы обеспечивают отказоустойчивость работы кластера и повышают надежность хранения данных.

Шардирование и репликация узлов задаются в блоке `remote_servers`.

▼ *Пример настройки шардирования и репликации*

```
<remote_servers>
    <default>
        <!-- Параметры шарда 1. -->
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>IP-адрес или FQDN узла 1</host>
                <port>9000</port>
                <user>writer</user>
                <password>writer</password>
            </replica>
        </shard>
        <!-- Параметры шарда 2. -->
        <shard>
```

```
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>IP-адрес или FQDN узла 2</host>
                <port>9000</port>
                <user>writer</user>
                <password>writer</password>
            </replica>
        </shard>
        <!-- Здесь пропущены параметры промежуточных шардов. -->
        <!-- Параметры шарда N. -->
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>IP-адрес или FQDN узла N</host>
                <port>9000</port>
                <user>writer</user>
                <password>writer</password>
            </replica>
        </shard>
    </default>
</remote_servers>
<macros>
    <cluster>default</cluster>
    <replica>IP-адрес или FQDN текущего узла</replica>
    <shard>Имя текущего узла</shard>
    <layer>clickhouse</layer>
</macros>
```

Если система работает только на одном узле, то настройка выглядит следующим образом:

▼ *Пример настройки шардирования для системы на одном узле*

```
<remote_servers>
    <default>
        <shard>
            <replica>
                <host>localhost</host>
                <port>9000</port>
                <user>writer</user>
                <password>writer</password>
            </replica>
        </shard>
    </default>
</remote_servers>
<macros>
    <cluster>default</cluster>
    <shard>1</shard>
    <replica>Имя узла</replica>
```

```
</macros>
```

Параметры ClickHouse

В зависимости от планируемой нагрузки на систему и доступных вычислительных ресурсов вы можете настроить параметры сервера ClickHouse и таблиц MergeTree, перечисленные в разделе Справка о параметрах ClickHouse.

▼ *Пример настройки параметров ClickHouse*

```
<background_buffer_flush_schedule_pool_size>6</background_buffer_flush_schedule_pool
_size>
<background_common_pool_size>4</background_common_pool_size>
<background_distributed_schedule_pool_size>8</background_distributed_schedule_pool_s
ize>
<background_fetches_pool_size>4</background_fetches_pool_size>
<background_message_broker_schedule_pool_size>8</background_message_broker_schedule_
pool_size>
<background_move_pool_size>4</background_move_pool_size>
<background_pool_size>8</background_pool_size>
<background_merges_mutations_concurrency_ratio>4</background_merges_mutations_concur
rency_ratio>
<background_schedule_pool_size>64</background_schedule_pool_size>
<mark_cache_size>5368709120</mark_cache_size>
<max_concurrent_queries>0</max_concurrent_queries>
<max_server_memory_usage_to_ram_ratio>0.9</max_server_memory_usage_to_ram_ratio>
<max_server_memory_usage>0</max_server_memory_usage>
```

▼ *Пример настройки параметров таблиц MergeTree*

```
<merge_tree>

<max_bytes_to_merge_at_max_space_in_pool>2147483648</max_bytes_to_merge_at_max_space
_in_pool>
    <merge_max_block_size>2048</merge_max_block_size>

<number_of_free_entries_in_pool_to_lower_max_size_of_merge>4</number_of_free_entries
_in_pool_to_lower_max_size_of_merge>
    <max_suspicious_broken_parts>500</max_suspicious_broken_parts>
</merge_tree>
```

▼ *Пример настройки параметров доступности БД в теге* `profiles → default`

```
<profiles>
    <default>
        <input_format_allow_errors_num>0</input_format_allow_errors_num>
        <input_format_allow_errors_ratio>1</input_format_allow_errors_ratio>
        <max_execution_time>10</max_execution_time>
        <queue_max_wait_ms>1000</queue_max_wait_ms>
        <connect_timeout_with_failover_ms>750</connect_timeout_with_failover_ms>
```

```
        <connections_with_failover_max_tries>2</connections_with_failover_max_tries>
        <skip_unavailable_shards>1</skip_unavailable_shards>
        <insert_distributed_sync>0</insert_distributed_sync>
        <prefer_localhost_replica>1</prefer_localhost_replica>
        <allow_suspicious_ttl_expressions>1</allow_suspicious_ttl_expressions>
    </default>
    <readonly>
        <readonly>1</readonly>
    </readonly>
</profiles>
```

Директории хранения данных

Директории хранения данных ClickHouse задаются в блоке `storage_configuration`:

- В блоке `disks` задаются диски, которые используются томами для хранения данных.
- В блоке `volumes` задаются тома для хранения данных:
  - `hot_volume` — том для хранения часто используемых данных (горячее хранение);
  - `cold_volume` — том для хранения редко используемых данных (холодное хранение).

Если хранение событий планируется в корневой директории ClickHouse, то используйте следующие настройки:

▼ *Настройки хранения событий в корневой директории ClickHouse*

```
<storage_configuration>
    <disks>
        <!-- Диски, используемые для хранения данных. -->
        <hot_disk_1>
            <!-- Пользовательское название диска. -->
            <path>/var/lib/clickhouse/hot-1/</path>
            <!-- Директория хранения данных. -->
        </hot_disk_1>
        <cold_disk_1>
            <path>/var/lib/clickhouse/cold-1/</path>
        </cold_disk_1>
    </disks>
    <policies>
        <!-- Политики хранения, задаваемые при создании таблицы. -->
        <streams_storage_policy>
            <!-- Название политики. -->
            <volumes>
                <!-- Тома хранения. Приоритет записи определяется порядком указания
томов. -->
                <hot_volume>
                    <!-- Том горячего хранения данных. -->
                    <disk>hot_disk_1</disk>
                    <!-- Диск из раздела disks. -->
                </hot_volume>
                <cold_volume>
```

```
                    <!-- Том холодного хранения данных. -->
                    <disk>cold_disk_1</disk>
                </cold_volume>
            </volumes>
        </streams_storage_policy>
    </policies>
</storage_configuration>
```

Если планируется хранить события в отдельной директории, то:

1. Создайте необходимые директории под горячее и холодное хранение данных с помощью команды:

   ```
   mkdir <dir_path>
   ```

   Здесь:

   - `<dir_path>` — путь до директории горячего или холодного хранения.

2. Выдайте права на созданные директории группе пользователей clickhouse с помощью команды:

   ```
   chown -R clickhouse:clickhouse <dir_path>
   ```

   Здесь:

   - `<dir_path>` — путь до директории горячего или холодного хранения.

3. Укажите пути до созданных директорий в `path` блоков `hot_disk_1` и `cold_disk_1`.

Пример настройки отдельных директорий

Например, необходимо хранить события ClickHouse в директориях `/storage/hot` (горячее хранение) и `/storage/cold` (холодное хранение).

Чтобы настроить хранение в этих директориях:

1. Создайте директории с помощью команд:

   ```
   mkdir /storage/hot
   mkdir /storage/cold
   ```

2. Выдайте права на директории группе пользователей clickhouse с помощью команд:

   ```
   chown -R clickhouse:clickhouse /storage/hot
   chown -R clickhouse:clickhouse /storage/cold
   ```



Если в директории `/storage` планируется хранение только данных событий

ClickHouse, то, вместо выдачи прав на каждую поддиректорию по отдельности, вы можете выдать права сразу на всю директорию `/storage`:

```
chown -R clickhouse:clickhouse /storage
```

3. Укажите пути до директорий в `path` блоков `hot_disk_1` и `cold_disk_1` в файле `config.xml`.

▼ *Пример настройки директорий горячего и холодного хранения*

```
<storage_configuration>
    <disks>
        <!-- Диски, используемые для хранения данных. -->
        <hot_disk_1>
            <!-- Пользовательское название диска. -->
            <path>/storage/hot/</path>
            <!-- Директория хранения данных. -->
        </hot_disk_1>
        <cold_disk_1>
            <path>/storage/cold/</path>
        </cold_disk_1>
    </disks>
    <policies>
        <!-- Политики хранения, задаваемые при создании таблицы. -->
        <streams_storage_policy>
            <!-- Название политики. -->
            <volumes>
                <!-- Тома хранения. Приоритет записи определяется порядком
указания томов. -->
                <hot_volume>
                    <!-- Том горячего хранения данных. -->
                    <disk>hot_disk_1</disk>
                    <!-- Диск из раздела disks. -->
                </hot_volume>
                <cold_volume>
                    <!-- Том холодного хранения данных. -->
                    <disk>cold_disk_1</disk>
                </cold_volume>
            </volumes>
        </streams_storage_policy>
    </policies>
</storage_configuration>
```

Настройка ClickHouse Keeper

ClickHouse Keeper — это служба для управления шардированием и репликацией данных в кластере ClickHouse. Подробную информацию о службе можно найти в официальной документации ClickHouse.

Настройка ClickHouse Keeper выполняется в блоке `keeper_server`.

▼ *Пример настроек ClickHouse Keeper*

```
<keeper_server>
    <tcp_port>9181</tcp_port>
    <server_id>ID текущего узла</server_id>
    <log_storage_path>/var/lib/clickhouse/coordination/log</log_storage_path>

<snapshot_storage_path>/var/lib/clickhouse/coordination/snapshots</snapshot_storage_
path>
    <coordination_settings>
        <operation_timeout_ms>10000</operation_timeout_ms>
        <session_timeout_ms>30000</session_timeout_ms>
        <raft_logs_level>trace</raft_logs_level>
    </coordination_settings>

    <raft_configuration>
        <!-- Параметры узла 1. -->
        <server>
            <id>ID узла 1</id>
            <hostname>FQDN узла 1</hostname>
            <port>9234</port>
        </server>
        <!-- Параметры узла 2. -->
        <server>
            <id>ID узла 2</id>
            <hostname>FQDN узла 2</hostname>
            <port>9234</port>
        </server>
        <!-- Здесь пропущены параметры промежуточных узлов. -->
        <!-- Параметры узла N. -->
        <server>
            <id>ID узла N</id>
            <hostname>FQDN узла N</hostname>
            <port>9234</port>
        </server>
    </raft_configuration>
</keeper_server>
```

Запуск кластера ClickHouse



Действия выполняются на каждом узле кластера.

Чтобы запустить кластер ClickHouse, необходимо запустить каждый его узел по отдельности:

1. Запустите текущий узел кластера:

   ```
   systemctl start clickhouse-server
   ```

   Ошибки запуска можно посмотреть в файле `clickhouse-server.err.log` или `clickhouse-`

`server.log` в директории `/var/log/clickhouse-server/`.

2. Проверьте доступ к текущему узлу:

```
clickhouse-client -q "select version()"
```

Если запуск узла произошел успешно, в консоль выведется установленная версия ClickHouse.



Доступ к другим узлам кластера можно проверить удаленно с помощью команды:

```
clickhouse-client -h <имя узла> -q "select version()"
```

Если запуск узла произошел успешно, в консоль выведется установленная версия ClickHouse.

Создание пользователей ClickHouse

Для работы системы необходимо, чтобы в ClickHouse были настроены три пользователя с разным набором прав:

- **owner** — пользователь с расширенным набором прав. Используется для создания, обновления и удаления таблиц (хранилищ событий), а также получения служебной информации.
- **writer** — пользователь с правами на запись. Используется в коллекторах для записи событий в ClickHouse.
- **reader** — пользователь с правами только на чтение. Используется для выполнения поисковых запросов.

Для создания требуемых пользователей и кластерных вариантов служебных таблиц используйте скрипт ниже. Выполните скрипт на любом экземпляре кластера ClickHouse:

```
-- Создать пользователя owner и выдать ему необходимые права.
CREATE USER IF NOT EXISTS owner ON CLUSTER '{cluster}' IDENTIFIED WITH sha256_password
BY '<OwnerPassword>';
GRANT ON CLUSTER '{cluster}' ALL on default.* TO 'owner';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.asynchronous_metrics TO 'owner';
```

```
-- Создать пользователя writer и выдать ему необходимые права.
CREATE USER IF NOT EXISTS writer ON CLUSTER '{cluster}' IDENTIFIED WITH
sha256_password BY '<WriterPassword>';
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' INSERT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER UPDATE ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER DELETE ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.asynchronous_metrics TO 'writer';

-- Создать пользователя reader и выдать ему необходимые права.
CREATE USER IF NOT EXISTS reader ON CLUSTER '{cluster}' IDENTIFIED WITH
sha256_password BY '<ReaderPassword>';
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'reader';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.asynchronous_metrics TO 'reader';

-- Создать необходимые распределенные таблицы.
CREATE TABLE IF NOT EXISTS default.disks ON CLUSTER '{cluster}' AS system.disks ENGINE
= Distributed('{cluster}', system, disks);
CREATE TABLE IF NOT EXISTS default.storage_policies ON CLUSTER '{cluster}' AS
system.storage_policies ENGINE = Distributed('{cluster}', system, storage_policies);
CREATE TABLE IF NOT EXISTS default.parts ON CLUSTER '{cluster}' AS system.parts ENGINE
= Distributed('{cluster}', system, parts);
CREATE TABLE IF NOT EXISTS default.query_log ON CLUSTER '{cluster}' AS
system.query_log ENGINE = Distributed('{cluster}', system, query_log);
CREATE TABLE IF NOT EXISTS default.columns ON CLUSTER '{cluster}' AS system.columns
ENGINE = Distributed('{cluster}', system, columns);
CREATE TABLE IF NOT EXISTS default.parts_columns ON CLUSTER '{cluster}' AS
system.parts_columns ENGINE = Distributed('{cluster}', system, parts_columns);
CREATE TABLE IF NOT EXISTS default.tables ON CLUSTER '{cluster}' AS system.tables
ENGINE = Distributed('{cluster}', system, tables);
CREATE TABLE IF NOT EXISTS default.clusters ON CLUSTER '{cluster}' AS system.clusters
```

```
ENGINE = Distributed('{cluster}', system, clusters);
CREATE TABLE IF NOT EXISTS default.asynchronous_metrics ON CLUSTER '{cluster}' AS
system.asynchronous_metrics ENGINE = Distributed('{cluster}', system,
asynchronous_metrics);
```

Здесь <OwnerPassword>, <WriterPassword> и <ReaderPassword> — это пароли пользователей owner, writer и reader соответственно. Пароли должны отвечать следующим требованиям:

- Пароль может содержать только строчные латинские буквы (a–z), заглавные латинские буквы (A–Z) и цифры (0–9).
- Минимальная длина пароля: 8 знаков.

Эти пароли необходимо использовать на этапах настройки общесистемного экземпляра ClickHouse и ClickHouse для событий ИБ при установке системы.

Справка о параметрах ClickHouse

Параметры сервера

Подробную информацию о параметрах сервера можно найти в официальной документации ClickHouse.

Вы можете настроить следующие параметры сервера ClickHouse:

- background_buffer_flush_schedule_pool_size — количество потоков для выполнения фонового сброса данных.

  Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- background_common_pool_size — количество потоков для освобождения памяти.

  Рекомендуемое значение: **(0—1) * количество ядер процессора**.

- background_distributed_schedule_pool_size — количество потоков для выполнения фоновых задач.

  Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- background_fetches_pool_size — количество потоков для репликации данных.

  Рекомендуемое значение: **(0—1) * количество ядер процессора**.

- background_message_broker_schedule_pool_size — количество потоков для фонового вывода сообщений.

  Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- background_move_pool_size — количество потоков для перемещений данных между дисками.

  Рекомендуемое значение: **(0—1) * количество ядер процессора**.

- background_pool_size — количество потоков для слияний и мутаций таблиц.

Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- `background_merges_mutations_concurrency_ratio` — отношение количества слияний и мутаций, которые могут выполняться одновременно, к количеству потоков `background_pool_size`. Например, если параметр принимает значение `2`, а `background_pool_size` — `16`, то сервер ClickHouse может одновременно выполнять 32 слияния.
- `background_schedule_pool_size` — количество потоков для выполнения фоновых задач.

  Рекомендуемое значение: **(8—16) * количество ядер процессора**.

- `mark_cache_size` — приблизительный размер кэша меток в байтах.

  Рекомендуемое значение: **1—5 ГБ**.

- `max_concurrent_queries` — максимальное количество одновременно обрабатываемых запросов. Если принимает значение `0`, то ограничений нет.

  Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- `max_server_memory_usage_to_ram_ratio` — доля оперативной памяти, используемой сервером ClickHouse. Измеряется от `0` до `1`. Если принимает значение `0`, то сервер может использовать всю оперативную память. По умолчанию параметр не задан и воспринимается системой равным `0.9`.

  Рекомендуемое значение: **0.9**.

- `max_server_memory_usage` — максимальный объем оперативной памяти в байтах, используемой сервером ClickHouse. Если принимает значение `0`, то объем памяти определяется автоматически следующим образом:

  ```
  <memory_amount> * max_server_memory_usage_to_ram_ratio
  ```

  Здесь:

  - `<memory_amount>` — общий объем оперативной памяти.

  Рекомендуемое значение: **0** при `max_server_memory_usage_to_ram_ratio = 0.9`.

- Параметры для ограничения количества ошибочных строк при вставке:

  - `input_format_allow_errors_num` — максимально допустимое количество строк с ошибками при выполнении операции вставки. Если строка содержит ошибку, но значение `input_format_allow_errors_num` не превышено, ClickHouse пропускает эту строку и продолжает выполнение запроса. Значение по умолчанию — `0`.
  - `input_format_allow_errors_ratio` — максимально допустимая доля строк с ошибками при выполнении операции вставки. Если строка содержит ошибку, но значение `input_format_allow_errors_ratio` не превышено, ClickHouse пропускает эту строку и продолжает выполнение запроса. Измеряется от `0` до `1`. Значение по умолчанию — `1`.

  Ошибка выполнения запроса возникает только при превышении одновременно `input_format_allow_errors_num` и `input_format_allow_errors_ratio`.

- `max_execution_time` — предельное время выполнения запроса в секундах. Значение по умолчанию — `10`.
- `queue_max_wait_ms` — предельное время ожидания запроса в очереди в миллисекундах, если количество параллельных запросов превышает лимит. Значение по умолчанию — `1000`.
- `connect_timeout` — предельное время ожидания подключения к БД в секундах, если в кластере не используется репликация. Значение по умолчанию — `10`.
- `connect_timeout_with_failover_ms` — предельное время ожидания подключения к БД в миллисекундах, если в кластере используются шардирование и репликация. Значение по умолчанию — `750`.
- `connections_with_failover_max_tries` — предельное количество попыток подключения к каждой реплике. Параметр следует настраивать, только если в кластере используется репликация. Значение по умолчанию — `2`.
- `skip_unavailable_shards` и `insert_distributed_sync` — параметры, которые определяют, будут ли пропущены недоступные шарды при операциях чтения (`SELECT`) и вставки (`INSERT`):

Операция	`skip_unavailable_shards`	`insert_distributed_sync`	**Результат**
`SELECT`	`0`	`0` или `1`	Ошибка при недоступности хотя бы одного шарда.
	`1`	`0` или `1`	Возвращаются данные только с доступных шардов. Часть данных, находящаяся на недоступных шардах, возвращена не будет, ошибок запроса не возникнет.
`INSERT`	`0`	`0` или `1`	Ошибка при недоступности хотя бы одного шарда.
	`1`	`0`	Будет произведена вставка в доступные шарды. Данные для вставки в недоступные шарды будут буферизованы для последующей вставки при восстановлении доступности шардов.
	`1`	`1`	Ошибка при недоступности хотя бы одного шарда.

- `prefer_localhost_replica` — если параметр включен, при получении распределенного запроса сервер ClickHouse проверяет наличие и доступность реплики на том же узле кластера, где он установлен. Если такая реплика доступна, запрос отправляется к ней, иначе сервер обращается к репликам на других узлах кластера. Значение по умолчанию — `true`.
- `allow_suspicious_ttl_expressions` — если параметр включен, TTL-выражения, не зависящие от столбцов таблицы, разрешены.



`allow_suspicious_ttl_expressions` должен иметь значение `1`, поскольку TTL-

выражения используются для внутрисистемных операций.

Параметры таблиц MergeTree

Таблицы семейства MergeTree позволяют использовать механизм репликации данных. Подробную информацию о MergeTree-параметрах можно найти в официальной документации ClickHouse.

Вы можете настроить следующие параметры MergeTree-таблиц:

- `max_bytes_to_merge_at_max_space_in_pool` — максимальный общий размер таблиц, которые могут быть слиты, в байтах.

  Рекомендуемое значение: **1—4 ГБ**.

- `merge_max_block_size` — количество строк, которые считываются в память из слитых таблиц.

  Рекомендуемое значение: **1024—4096**.

- `number_of_free_entries_in_pool_to_lower_max_size_of_merge` — количество свободных записей в пуле или реплицированной очереди, при преуменьшении которого уменьшится максимальный размер слияния.

  Рекомендуемое значение: **1—4**.

- `max_partition_size_to_drop` — максимальный размер партиции, которая может быть удалена с помощью запроса `DROP PARTITION`, в байтах.

  Значение по умолчанию: `50000000000`.



Рекомендуется установить для параметра `max_partition_size_to_drop` значение **0**, чтобы снять ограничение. Если его не снять, оно может привести к ошибке при удалении хранилища событий или активного списка с большим количеством записей.

- `max_suspicious_broken_parts` — максимальное количество поврежденных партиций. При превышении этого значения отключается автоматическое удаление.
- `max_table_size_to_drop` — максимальный размер таблицы, которая может быть удалена с помощью запроса `DROP` или `TRUNCATE`, в байтах.

  Значение по умолчанию: `50000000000`.



Рекомендуется установить для параметра `max_table_size_to_drop` значение **0**, чтобы снять ограничение. Если его не снять, оно может привести к ошибке при удалении хранилища событий или активного списка с большим количеством записей.

Вынесенная установка PostgreSQL

В данном разделе описана вынесенная установка PostgreSQL на одном сервере и в распределенном кластере.



Установка СУБД PostgreSQL на одном сервере не обеспечивает ее отказоустойчивости. Если вы планируете развертывание системы в отказоустойчивой конфигурации, рекомендуется установить PostgreSQL в выделенном кластере.

Содержание раздела:

- Установка PostgreSQL на выделенном сервере
- Установка PostgreSQL в выделенном кластере

Установка PostgreSQL на выделенном сервере

В этом разделе описан процесс установки СУБД PostgreSQL на выделенном сервере.



Установка СУБД PostgreSQL на одном сервере не обеспечивает ее отказоустойчивости. Если вы планируете развертывание системы в отказоустойчивой конфигурации, рекомендуется установить PostgreSQL в выделенном кластере.

Процесс распределенной установки PostgreSQL описан в разделе Установка PostgreSQL в выделенном кластере.

Требования к PostgreSQL

Для работы системы требуется реляционная СУБД:

- рекомендуется PostgreSQL версии 15 или выше;
- допустимо использование совместимых аналогов: Postgres Pro или Jatoba J4.

Установка PostgreSQL

Загрузите и установите PostgreSQL согласно инструкции с официального сайта.

Настройка PostgreSQL на выделенном сервере

Настройка конфигурационного файла `postgresql.conf`

Для корректной работы системы в конфигурационный файл `postgresql.conf` следует добавить следующие параметры:

- сетевой интерфейс и порт, прослушиваемые сервером PostgreSQL;
- параметры обработки строк.



Расположение файла зависит от используемой операционной системы:

- В дистрибутивах Linux, основанных на Debian, файл `postgresql.conf` находится в директории `/etc/postgresql/<version>/main/`.
- В дистрибутивах Linux, основанных на Red Hat, файл `postgresql.conf` находится в директории `/var/lib/pgsql/<version>/data/`.

> Если вы используете иную операционную систему, расположение файла `postgresql.conf` может быть другим.

Чтобы добавить вышеуказанные параметры в `postgresql.conf`, выполните следующие действия:

1. Откройте файл `postgresql.conf` для редактирования.
2. Добавьте в файл следующие строки:

   ```
   listen_addresses = '<postgresql_ip>'
   port = <port>
   standard_conforming_strings = 'on'
   escape_string_warning = 'on'
   backslash_quote = 'safe_encoding'
   ```

   Здесь:

   - `<postgresql_ip>` — IP-адрес сервера PostgreSQL, который система будет использовать для подключения к базе данных.
   - `<port>` — порт, который будет прослушивать сервер PostgreSQL в ожидании подключений. По умолчанию — 5432.
3. Сохраните и закройте файл `postgresql.conf`.

Настройка сетевого доступа кластера к PostgreSQL

Разрешения сетевого доступа клиентов к серверу PostgreSQL задаются в конфигурационном файле `pg_hba.conf`. Этот файл расположен в той же директории, что и файл `postgresql.conf`.

Чтобы настроить разрешения сетевого доступа:

1. Откройте файл `pg_hba.conf` для редактирования.
2. Разрешите подключение worker-узлов кластера Kubernetes, добавив их адреса в виде строк:

   ```
   host all all worker-node-ip1/32 scram-sha-256
   host all all worker-node-ip2/32 scram-sha-256
   host all all worker-node-ipN/32 scram-sha-256
   ```

   где `worker-node-ipX` — IP-адреса worker-узлов кластера, в котором развернута система. В зависимости от настроек PostgreSQL может использоваться алгоритм хэширования `md5` вместо приведенного в примере `scram-sha-256`.
3. Сохраните и закройте файл `pg_hba.conf`.

Проверка локали, используемой PostgreSQL

Чтобы система работала корректно, базы данных PostgreSQL должны использовать локаль `en_US.UTF-8`. Чтобы проверить используемую локаль и при необходимости изменить ее, выполните следующие действия:

1. Выполните следующий запрос к PostgreSQL:

   ```
   SELECT datname, datcollate, datctype
   FROM pg_database;
   ```

   Пример фрагмента вывода запроса

   ```
   datname     | datcollate  |  datctype
   ------------+-------------+------------
    postgres   | en_US.UTF-8 | en_US.UTF-8
    template1  | en_US.UTF-8 | en_US.UTF-8
    template0  | en_US.UTF-8 | en_US.UTF-8
    trigger    | en_US.UTF-8 | en_US.UTF-8
    auth       | en_US.UTF-8 | en_US.UTF-8
   ```

2. Если для баз данных `template0` и/или `template1` локаль отличается от `en_US.UTF-8`, выполните следующие действия:

   a. Остановите сервис `postgresql` и очистите созданные им данные:

      ```
      systemctl disable --now postgresql
      rm -rf <postgresql_data_path>
      ```

      Здесь:

      - `<postgresql_data_path>` — путь к каталогу данных PostgreSQL, например, `/var/lib/postgresql/15/main`.

   b. Выполните повторную инициализацию PostgreSQL с помощью следующей команды:

      ```
      initdb -D <postgresql_data_path> --locale=en_US.UTF-8
      ```

      Здесь:

      - `<postgresql_data_path>` — путь к каталогу данных PostgreSQL, например, `/var/lib/postgresql/15/main`.

Применение параметров

Перезапустите PostgreSQL, выполнив следующую команду:

```
service postgresql restart
```

или

```
systemctl restart postgresql
```

Добавление пользователя PostgreSQL

При установке PostgreSQL создает суперпользователя `postgres`. По умолчанию пароль для его учетной записи не задан. Рекомендуется создавать для системы отдельного пользователя с надежным паролем. Пользователь должен обладать привилегией `CREATEDB`, которая позволяет создавать новые базы данных и становиться их владельцем.

Чтобы добавить пользователя:

1. Переключитесь на пользователя `postgres` в командной строке:

   ```
   su postgres
   ```

   или

   ```
   sudo -i -u postgres
   ```

2. Добавьте пользователя, выполнив следующую команду:

   ```
   createuser --createdb --pwprompt system-user
   ```

   или

   ```
   psql -c "CREATE USER \"system-user\" WITH CREATEDB PASSWORD 'yourpassword';"
   ```

   где `system-user` — пример имени нового пользователя (может быть другим), `yourpassword` — пример пароля.

3. Пользователь добавлен. Завершите сеанс пользователя `postgres` командой `exit`.



При эксплуатации системы рекомендуется периодически создавать резервные копии баз данных.

Установка расширения `uuid-ossp`

Для работы PostgreSQL с данными типа `UUID` требуется установить расширение `uuid-ossp`. Для этого выполните следующие действия:

1. Получите список доступных расширений, выполнив следующую команду:

   ```
   psql -c "SELECT * FROM pg_available_extensions;"
   ```

   ▼ *Пример вывода списка доступных расширений*

   ```
        name           | default_version | installed_version |
   comment
   ```

```
-------------------+----------------+------------------+
----------------------------------------------------------------
 plpgsql            | 1.0            | 1.0              | PL/pgSQL procedural
language
 pg_stat_statements | 1.9            |                  | track execution
statistics of all SQL statements executed
 hstore             | 1.8            |                  | data type for storing
sets of key/value pairs
 uuid-ossp          | 1.1            |                  | generate universally
unique identifiers (UUIDs)
 pg_trgm            | 1.6            |                  | text similarity
measurement and index searching based on trigrams
```

Если в столбце `name` присутствует значение `uuid-ossp` и в строке `uuid-ossp` в столбце `installed_version` есть значение, дополнительных действий не требуется. Иначе переходите к следующим шагам.

2. Если в столбце `name` значение `uuid-ossp` отсутствует, значит, расширение не входит в установленный вами дистрибутив PostgreSQL. Его можно скачать вместе с пакетом `postgresql-contrib`.

   Пример команды для скачивания `postgresql-contrib` для PostgreSQL 15 с помощью пакетного менеджера apt:

   ```
   sudo apt install postgresql-contrib-15
   ```

3. Установите расширение `uuid-ossp`, выполнив следующую команду:

   ```
   psql -c "CREATE EXTENSION IF NOT EXISTS \"uuid-ossp\";"
   ```

Установка PostgreSQL в выделенном кластере

В этом разделе описан процесс установки СУБД PostgreSQL в выделенном отказоустойчивом кластере.



Инструкция описывает процесс установки PostgreSQL 16 в выделенном кластере в ОС Debian 11 и РЕД ОС 7.3.

Для установки PostgreSQL согласно этой инструкции требуется:

- Для PostgreSQL — не менее двух выделенных узлов.
- Для хранилища конфигураций etcd — нечетное количество выделенных узлов, не менее трех. Рекомендуется использовать три узла.

В процессе развертывания кластера PostgreSQL будут установлены следующие вспомогательные компоненты:

- хранилище конфигураций etcd;

- фреймворк для управления кластером Patroni.

Развертывание кластера PostgreSQL состоит из следующих этапов:

- Установка etcd
- Установка PostgreSQL и Patroni
- Настройка Patroni

Установка etcd

Чтобы установить etcd, выполните следующие действия на каждом узле кластера etcd:

1. Установите etcd.

   Необходимо использовать etcd версии **ниже 3.6**. Patroni для передачи данных в etcd использует API V2, поддержка которого была прекращена в etcd 3.6.

   РЕД ОС

   ```
   dnf install etcd-3.5.15-2.el7
   ```

   Debian

   ```
   apt-get update
   apt-get install etcd=3.3.25+dfsg-6
   ```

2. В терминале создайте переменные окружения, значения которых будут использованы в конфигурационном файле etcd:

   ```
   ETCD1="<etcd_1_ip>"
   ETCD2="<etcd_2_ip>"
   ETCD<N>="<etcd_n_ip>"
   ETCD_TOKEN="<etcd_token>"
   ```

   Здесь:

   - `<etcd_1_ip>`, `<etcd_2_ip>`, `<etcd_n_ip>` — IP-адреса узлов etcd.

     Необходимо указать IP-адреса всех узлов etcd в формате `ETCD<I>="<etcd_i_ip>"`.

   - `<N>` — количество узлов в кластере etcd.
   - `<etcd_token>` — токен доступа к кластеру etcd. Токен представляет собой последовательность из произвольных латинских букв и цифр, например, `ArhXS2pIyurzwuFU`.

3. Создайте конфигурационный файл etcd, выполнив команду ниже.



Перед выполнением команды необходимо заменить шаблоны в угловых скобках (<>) на значения параметров, актуальные для вашего кластера.

```
echo -e """
ETCD_NAME="<etcd_node_name>"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_LISTEN_CLIENT_URLS="http://0.0.0.0:2379"
ETCD_ADVERTISE_CLIENT_URLS="http://${ETCD<I>}:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://${ETCD<I>}:2380"
ETCD_INITIAL_CLUSTER_TOKEN="${ETCD_TOKEN:?}"
# Если в конфигурации вашего кластера более трех выделенных узлов etcd, добавьте их
в эту строку.
ETCD_INITIAL_CLUSTER="etcd1=http://${ETCD1:?}:2380,etcd2=http://${ETCD2:?}:2380,etc
d3=http://${ETCD3:?}:2380"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_ELECTION_TIMEOUT="5000"
ETCD_HEARTBEAT_INTERVAL="1000"
ETCD_ENABLE_V2="true"
""" > cat $(systemctl show -P FragmentPath etcd.service --no-pager) | grep
EnvironmentFile | cut -d'-' -f2 | sed 's/%p/etcd/g'
```

Здесь:

- <etcd_node_name> — имя узла кластера etcd, например, etcd1.
- <I> — номер текущего узла кластера etcd.

4. Запустите сервис etcd:

```
systemctl enable --now etcd
```

5. Чтобы убедиться, что etcd успешно запущен, выведите таблицу статусов узлов кластера etcd:

```
etcdctl --endpoints="${ETCD1}:2379,${ETCD2}:2379,${ETCD3}:2379" endpoint status
--write-out=table
```



Если количество узлов etcd в вашем кластере отличается, необходимо соответственно изменить значение аргумента --endpoints. Например, для пяти узлов:

```
etcdctl
--endpoints="${ETCD1}:2379,${ETCD2}:2379,${ETCD3}:2379,${ETCD4}:2379
,${ETCD5}:2379" endpoint status --write-out=table
```

Установка PostgreSQL и Patroni

Установите пакеты PostgreSQL и Patroni на каждом узле кластера PostgreSQL с помощью команд:

РЕД ОС

```
dnf install -y postgresql16 postgresql16-server postgresql16-contrib patroni
python3-psycopg2 patroni-etcd
```

Debian

```
echo "deb [signed-by=/usr/share/postgresql-common/pgdg/apt.postgresql.org.asc]
https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list
mkdir -p /usr/share/postgresql-common/pgdg/
curl -o /usr/share/postgresql-common/pgdg/apt.postgresql.org.asc --fail
https://www.postgresql.org/media/keys/ACCC4CF8.asc
apt-get update
apt-get install -y postgresql-16 postgresql-contrib-16 postgresql-common
postgresql-client-16 patroni
```

Сервис `postgresql` может запуститься автоматически после установки. Если он запустился, его необходимо остановить. Для этого выполните следующие действия:

1. Проверьте статус сервиса `postgresql`, выполнив следующую команду:

   ```
   systemctl status postgresql
   ```

   Сервис не должен быть запущен.

2. Если сервис `postgresql` запущен, остановите его и очистите созданные им данные:

   ```
   systemctl disable --now postgresql
   rm -rf /var/lib/postgresql/16/main
   ```

Настройка Patroni

Создайте конфигурацию сервиса Patroni и запустите его. Для этого выполните следующие действия на каждом узле кластера PostgreSQL:

1. Создайте директорию для конфигурационного файла, выполнив следующие команды:

   ```
   CONFIG_PATH=$(grep -E 'ExecStart.*yml' $(systemctl show -P FragmentPath
   patroni.service --no-pager) | sed -n 's/.* \(\/.*\.yml\)/\1/p')
   mkdir -p $(dirname $CONFIG_PATH)
   ```

2. В терминале создайте переменные окружения, значения которых будут использованы в конфигурационном файле Patroni:

РЕД ОС

```
ETCD1_IP="<etcd_1_ip>"
ETCD2_IP="<etcd_2_ip>"
ETCD<M>_IP="<etcd_m_ip>"
POSTGRES1_IP="<postgres_1_ip>"
POSTGRES2_IP="<postgres_2_ip>"
POSTGRES<N>_IP="<postgres_n_ip>"
POSTGRES_PASSWORD="<postgres_password>"
REPLICATOR_PASSWORD="<replicator_password>"
ETCD_PORT="2379"
PATRONI_SCOPE_NAME="<patroni_scope_name>"
RESTAPI_PASSWORD="<rest_api_pass>"

DATA_DIR="/var/lib/pgsql/16/data"
BIN_DIR="/usr/pgsql-16/bin"
CONFIG_DIR="/var/lib/pgsql/16/data"
```

Debian

```
ETCD1_IP="<etcd_1_ip>"
ETCD2_IP="<etcd_2_ip>"
ETCD<M>_IP="<etcd_m_ip>"
POSTGRES1_IP="<postgres_1_ip>"
POSTGRES2_IP="<postgres_2_ip>"
POSTGRES<N>_IP="<postgres_n_ip>"
POSTGRES_PASSWORD="<postgres_password>"
REPLICATOR_PASSWORD="<replicator_password>"
ETCD_PORT="2379"
PATRONI_SCOPE_NAME="<patroni_scope_name>"
RESTAPI_PASSWORD="<rest_api_pass>"

DATA_DIR="/var/lib/postgresql/16/main"
BIN_DIR="/usr/lib/postgresql/16/bin"
CONFIG_DIR="/etc/postgresql/16/main"
```

Здесь:

- `<etcd_1_ip>`, `<etcd_2_ip>`, `<etcd_m_ip>` — IP-адреса соответствующих узлов etcd.
- `<M>` — количество узлов в кластере etcd.

> Необходимо указать IP-адреса всех узлов etcd в формате `ETCD<I>_IP="<etcd_i_ip>"`.

- `<postgres_1_ip>`, `<postgres_2_ip>`, `<postgres_n_ip>` — IP-адреса соответствующих узлов

PostgreSQL.



Необходимо указать IP-адреса всех узлов PostgreSQL в формате `POSTGRES<I>_IP="<postgres_i_ip>"`.

- `<N>` — количество узлов в кластере PostgreSQL.
- `<postgres_password>` — пароль пользователя postgres.



Этот пароль потребуется указать на этапе **PostgreSQL setup** установки системы.

- `<replicator_password>` — пароль пользователя replicator. Учетная запись replicator используется для репликации данных между узлами кластера PostgreSQL.
- `<patroni_scope_name>` — имя кластера PostgreSQL для Patroni, например, `evo`.
- `<rest_api_pass>` — пароль для взаимодействия между узлами Patroni по REST API.

3. Создайте конфигурационный файл Patroni, выполнив команду ниже.



Перед выполнением команды необходимо заменить шаблоны в угловых скобках (<>) на значения параметров, актуальные для вашего кластера.

```
echo -e """name: <postgres_node_name>
scope: ${PATRONI_SCOPE_NAME:?}

log:
  level: WARNING
  format: '%(asctime)s %(levelname)s: %(message)s'
  dateformat: ''
  max_queue_size: 1000
  dir: /var/log/postgresql
  file_num: 4
  file_size: 10000000
  loggers:
    postgres.postmaster: WARNING
    urllib3: DEBUG

etcd:
  hosts:
  - ${ETCD1_IP:?}:${ETCD_PORT:?}
  - ${ETCD2_IP:?}:${ETCD_PORT}
  # Здесь необходимо перечислить все узлы кластера etcd согласно шаблону:
  # - ${ETCD<I>_IP:?}:${ETCD_PORT}
  - ${ETCD<M>_IP:?}:${ETCD_PORT}

restapi:
  listen: 0.0.0.0:8008
  connect_address: ${POSTGRES<I>_IP}:8008
  authentication:
    username: patroni
```

```
    password: ${RESTAPI_PASSWORD:?}

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        archive_mode: 'off'
        archive_command: 'mkdir -p ../wal_archive && test ! -f ../wal_archive/%f &&
cp %p ../wal_archive/%f'
        autovacuum: 'on'
        autovacuum_max_workers: '6'
        autovacuum_vacuum_scale_factor: '0.1'
        autovacuum_analyze_scale_factor: '0.05'
        autovacuum_vacuum_threshold: '50'
        effective_io_concurrency: '100'
        hot_standby: 'on'
        listen_addresses: '*'
        maintenance_work_mem: '256MB'
        max_connections: 500
        max_replication_slots: '10'
        max_wal_senders: '10'
        port: '5432'
        random_page_cost: '1.1'
        seq_page_cost: '1'
        shared_buffers: '512MB'
        shared_preload_libraries: 'pg_stat_statements'
        timezone: 'UTC'
        wal_compression: 'off'
        wal_level: 'replica'
        wal_keep_segments: '8'
        wal_log_hints: 'on'
        work_mem: '16MB'
        standard_conforming_strings: 'on'
        escape_string_warning: 'on'
        backslash_quote: 'safe_encoding'

  initdb:
  - encoding: UTF8
  - locale: en_US.UTF-8
  - data-checksums
  pg_hba:
  - host replication replicator ${POSTGRES1_IP:?}/32 scram-sha-256
  - host replication replicator ${POSTGRES2_IP:?}/32 scram-sha-256
  # Здесь необходимо перечислить все узлы кластера PostgreSQL согласно шаблону:
  # - host replication replicator ${POSTGRES<I>_IP:?}/32 scram-sha-256
```

```
  - host replication replicator ${POSTGRES<N>_IP:?}/32 scram-sha-256
  - host all all 0.0.0.0/0 scram-sha-256
  users:
    postgres:
      password: ${POSTGRES_PASSWORD:?}
    replicator:
      password: ${REPLICATOR_PASSWORD:?}
      options:
        - replication

postgresql:
  listen: 0.0.0.0:5432
  connect_address: ${POSTGRES<I>_IP}:5432
  data_dir: ${DATA_DIR:?}
  bin_dir: ${BIN_DIR:?}
  config_dir: ${CONFIG_DIR:?}
  pgpass: /tmp/pgpass
  pg_hba:
    - local     all             all                             scram-sha-256
    - host      all             postgres        all             scram-sha-256
    - host      replication     replicator      ${POSTGRES1_IP}/32 scram-sha-256
    # Здесь необходимо перечислить все узлы кластера PostgreSQL согласно шаблону:
    # - host      replication     replicator      ${POSTGRES<I>_IP}/32 scram-sha-
256
    - host      replication     replicator      ${POSTGRES2_IP}/32 scram-sha-256
    - host      replication     replicator      127.0.0.1/32    scram-sha-256
    - host      all             all             0.0.0.0/0       scram-sha-256
  authentication:
    superuser:
      username: postgres
      password: ${POSTGRES_PASSWORD}
    replication:
      username: replicator
      password: ${REPLICATOR_PASSWORD}
  parameters:
    unix_socket_directories: '/var/run/postgresql'
    stats_temp_directory: /var/run/postgresql/16-main.pg_stat_tmp
    log_autovacuum_min_duration: 60s2
    log_checkpoints: 'on'
    log_connections: 'on'
    log_disconnections: 'on'
    log_line_prefix: '%m [%p]: [%l-1] db=%d,user=%u,app=%a,client=%h'
    log_lock_waits: 'on'
    log_min_duration_statement: '30s'
    log_min_error_statement: 'NOTICE'
    log_min_messages: 'WARNING'
    log_statement: 'ddl'
    log_temp_files: '0'
    log_timezone: 'UTC'
    log_truncate_on_rotation: 'on'
    log_destination: 'stderr'
```

```
    log_file_mode: '0600'
    log_error_verbosity: 'verbose'
    logging_collector: 'on'
    datestyle: 'iso, dmy'
    password_encryption: 'scram-sha-256'
    tags:
    nofailover: false
    noloadbalance: false
    clonefrom: false
    nosync: false
""" > ${CONFIG_PATH:?}
```

Здесь:

- `<postgres_node_name>` — имя узла кластера PostgreSQL, например, `postgres1`.
- `<M>` — количество узлов в кластере etcd.
- `<I>` — номер текущего узла кластера PostgreSQL.
- `<N>` — количество узлов в кластере PostgreSQL.

4. Чтобы убедиться, что файл конфигурации создан успешно, выведите его, выполнив следующую команду:

```
patronictl -c ${CONFIG_PATH} show-config
```

5. Создайте директории для хранения логов PostgreSQL, выполнив следующие команды:

```
mkdir -p /var/log/postgresql
chown postgres:postgres /var/log/postgresql
```

6. Запустите сервис Patroni:

```
systemctl enable --now patroni
```

7. Чтобы убедиться, что Patroni успешно запущен, выведите список узлов кластера PostgreSQL, активных на текущий момент:

```
patronictl -c ${CONFIG_PATH} list
```



После установки кластера PostgreSQL рекомендуется настроить балансировку запросов к его узлам. Методы балансировки запросов к СУБД описаны в разделе Настройка балансировки запросов к ClickHouse и PostgreSQL.

Настройка балансировки запросов к ClickHouse и PostgreSQL

При установке системы на этапах настройки ClickHouse и PostgreSQL можно указать по одному адресу узла из их кластеров. По умолчанию на указанные узлы будут отправляться все запросы от системы к СУБД, и поэтому данные узлы могут стать точками отказа.

Для обеспечения равномерной нагрузки на узлы кластеров СУБД рекомендуется настроить балансировку запросов. Можно использовать следующие методы балансировки:

- Независимая балансировка запросов к ClickHouse и PostgreSQL
- Предварительно настроенный балансировщик нагрузки
- Один сервер HAProxy
- Несколько серверов HAProxy и VRRP



Если вы установили PostgreSQL на одном сервере, действия, связанные с балансировкой запросов к PostgreSQL, выполнять не нужно.

Независимая балансировка запросов к ClickHouse и PostgreSQL

Данный метод обеспечивает балансировку запросов следующим образом:

- Для ClickHouse балансировка запросов осуществляется на уровне DNS-сервера по алгоритму Round-robin DNS: одно доменное имя (A-запись) сопоставляется с IP-адресами всех узлов кластера ClickHouse, и на каждый запрос к этому доменному имени возвращается IP-адрес нового узла.
- Для PostgreSQL балансировка осуществляется с помощью сервиса keepalived по протоколу VRRP.

Преимущества:

- Выделенные узлы балансировщика не требуются.
- Для Round-robin DNS — простая настройка и минимальные расходы вычислительных ресурсов на балансировку.

Недостатки:

- При использовании метода Round-robin DNS нагрузка и доступность узлов кластера ClickHouse не отслеживаются: если узел окажется перегружен или недоступен, часть запросов к ClickHouse по-прежнему будет направляться к нему.
- Сервис keepalived разворачивается на узлах кластера PostgreSQL и создает дополнительную нагрузку на них.

Балансировка запросов к ClickHouse

Алгоритм Round-robin DNS позволяет производить циклическую ротацию адресов, возвращаемых сервером в ответ на запросы DNS Lookup. В контексте R-Vision SIEM это означает, что каждый последующий запрос системы к ClickHouse будет обрабатываться разными узлами кластера ClickHouse. В случае недоступности узла, указанного во время установки системы, запросы будут обработаны другими узлами кластера.

В данном разделе приведена инструкция по настройке Round-robin DNS для DNS-сервера на базе Windows Server. Для выполнения настройки на другом DNS-сервере обратитесь к официальной документации его производителя.

Чтобы настроить Round-robin DNS, выполните следующие действия на DNS-сервере вашей организации:

1. Включите алгоритм Round-robin DNS:
   a. Откройте оснастку **Диспетчер DNS**, выполнив следующую команду:

      ```
      dnsmgmt.msc
      ```

   b. Откройте окно свойств DNS-сервера.
   c. Перейдите на вкладку **Дополнительно**.
   d. Установите флажок **Включить циклическое обслуживание**.
2. В оснастке **Диспетчер DNS** добавьте доменное имя кластера ClickHouse.

   Выбранное доменное имя кластера ClickHouse потребуется указать в качестве имени хоста на этапе **ClickHouse setup** установки системы.

3. Для вышеуказанного доменного имени добавьте A-записи для каждого узла кластера ClickHouse. Каждая из записей должна содержать IP-адрес определенного узла.

   Пример настройки A-записей для кластера ClickHouse с тремя узлами

Имя	Тип	Данные
clickhouse-cluster.example.local	Host (A)	192.168.0.1
clickhouse-cluster.example.local	Host (A)	192.168.0.2
clickhouse-cluster.example.local	Host (A)	192.168.0.3

4. Чтобы каждый запрос к ClickHouse гарантированно обращался к новому узлу кластера, а не использовал IP-адрес из кэша DNS, отключите кэширование результатов DNS-запросов для созданных A-записей:
   a. В оснастке **DNS-сервер** включите режим **Вид → Расширенный**.
   b. В свойствах каждой из созданных A-записей установите свойство **Срок жизни (TTL) записи** равным нулю.

Балансировка запросов к PostgreSQL

Действия, описанные в этом разделе, необходимо выполнять, только если СУБД PostgreSQL установлена в выделенном кластере.

Для работы протокола VRRP нужны определенные сетевые условия: все узлы кластера PostgreSQL должны находиться в одном L2-сегменте (multicast).

Чтобы настроить балансировку запросов к узлам кластера PostgreSQL, выполните следующие действия на каждом узле кластера:

1. Установите пакет keepalived:

   РЕД ОС

   ```
   dnf install keepalived
   ```

   Debian

   ```
   apt-get install -y keepalived
   ```

2. В терминале создайте переменные окружения, значения которых будут использованы в конфигурационном файле VRRP:

   ```
   VRRP_ADDRESS="<virtual_ip>"
   VRRP_NETMASK="<virtual_netmask>"
   VRRP_ROUTER_ID="<virtual_router_id>"
   ETH_INTERFACE="<vrrp_interface>"
   VRRP_AUTH_PASS="<auth_pass>"
   ```

   Здесь:

   - `<virtual_ip>` — выделенный IP-адрес, который будет привязан к сетевому интерфейсу на сервере, являющемся лидером VRRP, например, `10.99.57.50`.
   - `<virtual_netmask>` — маска подсети для VRRP-адреса, например, `24`.
   - `<virtual_router_id>` — индивидуальный идентификатор группы VRRP внутри L2-сегмента сети. Целое положительное число, например, `50`.
   - `<vrrp_interface>` — имя интерфейса, к которому будет привязан VRRP-адрес. По умолчанию — `ens192`.
   - `<auth_pass>` — пароль для аутентификации в VRRP-сети.

3. Создайте конфигурационный файл VRRP, выполнив команду ниже.

   Перед выполнением команды необходимо заменить шаблоны в угловых скобках (<>) на значения параметров, актуальные для вашего кластера.

   ```
   echo -e """global_defs {
     router_id <postgres_node_name>
   }
   vrrp_script track_pgmaster {
     script \"/usr/local/bin/chk_master.sh\"
       interval 5
       rise 3
   ```

```
    fall 3
    weight -25
  }
vrrp_instance ${VRRP_ADDRESS} {
  state BACKUP
  interface ${ETH_INTERFACE}
  virtual_router_id ${VRRP_ROUTER_ID:?}
  priority 100
  preempt
  advert_int 1
  authentication {
    auth_type PASS
    auth_pass ${VRRP_AUTH_PASS}
  }
  virtual_ipaddress {
    ${VRRP_ADDRESS:?}/${VRRP_NETMASK:?} dev ${ETH_INTERFACE:?}
  }
  track_script {
    track_pgmaster
  }
}
""" > /etc/keepalived/keepalived.conf
```

Здесь:

- `<postgres_node_name>` — имя узла кластера PostgreSQL, например, `postgres1`.

4. Создайте скрипт для отслеживания текущего лидера Patroni:

```
echo '''#!/bin/bash
[ $(curl -s -X OPTIONS -o /dev/null --max-time 1 -w '%{http_code}'
http://localhost:8008/master) -eq 200 ]
''' > /usr/local/bin/chk_master.sh
chmod +x /usr/local/bin/chk_master.sh
```

5. Проверьте конфигурацию сервиса keepalived, выполнив следующую команду:

```
keepalived -t
```

6. Запустите сервис keepalived:

```
systemctl enable --now keepalived
```

Предварительно настроенный балансировщик нагрузки

Если в вашей инфраструктуре есть выделенный балансировщик, который вы планируете использовать для распределения нагрузки в кластерах СУБД, настройте на нем балансировку

трафика по следующим правилам:

- Для ClickHouse — по портам для взаимодействия системы и ClickHouse (по умолчанию 9100, 8123 и 8443).
- Для PostgreSQL:
  - Трафик по порту 5432 должен направляться на порт 5432 master-узла Patroni.
  - Трафик по порту 5433 должен направляться на порт 5432 replica-узла Patroni.



Чтобы проверить, является ли узел кластера PostgreSQL master-узлом или replica-узлом в кластере Patroni, можно выполнить в его терминале следующие команды:

```
curl -s -o /dev/null -w "%{http_code}" localhost:8008/master
curl -s -o /dev/null -w "%{http_code}" localhost:8008/replica
```

Если первая команда вернет ответ `200`, то узел является master-узлом, иначе — replica-узлом.

Настройку следует производить согласно документации производителя балансировщика.

Преимущество: дополнительные узлы балансировщика не требуются.

Недостаток: настройку выделенного балансировщика необходимо производить самостоятельно.

Один сервер HAProxy

Балансировка с помощью выделенного сервера HAProxy обладает следующими свойствами:

Преимущества:

- Балансировка выполняется для кластеров ClickHouse и PostgreSQL одновременно.
- Реализован алгоритм проверки состояния узлов (healthcheck).
- При балансировке запросов учитывается нагрузка на узлы.

Недостатки:

- Узел, на котором расположен единственный балансировщик, может стать точкой отказа.
- Для балансировщика требуется выделенный сервер.

Чтобы настроить балансировку запросов с помощью выделенного сервера HAProxy, выполните следующие действия:

1. Перейдите в директорию с файлами распакованного архива `clickhouse-cluster.tar.gz`.
2. Откройте файл `inventory/hosts.ini` и дополните его содержимое.



В файле `inventory/hosts.ini` должны быть заполнены разделы `[clickhouse]`, `[clickhouse_keeper]`, `[clickhouse_shard<shard_id>]`.

Пример содержимого файла `inventory/hosts.ini` приведен в разделе документации Установка ClickHouse.

a. В раздел `[balancer]` добавьте FQDN узла, на котором будет расположен сервер балансировщика.

b. Если СУБД PostgreSQL установлена в выделенном кластере, в раздел `[patroni]` добавьте FQDN узлов кластера PostgreSQL.

Пример конфигурации балансировщика в файле `inventory/hosts.ini`

```
[balancer]
balancer.my.lan

[patroni]
postgres1.my.lan
postgres2.my.lan
```

3. Если при настройке кластера ClickHouse было включено шифрование трафика между системой и кластером (в файле `clickhouse.yml` указан параметр `dcc_clickhouse_ssl_enabled: true`), необходимо настроить TLS-сертификат для узла балансировщика.

   По умолчанию роль `generate_certificates`, встроенная в плейбук, производит генерацию самоподписанного сертификата. Генерируемый сертификат и ключ сохраняются в директории `<clickhouse_playbook_dir>/files/certs` на хосте, с которого производится запуск плейбука.

   Здесь и далее: `<clickhouse_playbook_dir>` — полный путь к директории, в которую распакован архив плейбука.

   Если у вас есть заранее подготовленный TLS-сертификат, который вы хотите использовать для балансировщика, добавьте в директорию `<clickhouse_playbook_dir>/files/certs` файлы сертификата и ключа, соблюдая следующие правила именования:

Имя файла	**Назначение**
`ca-certificate.crt`	Сертификат доверенного центра сертификации.
`ca-certificate.key`	Ключ доверенного центра сертификации.  Данный ключ используется для выпуска и подписания сертификатов, которые генерируются ролью `generate_certificates` при работе плейбука. При использовании своих сертификатов создайте пустой файл с таким именем.
`<balancer_fqdn>_server.crt`	Сертификат узла балансировщика, выпущенный доверенным центром сертификации.
`<balancer_fqdn>_server.key`	Приватный ключ для сертификата `<balancer_fqdn>_server.crt`.

Здесь:

- `<balancer_fqdn>` — FQDN узла балансировщика, указанное в файле `hosts.ini`.

При запуске плейбука с заранее подготовленными сертификатами генерация сертификатов производиться не будет. Произойдет копирование подготовленных сертификатов на узлы кластера.



Если установка системы будет производиться **не с того же хоста**, с которого выполняется развертывание ClickHouse, скопируйте сертификат доверенного центра сертификации `<clickhouse_playbook_dir>/files/certs/ca-certificate.crt` на этот хост.

Путь к сертификату центра сертификации на хосте, с которого производится установка системы, потребуется указать на этапе **ClickHouse setup** установки системы.

4. Если СУБД PostgreSQL установлена в выделенном кластере, в файл `playbooks/deploy_balancer.yml` добавьте конфигурацию для создания точек отслеживания состояния PostgreSQL:

```
- name: "Deploying Balancer"
  hosts: balancer
  any_errors_fatal: true
  tasks:
    # ...
    - name: "Include role haproxy"
      ansible.builtin.include_role:
        name: haproxy
      vars:
        haproxy_cfg_configuration:
          # ...
          postgresql_master_tcp:
            frontend: |
              bind *:5432
              mode tcp
            backend: |
              option httpchk OPTIONS /master
              http-check expect status 200
              default-server inter 3s fall 3 rise 2 on-marked-down shutdown-
sessions
              {% for host in groups['patroni'] -%}
              server patroni{{ groups['patroni'].index(host) + 1 }} {{ host }}:5432
maxconn 3000 check port 8008
              {% endfor %}
          postgresql_slave_tcp:
            frontend: |
              bind *:5433
              mode tcp
            backend: |
```

```
              option httpchk OPTIONS /replica
              http-check expect status 200
              default-server inter 3s fall 3 rise 2 on-marked-down shutdown-
sessions
              {% for host in groups['patroni'] -%}
              server patroni{{ groups['patroni'].index(host) + 1 }} {{ host }}:5432
maxconn 3000 check port 8008
              {% endfor %}
```

5. Запустите плейбук для развертывания балансировщика:

```
cd ..
ansible-playbook -i inventory/hosts.ini playbooks/deploy_balancer.yml
```

Несколько серверов HAProxy и VRRP

Балансировка с помощью нескольких серверов HAProxy и протокола VRRP обладает следующими свойствами:

Преимущества:

- Балансировка выполняется для кластеров ClickHouse и PostgreSQL одновременно.
- Реализован алгоритм проверки состояния узлов (healthcheck).
- При балансировке запросов учитывается нагрузка на узлы.
- Серверы балансировщика не являются точками отказа.

Недостаток: для балансировщика требуется несколько выделенных серверов.



Для работы протокола VRRP нужны определенные сетевые условия: все узлы HAProxy должны находиться в одном L2-сегменте (multicast).

1. Перейдите в директорию с файлами распакованного архива `clickhouse-cluster.tar.gz`.
2. Откройте файл `inventory/hosts.ini` и дополните его содержимое.



В файле `inventory/hosts.ini` должны быть заполнены разделы `[clickhouse]`, `[clickhouse_keeper]`, `[clickhouse_shard<shard_id>]`.

Пример содержимого файла `inventory/hosts.ini` приведен в разделе документации Установка ClickHouse.

   a. В раздел `[balancer]` добавьте FQDN узлов балансировщика.
   b. Если СУБД PostgreSQL установлена в выделенном кластере, в раздел `[patroni]` добавьте FQDN узлов кластера PostgreSQL.

Пример конфигурации двух узлов балансировщика в файле `inventory/hosts.ini`

```
[balancer]
```

```
balancer1.my.lan
balancer2.my.lan

[patroni]
postgres1.my.lan
postgres2.my.lan
```

3. Если при настройке кластера ClickHouse было включено шифрование трафика между системой и кластером (в файле `clickhouse.yml` указан параметр `dcc_clickhouse_ssl_enabled: true`), необходимо настроить TLS-сертификаты для узлов балансировщика.

   По умолчанию роль `generate_certificates`, встроенная в плейбук, производит генерацию самоподписанных сертификатов. Генерируемые сертификаты и ключи сохраняются в директории `<clickhouse_playbook_dir>/files/certs` на хосте, с которого производится запуск плейбука.

   Здесь и далее: `<clickhouse_playbook_dir>` — полный путь к директории, в которую распакован архив плейбука.

   Если у вас есть заранее подготовленные TLS-сертификаты, которые вы хотите использовать для балансировщика, добавьте в директорию `<clickhouse_playbook_dir>/files/certs` файлы сертификатов и ключей, соблюдая следующие правила именования:

Имя файла	**Назначение**
`ca-certificate.crt`	Сертификат доверенного центра сертификации.
`ca-certificate.key`	Ключ доверенного центра сертификации.  Данный ключ используется для выпуска и подписания сертификатов, которые генерируются ролью `generate_certificates` при работе плейбука. При использовании своих сертификатов создайте пустой файл с таким именем.
`<balancer_fqdn>_server.crt`	Сертификат узла балансировщика, выпущенный доверенным центром сертификации. Сертификаты должны быть сгенерированы для каждого узла балансировщика.
`<balancer_fqdn>_server.key`	Приватный ключ для сертификата `<balancer_fqdn>_server.crt`. Приватные ключи должны быть сгенерированы для каждого узла балансировщика.

   Здесь:

   - `<balancer_fqdn>` — FQDN узла балансировщика, указанное в файле `hosts.ini`.

При запуске плейбука с заранее подготовленными сертификатами генерация сертификатов производиться не будет. Произойдет копирование подготовленных сертификатов на узлы кластера.



Если установка системы будет производиться **не с того же хоста**, с которого выполняется развертывание ClickHouse, скопируйте сертификат доверенного центра сертификации `<clickhouse_playbook_dir>/files/certs/ca-certificate.crt` на этот хост.

Путь к сертификату центра сертификации на хосте, с которого производится установка системы, потребуется указать на этапе **ClickHouse setup** установки системы.

4. Если СУБД PostgreSQL установлена в выделенном кластере, в файл `playbooks/deploy_balancer.yml` добавьте конфигурацию для создания точек отслеживания состояния PostgreSQL:

```
- name: "Deploying Balancer"
  hosts: balancer
  any_errors_fatal: true
  tasks:
    # ...
    - name: "Include role haproxy"
      ansible.builtin.include_role:
        name: haproxy
      vars:
        haproxy_cfg_configuration:
          # ...
          postgresql_master_tcp:
            frontend: |
              bind *:5432
              mode tcp
            backend: |
              option httpchk OPTIONS /master
              http-check expect status 200
              default-server inter 3s fall 3 rise 2 on-marked-down shutdown-
sessions
              {% for host in groups['patroni'] -%}
              server patroni{{ groups['patroni'].index(host) + 1 }} {{ host }}:5432
maxconn 3000 check port 8008
              {% endfor %}
          postgresql_slave_tcp:
            frontend: |
              bind *:5433
              mode tcp
            backend: |
              option httpchk OPTIONS /replica
              http-check expect status 200
              default-server inter 3s fall 3 rise 2 on-marked-down shutdown-
sessions
```

```
            {% for host in groups['patroni'] -%}
            server patroni{{ groups['patroni'].index(host) + 1 }} {{ host }}:5432
maxconn 3000 check port 8008
            {% endfor %}
```

5. В файле `inventory/group_vars/balancer.yml` определите переменные для работы VRRP:

   ```
   vrrp_virtual_ipaddress: "<virtual_ip>"
   vrrp_virtual_router_id: "<virtual_router_id>"
   vrrp_interface: "<vrrp_interface>"
   ```

   Здесь:

   - `<virtual_ip>` — выделенный IP-адрес, который будет привязан к сетевому интерфейсу на сервере, являющемся лидером VRRP.
   - `<virtual_router_id>` — индивидуальный идентификатор группы VRRP внутри L2-сегмента сети. Целое положительное число.
   - `<vrrp_interface>` — имя интерфейса, к которому будет привязан VRRP-адрес. По умолчанию — `ens192`.

6. Запустите плейбук для развертывания балансировщика:

   ```
   cd ..
   ansible-playbook -i inventory/hosts.ini playbooks/deploy_balancer.yml
   ```

Подготовка к установке системы

В данном разделе описаны подготовительные действия, которые необходимо выполнить перед установкой системы.

Подготовка состоит из следующих этапов:

1. Настройка доступа к кластеру Kubernetes.
2. Настройка сетевого доступа к кластеру системы.
3. Настройка SSH-доступа к worker-узлам кластера.
4. Применение дополнительных настроек, если узлы кластера работают под управлением ОС Альт Сервер или Альт СП Сервер.
5. Скачивание архива с файлами для установки и проверка его целостности.

Настройка доступа к кластеру Kubernetes

До начала установки должен быть развернут кластер Kubernetes.

Установка может выполняться как с одного из master-узлов кластера Kubernetes, так и с машины администратора кластера. На машине, с которой будет выполняться установка, должны быть

соблюдены следующие требования:

1. Установлена утилита ansible.
2. В системных переменных окружения указан путь к файлу `kubeconfig`.

Linux

В ОС Linux путь к файлу `kubeconfig` по умолчанию — `$HOME/.kube/config`. Вы можете изменить его с помощью переменной окружения `KUBECONFIG`:

```
export KUBECONFIG=<path_to_confg>
```

Здесь:

- `<path_to_config>` — путь к файлу `kubeconfig`.

Windows

В ОС Windows путь к файлу `kubeconfig` по умолчанию — `$%USERPROFILE%\.kube\config`. Вы можете изменить его с помощью переменной окружения `KUBECONFIG`:

```
$env:KUBECONFIG = <path_to_confg>
```

Здесь:

- `<path_to_config>` — путь к файлу `kubeconfig`.

Проверить доступ к кластеру можно, выполнив следующую команду:

```
kubectl get nodes
```

Если доступ к кластеру есть, отобразится список узлов кластера, например:

```
NAME                    STATUS   ROLES           AGE    VERSION   INTERNAL-IP
EXTERNAL-IP   OS-IMAGE
master01.company.local   Ready    control-plane   140d   v1.26.3   10.10.10.1
<none>        RED OS MUROM (7.3.2)
worker01.company.local   Ready    <none>          140d   v1.26.3   10.10.10.2
<none>        RED OS MUROM (7.3.2)
worker02.company.local   Ready    <none>          140d   v1.26.3   10.10.10.3
<none>        RED OS MUROM (7.3.2)
worker03.company.local   Ready    <none>          140d   v1.26.3   10.10.10.4
<none>        RED OS MUROM (7.3.2)
```

Настройка доменного имени и сетевого адреса



Если система будет установлена в кластер **из одного узла**, пропустите данный этап. На этапе **Web access** установки системы укажите полное доменное имя (FQDN) виртуальной машины, на которой функционирует кластер.

Если система будет установлена в кластер **из нескольких узлов**, в качестве доменного имени **нельзя** использовать FQDN master-узла кластера. Используйте доменное имя, не занятое ни одним другим сервисом.

Создайте A-запись на DNS-сервере вашей организации. Она позволит ассоциировать доменное имя, по которому будет доступна система, с IP-адресом, зарезервированным для использования кластером системы. Запросы, поступающие на этот адрес, будут перенаправляться на сервисы кластера балансировщиком MetalLB и контроллером ingress-nginx, входящими в пакет установки системы.



Доменное имя должно удовлетворять следующим требованиям:

- Допустимые символы: `a–z`, `0–9`, `-`, `.`.
- Имя не должно начинаться и заканчиваться дефисом или точкой.

Пример допустимого доменного имени: `siem.company.local`.



- Диапазон IP-адресов, зарезервированных для системы, и ее доменное имя потребуется указать на этапе **Web access** установки системы.
- Инструкция по созданию A-записи приведена для DNS-сервера на базе Windows Server. Для выполнения настройки на другом DNS-сервере обратитесь к официальной документации его производителя.

Чтобы настроить сетевой доступ к кластеру системы:

1. Запросите у сетевых инженеров IP-адрес, который будет зарезервирован для использования кластером системы. Убедитесь, что этот адрес доступен для обращений из внешних сетей.
2. Добавьте исключение в диспетчере DHCP, чтобы адрес не мог быть динамически назначен другим хостам в сети:
   a. Откройте диспетчер DHCP. Например, это можно сделать, выполнив команду:

      ```
      dhcpmgmt.msc
      ```

   b. Выберите область адресов, соответствующих подсети, в которую входит зарезервированный адрес.
   c. Откройте контекстное меню пула адресов области и выберите опцию **Диапазон исключения**.
   d. В открывшемся окне укажите зарезервированный адрес и нажмите на кнопку **Добавить**.
3. Добавьте A-запись в диспетчере DNS:
   a. Откройте диспетчер DNS. Например, это можно сделать, выполнив команду:

```
dnsmgmt.msc
```

b. Выберите необходимую зону прямого просмотра.

c. Откройте контекстное меню зоны и выберите опцию **Создать узел (A или AAAA)**.

d. В открывшемся окне укажите доменное имя и IP-адрес, который будет с ним ассоциирован.

e. Чтобы обеспечить возможность преобразования IP-адреса в доменное имя, установите флажок **Создать соответствующую PTR-запись**.

f. Нажмите на кнопку **Добавить узел**.

4. Если требуется, добавьте CNAME-запись для обращения к специфическим службам кластера, например:

```
grafana.rvision.company.local
```

Чтобы добавить CNAME-запись:

a. Откройте контекстное меню той же зоны, в которой создана A-запись, и выберите опцию **Создать псевдоним (CNAME)**.

b. В открывшемся окне укажите псевдоним и доменное имя системы.

c. При необходимости измените время существования (TTL) псевдонима.

5. Убедитесь, что преобразование доменного имени в IP-адрес и обратно выполняется успешно. Для этого:

a. Откройте терминал на вашем сервере.

b. Выполните команды:

```
nslookup <system_domain_name>
nslookup <system_ip>
```

Здесь:

- `<system_domain_name>` — доменное имя системы, используемое в A-записи.
- `<system_ip>` — IP-адрес системы, используемый в A-записи.

Обе эти команды должны возвращать доменное имя и IP-адрес системы. Если преобразование доменного имени в IP-адрес или обратно не выполняется, проверьте корректность A-записи, повторно выполнив шаги инструкции.

Настройка SSH-доступа к worker-узлам кластера

Для загрузки образов контейнеров на worker-узлы установщик использует Ansible. Необходимо настроить SSH-доступ с машины установщика до worker-узлов под единой учетной записью. Поддерживается доступ по ключам или паролю.



Если на master-узле кластера установлена ОС Альт Сервер или Альт СП Сервер и планируется доступ по паролю, установите утилиту sshpass на master-узле:

```
sudo apt-get install sshpass
```

Для настройки SSH-доступа выполните следующие действия на каждом worker-узле кластера:

1. Пользователю, от имени которого будет установлен кластер, необходимо выдать права суперпользователя (root). Для этого добавьте следующую строку в файл `/etc/sudoers`:

   ```
   <user> ALL=(ALL:ALL) NOPASSWD: ALL
   ```

   Здесь:

   - `<user>` — пользователь, от имени которого будет установлен кластер.

2. Настройте SSH-доступ по паролю или по ключу:
   - Если планируется доступ по паролю:
     a. Установите пакет sshpass:

     РЕД ОС

     ```
     sudo dnf install sshpass
     ```

     Альт Сервер, Альт СП Сервер, Astra Linux

     ```
     sudo apt-get install sshpass
     ```

     b. Проверьте возможность подключения к узлу по SSH:

     ```
     ssh <user>@<fqdn>
     ```

     Здесь:

     - `<user>` — пользователь, от имени которого будет установлен кластер.
     - `<fqdn>` — полное доменное имя (FQDN) текущего узла.

     Чтобы узнать FQDN текущего узла, выполните команду:

     ```
     hostname -f
     ```

   - Если планируется доступ по ключу:

a. Проверьте возможность подключения к узлу по SSH:

```
ssh <user>@<fqdn>
```

Здесь:

- `<user>` — пользователь, от имени которого будет установлен кластер.
- `<fqdn>` — полное доменное имя (FQDN) текущего узла.



Чтобы узнать FQDN текущего узла, выполните команду:

```
hostname -f
```

b. Если подключение по SSH-ключу не настроено, сгенерируйте ключи и добавьте открытый ключ на узел:

```
ssh-keygen
ssh-copy-id <user>@<fqdn>
```

Здесь:

- `<user>` — пользователь, от имени которого будет установлен кластер.
- `<fqdn>` — полное доменное имя (FQDN) текущего узла.

Дополнительные настройки для Альт Сервер и Альт СП Сервер

Если узлы кластера работают под управлением ОС Альт Сервер или Альт СП Сервер, выполните следующие действия на master-узле:

1. Установите коллекцию ansible.posix:

```
ansible-galaxy collection install ansible.posix
```

2. Скачайте пакеты libnewt52 и newt52 из репозитория Альт:

```
wget
http://ftp.altlinux.org/pub/distributions/ALTLinux/Sisyphus/x86_64/RPMS.classic/lib
newt52-0.52.24-alt1.x86_64.rpm
wget
http://ftp.altlinux.org/pub/distributions/ALTLinux/Sisyphus/x86_64/RPMS.classic/new
t52-0.52.24-alt1.x86_64.rpm
```

3. Установите пакеты libnewt52 и newt52:

```
rpm -i libnewt52-0.52.24-alt1.x86_64.rpm
rpm -i newt52-0.52.24-alt1.x86_64.rpm
```

4. Расширьте системную переменную `PATH`, чтобы утилиты, расположенные в каталоге `<setup_dir>/r-vision/common/bin`, были доступны из любой директории без указания полного пути:

   ```
   export PATH="$PATH:<setup_dir>/r-vision/common/bin"
   ```

   Здесь:

   - `<setup_dir>` — каталог, в который будет установлена система. По умолчанию: `/opt`.

Получение дистрибутива

Скачайте архив с установочными файлами `evo.platform_<version>.run` и проверьте его контрольную сумму. Ссылка на дистрибутив предоставляется по запросу в службу поддержки по адресу support@rvision.ru.

```
# Получение дистрибутива.
curl -O https://files.rvision.ru/platform/<random_part>/evo.platform_<version>.run

# Получение контрольной суммы.
curl -O
https://files.rvision.ru/platform/<random_part>/evo.platform_<version>.run.sha256

# Проверка контрольной суммы.
sha256sum -c evo.platform_<version>.run.sha256

# При успешной проверке контрольной суммы будет выведено следующее сообщение.
evo.platform_<version>.run: OK

# При нарушении контрольной суммы будет выведено следующее сообщение.
evo.platform_<version>.run: FAILED
```



При выявлении нарушения контрольной суммы выполнять установку **недопустимо**.

Необходимо убедиться, что скачивание run-файла дистрибутива было полностью завершено, и повторно проверить контрольную сумму.

При подтверждении нарушения контрольной суммы следует сообщить в техническую поддержку по адресу support@rvision.ru.

Установка системы

Система предназначена для работы в среде Kubernetes и поставляется в виде контейнерных образов

и helm-чартов. В данном разделе приведены инструкции по офлайн-установке системы.

Содержание раздела:

- Пакет установки
- Процесс установки системы

Пакет установки

В данном разделе приведено описание дистрибутива и установщика системы.

Описание дистрибутива

Дистрибутив системы рассчитан на автономную установку в существующий кластер Kubernetes без необходимости доступа к сети Интернет. Дистрибутив представляет собой самораспаковывающийся run-файл (bash-скрипт) с полезной нагрузкой.

Дистрибутив состоит из компонентов установщика и модулей платформы.

Компоненты установщика

- Скрипт установки (bash).
- Скрипт управления экземпляром платформы `evoctl`.
- Плейбуки Ansible.
- Вспомогательные утилиты (исполняемые файлы: `helm`, `jq`, `yq`).

Модули платформы

- Модуль evo.infra, содержащий следующие компоненты:
  - Инфраструктурные сервисы:
    - мониторинг: kube-prometheus-stack;
    - контроллер: ingress-nginx;
    - провайдер хранилища: local-path-provisioner;
    - логирование: loki и promtail;
    - балансировщик нагрузки: metallb;
    - сервер метрик: metrics-server.
  - Сторонние сервисы:
    - ClickHouse;
    - NATS;
    - Consul;
    - Dkron;
    - MinIO;
    - PostgreSQL.

- Архивы модулей платформы.

Описание установщика

Установщик системы — это самораспаковывающийся run-файл (bash-скрипт) с полезной нагрузкой.

Формат имени файла: `evo.platform_<version>.run`

Для вывода встроенной справки запустите run-файл `evo.platform_<version>.run` с параметром `-h` или `--help`:

```
./evo.platform_<version>.run --help
```



Если файл не имеет прав на выполнение, установите их с помощью команды `chmod`:

```
sudo chmod +x evo.platform_<version>.run
```

▼ *Результат*

```
Usage: evo.platform_<version>.run [OPTIONS]

Self-extracting installer for R-Vision EVO Platform, version: <version>.

Available main actions:
  - new installation;
  - upgrading existing installation;
  - removing product;
  - downloading additional packages;
  - unpacking packages.

Options:
  -c, --check         only check if system is ready for installation, don't execute
actions
  -h, --help          show usage
  -V, --version       show version
  -v, --verbose       verbose output
  -d, --debug         debug mode
  -y, --yes           don't ask user confirmation
  --skip-integrity    skip integrity check before unpacking

Supported variables:
  - ACCEPTED          don't ask user confirmation
  - ACTION_MODE       action from list: install, update, remove, unpack, download,
check
  - VERBOSE           verbose output
  - DEBUG             debug mode
  - SETUP_ROOT_DIR    setup root directory, default: /opt
  - SKIP_INTEGRITY    skip integrity check before unpacking
```

```
Required resources (for installation process only):
  CPU, cores: 1
  RAM, Gb:    2
  DISK, Gb:   20 (in /opt/r-vision)
  INODES:     5000 (in /opt/r-vision)
```

Структура каталогов

Установщик содержит как общие разделяемые компоненты, так и специфичные для продукта.

По умолчанию распаковка выполняется в каталог `/opt`, например `/opt/r-vision`. Корневой каталог (`/opt` по умолчанию) можно изменить с помощью переменной `SETUP_ROOT_DIR`.

Таким образом:

- `/opt` — корневой каталог установщика;
- `/opt/r-vision` — каталог производителя.

Если пользователь не имеет права записи в каталог `/opt`, необходимо предварительно создать каталог производителя и назначить его владельцем пользователя, от имени которого будет выполняться установка.

Пример создания каталога установки

```
sudo mkdir -pv /opt/r-vision
sudo chown -R <user>:<group> /opt/r-vision
```

Здесь:

- `<user>` — имя пользователя;
- `<group>` — группа, в которую входит пользователь.

▼ *Пример возможной структуры каталогов после распаковки*

```
/opt/r-vision/               # Каталог производителя.
 ├─── common                    # Общие компоненты установщика.
 │    ├─── ansible               # Роли и плейбуки Ansible.
 │    ├─── bin                   # Утилиты.
 │    │    ├─── evoctl
 │    │    ├─── helm
 │    │    ├─── jq
 │    │    └─── yq
 │    ├─── lib                   # Библиотека bash-функций.
 │    ├─── logs                  # Логи установщика.
 │    └─── platform.yaml         # Манифест дистрибутива.
 ├─── core                      # Каталог модуля core.
 │    ├─── helm
 │    ├─── images
 │    └─── module.yaml
 ├─── infra                     # Каталог модуля infra.
```

```
│     ├──── helm
│     ├──── images
│     ├──── menu.sh
│     ├──── module.yaml
│     ├──── postinstall.sh
│     └──── preinstall.sh
├──── streams                  # Каталог модуля streams.
│     ├──── helm
│     ├──── images
│     ├──── menu.sh
│     ├──── module.yaml
│     ├──── postinstall.sh
│     └──── preinstall.sh
└──── space                    # Каталог модуля space.
      ├──── helm
      ├──── images
      ├──── module.yaml
      ├──── postinstall-space.sh
      └──── preinstall-space.sh
```

Логирование

Все действия логируются. При каждом запуске установщика создается новый лог-файл в каталоге `logs` системы. В лог сохраняется более подробная информация, чем выводится на экран, даже в verbose-режиме. Также в лог сохраняется информация об операционной системе, ресурсах, программном окружении и переменных. Значения переменных с именами, содержащими `pass` или `private`, маскируются.

▼ *Пример системной информации в логе*

```
System information:
  OS ID:          redos
  OS name:        RED OS MUROM (7.3.4)
  Kernel:         6.1.52-1.el7.3.x86_64
  User:           user
  CPU, cores:     8
  RAM, GB:        16
App directory info:
  Directory:      /opt/r-vision
  Free space:     53G
  Free inodes:    50903398


Suitable tools:
  bash|4.0.0:           5.2.9
  gunzip:               1.9.0
  sed:                  4.5.0
  tar:                  1.30.0
  diff:                 3.6.0
  ansible|2.9:          2.12.5
  kubectl:              1.26.3
```

```
  python3|3.6.8:       3.8.18
  curl:                7.85.0
  whiptail:            0.52.21
  dialog:              <not installed>

Environment:
------------
HOME=/home/user
LANG=en_US.UTF-8
LANGUAGE=en_US.UTF-8
TERM=xterm-256color
TEST_PASSWORD=<masked>
private_key=<masked>
USER=user
------------
```

Интерфейс установщика

Для взаимодействия с пользователем используется консольная диалоговая утилита `whiptail`. Если она не установлена, используется режим консоли.

У пользователя интерактивно запрашиваются необходимые параметры установки, выполняется проверка введенных значений.

Также поддерживается неинтерактивный режим установки с получением параметров из переменных окружения, определенных в env-файле.

▼ *Встроенная справка установщика*

```
Usage: evo.platform_<version>.run [OPTIONS]

Self-extracting installer for R-Vision EVO Platform, version: 1.0.0.

Available main actions:
  - new installation;
  - upgrading existing installation;
  - removing product;
  - downloading additional packages;
  - unpacking packages.

Options:
  -c, --check         only check if system is ready for installation, don't execute
actions
  -h, --help          show usage
  -V, --version       show version
  -v, --verbose       verbose output
  -d, --debug         debug mode
  -y, --yes           don't ask user confirmation
  --skip-integrity    skip integrity check before unpacking

Supported variables:
```

```
  - ACCEPTED          don't ask user confirmation
  - ACTION_MODE       action from list: install, update, remove, unpack, download,
check
  - VERBOSE           verbose output
  - DEBUG             debug mode
  - SETUP_ROOT_DIR    setup root directory, default: /opt
  - SKIP_INTEGRITY    skip integrity check before unpacking

Required resources (for installation process only):
  CPU, cores: 1
  RAM, Gb:    2
  DISK, Gb:   20 (in /opt/r-vision)
  INODES:     5000 (in /opt/r-vision)
```

Порядок работы установщика

Процесс работы установщика состоит из следующих этапов:

1. Подготовка (общий этап для всех продуктов R-Vision, работающих на платформе Kubernetes):
   - Проверка соответствия программным и аппаратным требованиям для работы установщика.
   - Проверка целостности файла установщика.
   - Обнаружение и проверка целостности загружаемых пакетов.
   - Выбор режима работы установщика:
     - Новая установка в существующий кластер Kubernetes.
     - Обновление предыдущей версии.
     - Удаление установленного экземпляра системы.
     - Распаковка имеющихся архивов.
     - Скачивание недостающих пакетов.
2. Действия, специфичные для выбранного режима работы установщика.

При выборе установки, обновления или удаления выполняется распаковка пакетов в каталог продукта или их удаление из каталога, а управление передается скрипту установщика системы.

Процесс установки системы

В данном разделе описаны действия по выполнению новой установки системы.

Установка состоит из следующих этапов:

1. Аудит системного окружения
2. Загрузка пакетов экспертизы
3. Запуск установщика
4. Начало установки
5. Выбор пакетов экспертизы для импорта

6. Настройка Ansible
7. Выбор пространства имен Kubernetes
8. Настройка доменного имени
9. Настройка пула адресов балансировщика
10. Настройка PostgreSQL
11. Настройка ClickHouse
12. Настройка Prometheus
13. Настройка Loki
14. Настройка MinIO
15. Настройка Dkron
16. Настройка NATS
17. Настройка СУБД для работы с событиями ИБ
18. Завершение установки
19. Удаление файлов установщика
20. Авторизация в системе

Аудит системного окружения

Отключение Kaspersky Endpoint Security for Linux (KESL)

Если на узлах кластера установлен Kaspersky Endpoint Security for Linux (KESL), рекомендуется отключить его на время установки, поскольку работа KESL может препятствовать корректному развертыванию сети Kubernetes.

Чтобы временно остановить задачи KESL, выполните следующие действия на каждом узле кластера, где он установлен:

1. Выполните следующие команды от имени суперпользователя:

```
kesl-control --stop-task Network_Threat_Protection
kesl-control --stop-task Web_Threat_Protection
```



Если используется другая конфигурация или дополнительные задачи KESL, возможно, потребуется их также временно остановить.

2. Убедитесь, что задачи действительно остановлены, а службы KESL отключены и не влияют на установку:

```
kesl-control --list-tasks
```

3. После завершения установки системы восстановите защиту, выполнив следующие команды:

```
kesl-control --start-task Network_Threat_Protection
kesl-control --start-task Web_Threat_Protection
```

Отключение IPv6

Чтобы избежать ошибок разрешения имен, убедитесь, что при развертывании кластера на его узлах было отключено использование IPv6. Инструкция по отключению приведена в разделе по установке кластера Kubernetes.

Отключение брандмауэра firewalld

Если на узлах кластера установлен брандмауэр firewalld, на время установки системы рекомендуется отключить его, поскольку он может препятствовать сетевому взаимодействию между подами.

Чтобы отключить firewalld:

1. Проверьте статус firewalld:

   ```
   systemctl is-active firewalld.service
   ```

2. Если команда выше вывела `active`, остановите firewalld:

   ```
   systemctl stop firewalld.service
   ```

3. Проверьте, что firewalld остановлен:

   ```
   systemctl is-active firewalld.service
   ```

   Команда должна вывести `inactive`.

4. Установите систему.
5. По завершении установки включите firewalld:

   ```
   systemctl start firewalld
   ```

Загрузка пакетов экспертизы

Если у вас имеются пакеты элементов экспертизы с расширением .roc, загрузите их в каталог, где находится установщик. На этапе Выбор пакетов экспертизы для импорта вы сможете указать, чтобы они были автоматически импортированы в систему по завершении установки.

Запуск установщика

Чтобы запустить установщик системы:

1. После скачивания run-файла перейдите в директорию его хранения и запустите его с помощью bash:

```
bash ./evo.platform_<version>.run
```



Требуемая версия bash: 4.0 или выше.

Или сделайте его исполняемым и запустите непосредственно:

```
chmod +x ./evo.platform_<version>.run
./evo.platform_<version>.run
```



По умолчанию установка выполняется в каталог `/opt`. При необходимости вы можете изменить каталог установки с помощью переменной `SETUP_ROOT_DIR`:

```
SETUP_ROOT_DIR=<setup_dir> bash ./evo.platform_<version>.run
```

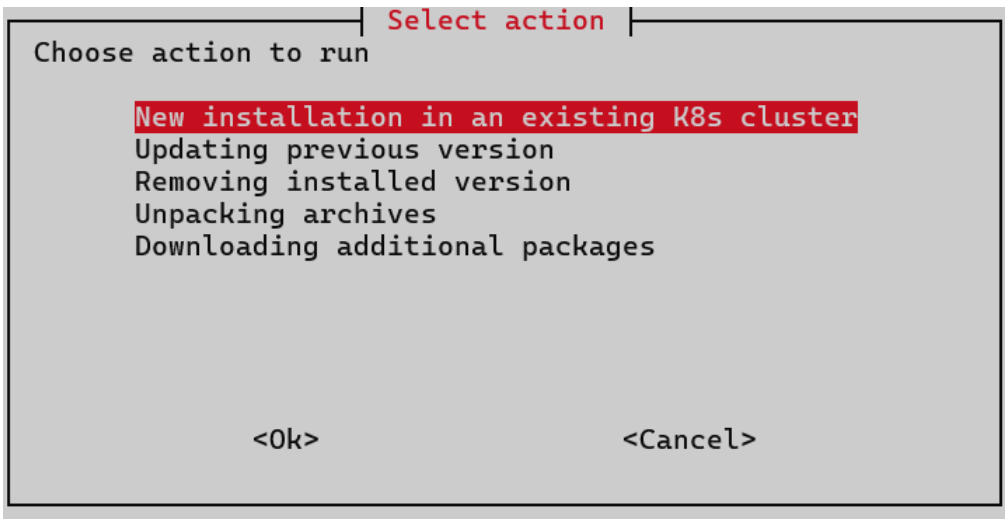
Здесь:

- `<setup_dir>` — директория установки.

Установщик выведет следующие сведения:

- версия системы и дата ее выпуска;
- путь к файлу хранения логов установки;
- результаты проверок:
  - удовлетворения системным требованиям;
  - наличия необходимых утилит;
  - целостности дистрибутива.

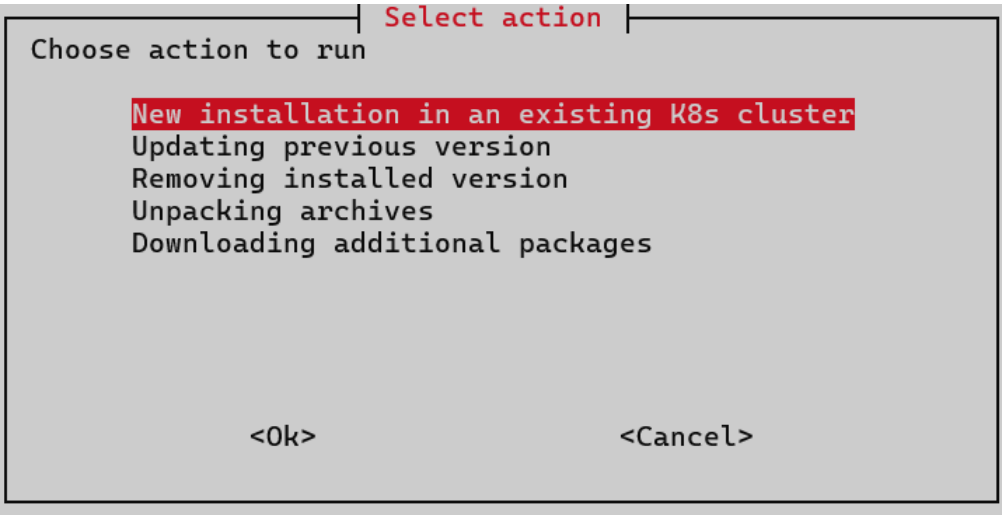
2. Выберите режим работы установщика и следуйте инструкциям на экране.



- **New installation in an existing K8s cluster** — новая установка в существующий кластер K8s.
- **Updating previous version** — обновление предыдущей версии.
- **Removing installed version** — удаление установленного экземпляра системы.
- **Unpacking archives** — распаковка имеющихся архивов без их установки.
- **Downloading additional packages** — скачивание дополнительных пакетов:
  - `kubespray-installer-redos.tar.gz` — автономный установщик Kubernetes для РЕД ОС;
  - `kubespray-installer-debian.tar.gz` — автономный установщик Kubernetes для Debian;
  - `kubespray-installer-rocky.tar.gz` — автономный установщик Kubernetes для Rocky Linux;
  - `k0s-installer.run` — автономный установщик K0s для развертывания кластера сателлита;
  - `clickhouse-cluster.tar.gz` — архив с Ansible-плейбуком для офлайн-развертывания кластера ClickHouse;
  - `sys-prep-clickhouse-redos.tar.gz` — архив пакетов для развертывания кластера ClickHouse в окружении РЕД ОС;
  - `sys-prep-clickhouse-debian.tar.gz` — архив пакетов для развертывания кластера ClickHouse в окружении Debian;
  - `sys-prep-clickhouse-rocky.tar.gz` — архив пакетов для развертывания кластера ClickHouse в окружении Rocky Linux.

Начало установки

Чтобы начать новую установку системы, запустите установщик и выберите режим работы **New installation in an existing K8s cluster**.



При выборе установки выполняется распаковка пакетов в каталог продукта и управление передается скрипту установщика.

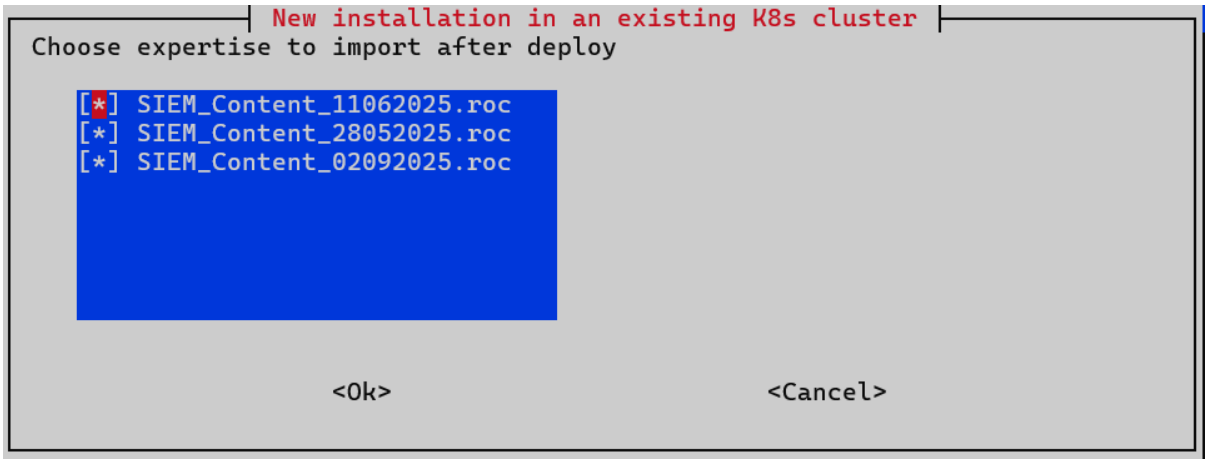


Если установка была прервана пользователем или из-за ошибки, можно перезапустить установщик без повторной распаковки дистрибутива, выполнив следующую команду:

```
$SETUP_ROOT_DIR/r-vision/common/bin/evoctl instance install
```

Выбор пакетов экспертизы для импорта

Если вы добавили в каталог установщика пакеты экспертизы с расширением .roc, установщик предложит выбрать пакеты, которые будут импортированы в систему по завершении установки.

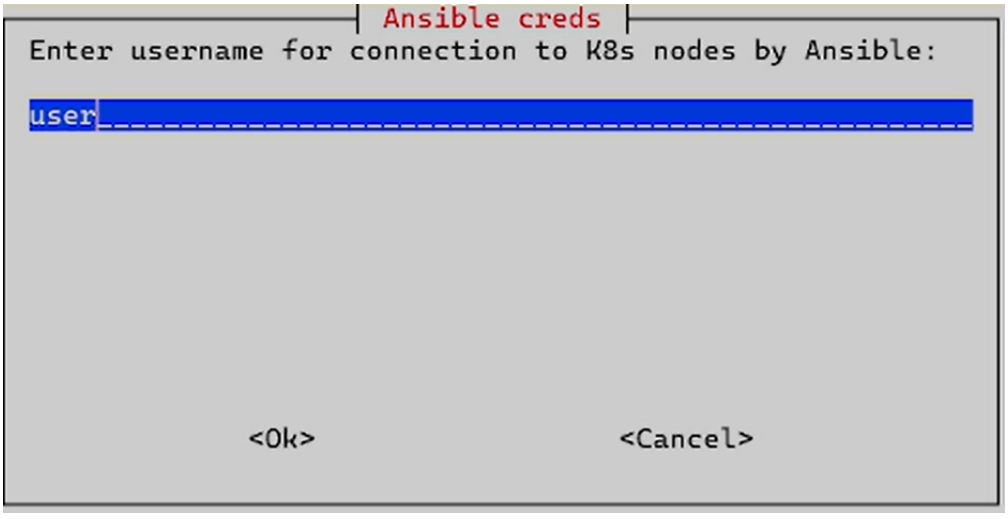


Перемещение по списку опций осуществляется с помощью клавиш со стрелками вверх и вниз. Чтобы выбрать опцию или отменить ее выбор, нажмите на клавишу SPACE.

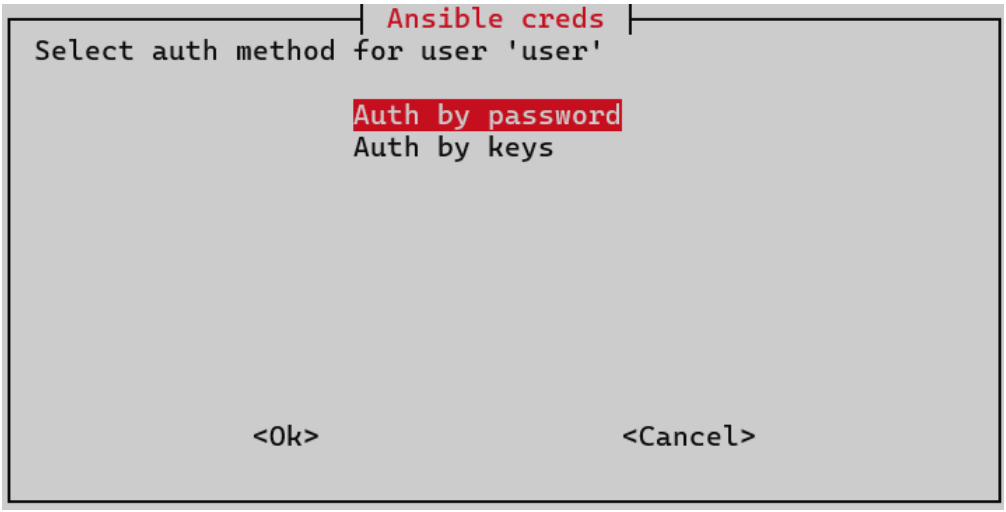
Настройка Ansible

Задайте параметры Ansible на этапе **Ansible creds**:

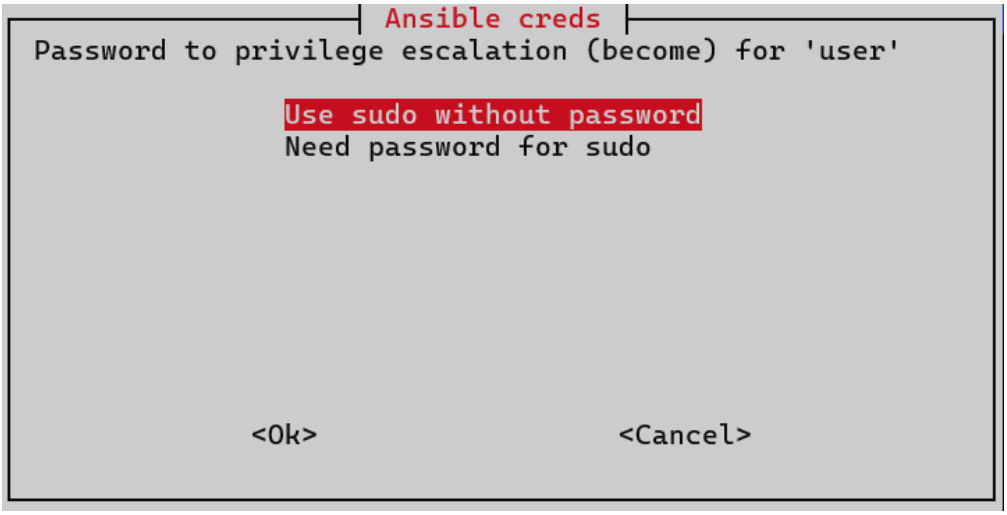
1. Задайте имя пользователя.



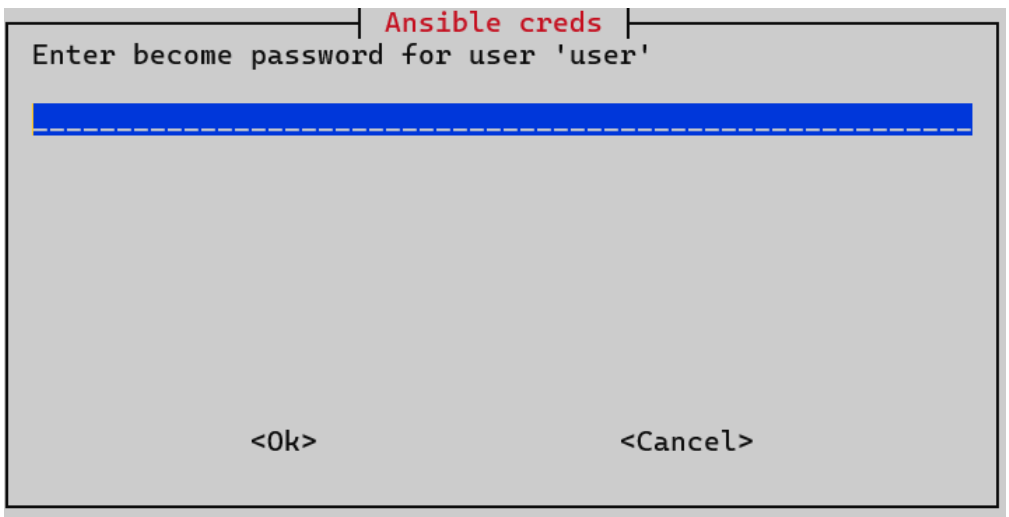
2. Выберите метод аутентификации: по паролю или через ключ.



3. Если выбрана аутентификация по паролю, задайте и подтвердите пароль пользователя.
4. Если выбрана аутентификация через ключ, выберите вариант повышения привилегий указанного пользователя через sudo:

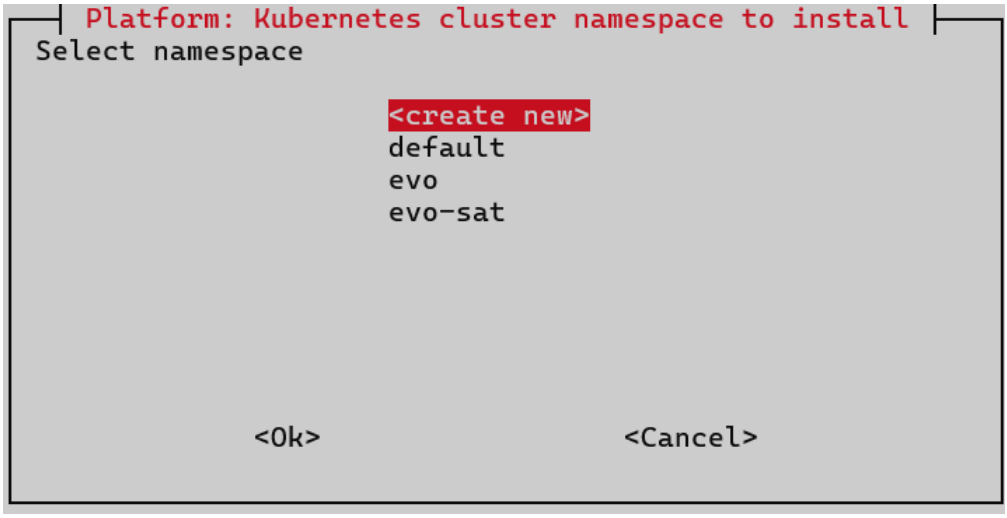


a. **Use sudo without password** — выберите этот вариант, если для пользователя настроено повышение привилегий через sudo без пароля.
b. **Need password for sudo** — выберите этот вариант, если для повышения привилегий пользователя через sudo необходимо ввести пароль. В таком случае введите этот пароль на следующем шаге:



Выбор пространства имен Kubernetes

Выберите пространство имен (namespace) Kubernetes, в которое будет установлена система, на этапе **Kubernetes cluster namespace to install**.



Если требуется создать новое пространство имен, выберите опцию **create new** и введите название пространства.

Название пространства имен должно удовлетворять следующим требованиям:

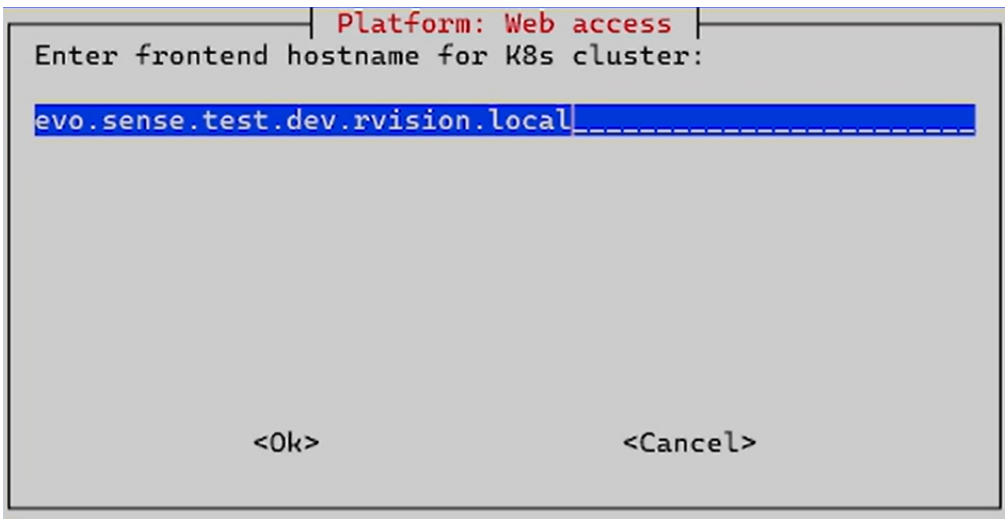
- Допустимые символы: `a–z`, `0–9`, `-`.
- Название не должно начинаться и заканчиваться дефисом.

Настройка доменного имени

Введите доменное имя для внешних запросов на этапе **Web access**.

Если система будет установлена в кластер **из одного узла**, следует указать полное доменное имя (FQDN) виртуальной машины, на которой функционирует кластер.

Если система будет установлена в кластер **из нескольких узлов**, в качестве доменного имени **нельзя** использовать FQDN master-узла кластера. Используйте доменное имя, не занятое ни одним другим сервисом.



Доменное имя должно удовлетворять следующим требованиям:

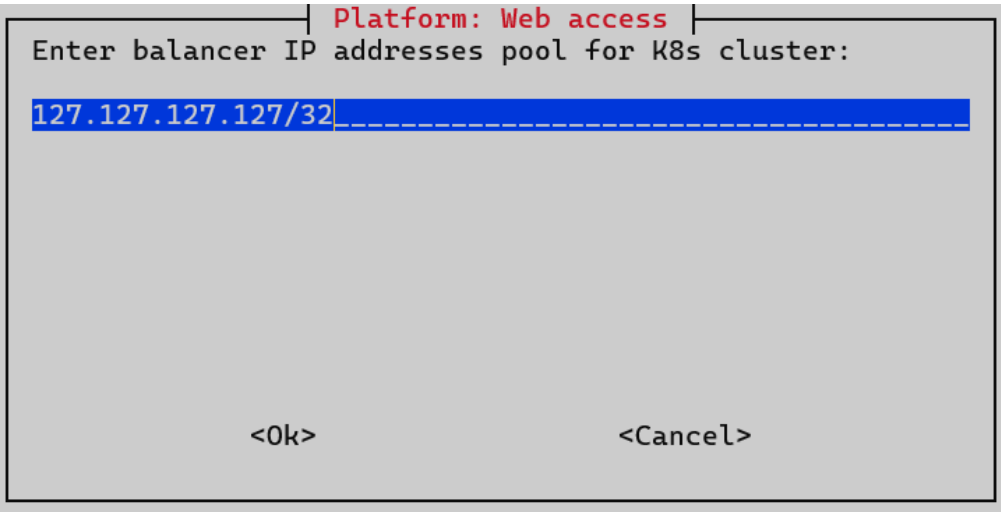
- Допустимые символы: `a–z`, `0–9`, `-`, `.`.
- Имя не должно начинаться и заканчиваться дефисом или точкой.

Пример допустимого доменного имени: `siem.company.local`.

Настройка пула адресов балансировщика

Настраивать пул адресов балансировщика требуется, только если выполняется распределенная установка в кластере, где есть хотя бы один master-узел и один worker-узел. При установке системы на один сервер данный этап будет пропущен.

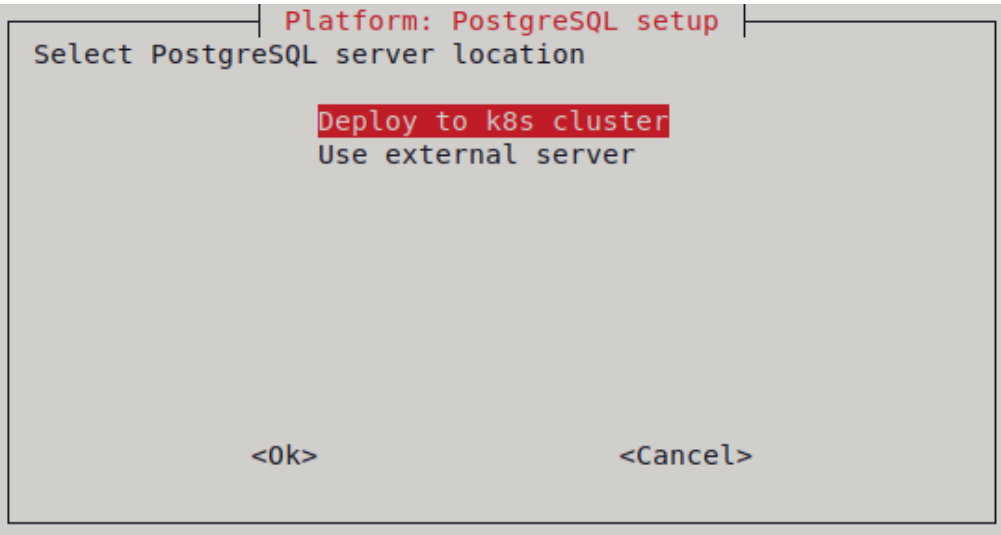
Введите пул адресов балансировщика на этапе **Web access**. Этот пул должен быть определен перед установкой системы на этапе настройки доменного имени. При использовании диапазона из одного адреса нужно указать префикс сети `/32`.



Настройка PostgreSQL

На данном этапе потребуется учетная запись пользователя PostgreSQL с правами на создание новых баз данных и пользователей. Для этого можно использовать суперпользователя `postgres` или создать нового.

Задайте параметры PostgreSQL на этапе **PostgreSQL setup**.



Выберите расположение:

- **Deploy to k8s cluster** — в кластере Kubernetes.
- **Use external server** — внешнее расположение в выделенном кластере.

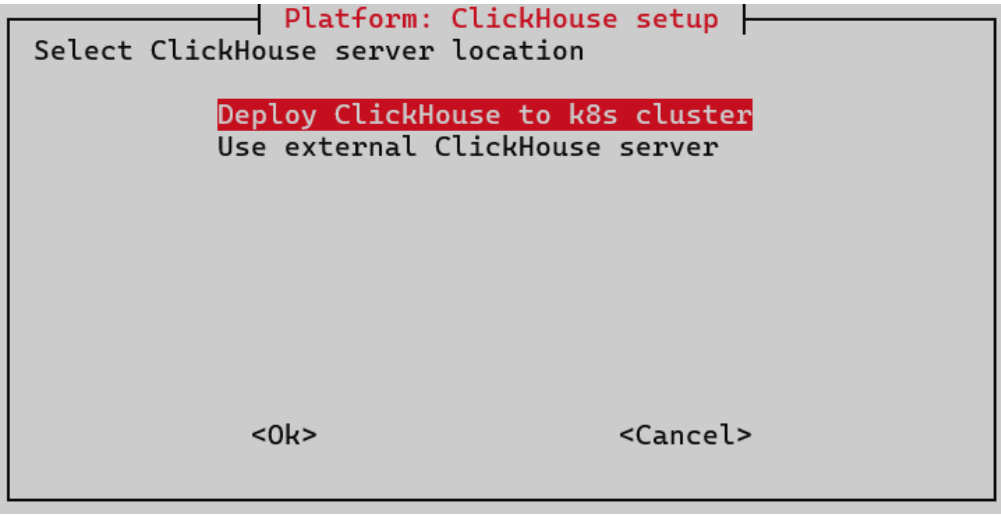
При установке в **кластер Kubernetes** дополнительные параметры не требуются.

Для расположения на **выделенном сервере** укажите следующие параметры для доступа к PostgreSQL:

- имя хоста или IP-адрес сервера PostgreSQL;
- порт сервера;
- имя базы данных по умолчанию;
- имя пользователя;
- пароль пользователя.

Настройка ClickHouse

Задайте параметры ClickHouse на этапе **ClickHouse setup**.



Выберите расположение:

- **Deploy ClickHouse to k8s cluster** — в кластере Kubernetes.
- **Use external ClickHouse server** — внешнее расположение на выделенном сервере или в выделенном кластере. Для этого выделенный сервер или кластер должен быть предварительно настроен.

Вне зависимости от расположения ClickHouse укажите, требуется ли включить шифрование трафика между системой и кластером ClickHouse:

- **Use standard connection without encryption** — шифрование трафика не требуется.
- **Use secure connection over TLS** — будет включено TLS-шифрование трафика.



Для установки в **кластер Kubernetes** укажите следующие параметры:

- Количество реплик и шардов ClickHouse.

> Количество реплик и шардов требуется указывать, только если в кластере установлено более одного worker-узла.

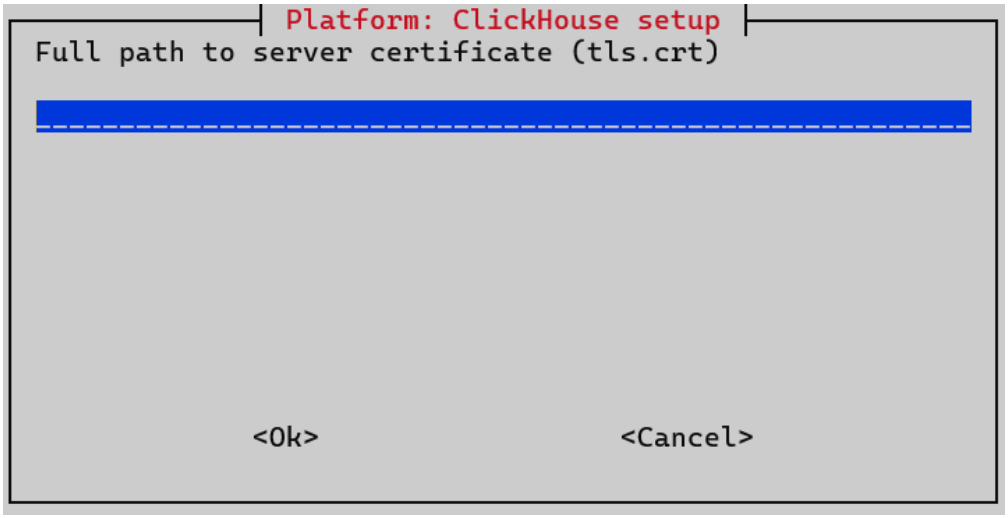
- Максимальный размер хранилища ClickHouse в ГБ.

> Если в кластере не установлен провайдер хранилища или установлен local-path-provisioner, максимальный размер хранилища указывать не требуется.

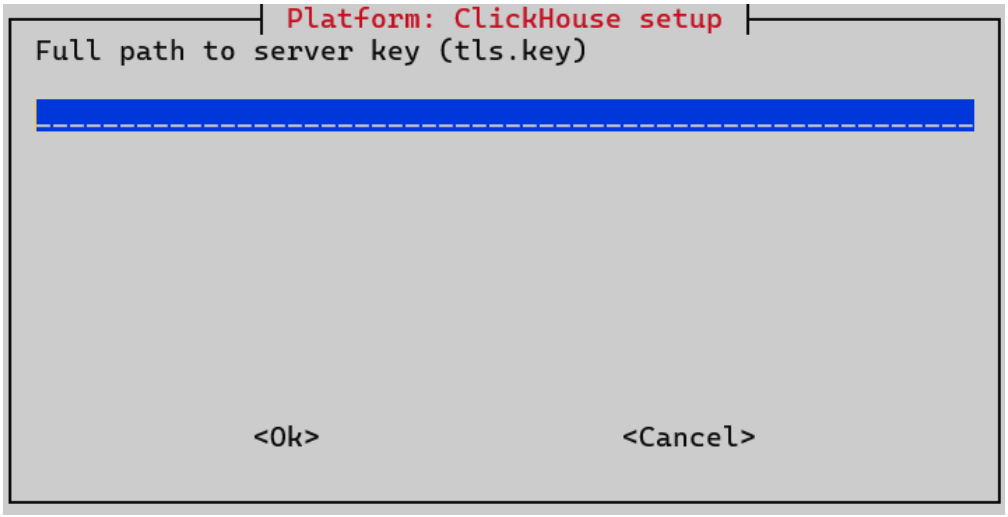
- Если было включено шифрование трафика, выполните следующие действия:
  - Выберите, требуется ли сгенерировать TLS-сертификаты для ClickHouse или необходимо использовать сертификаты, сгенерированные заранее.



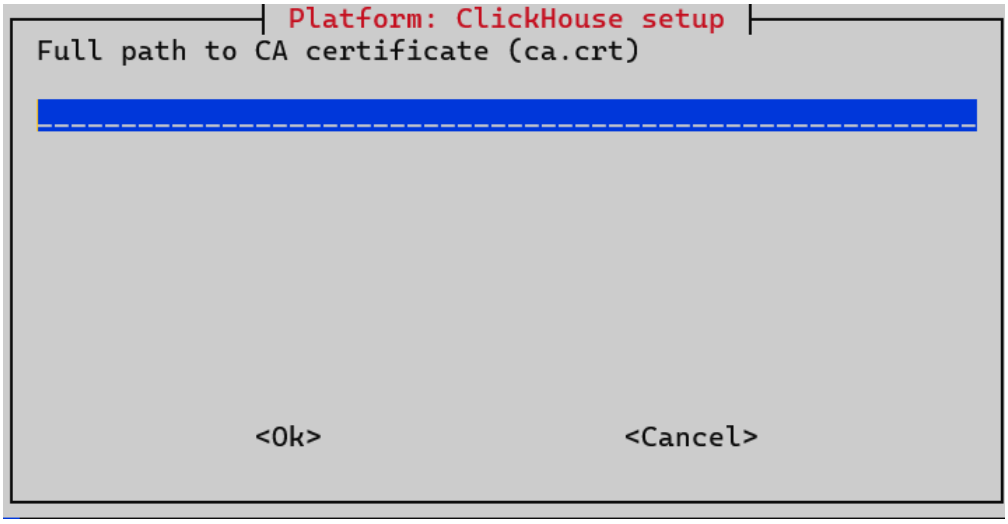
- Если выбрана опция **Provide your own existing certificates**, укажите следующие данные:
  - Абсолютный путь к TLS-сертификату сервера. Например: `/etc/clickhouse-server/certs/tls.crt`.



  - Абсолютный путь к приватному ключу TLS-сертификата сервера. Например: `/etc/clickhouse-server/certs/tls.key`.

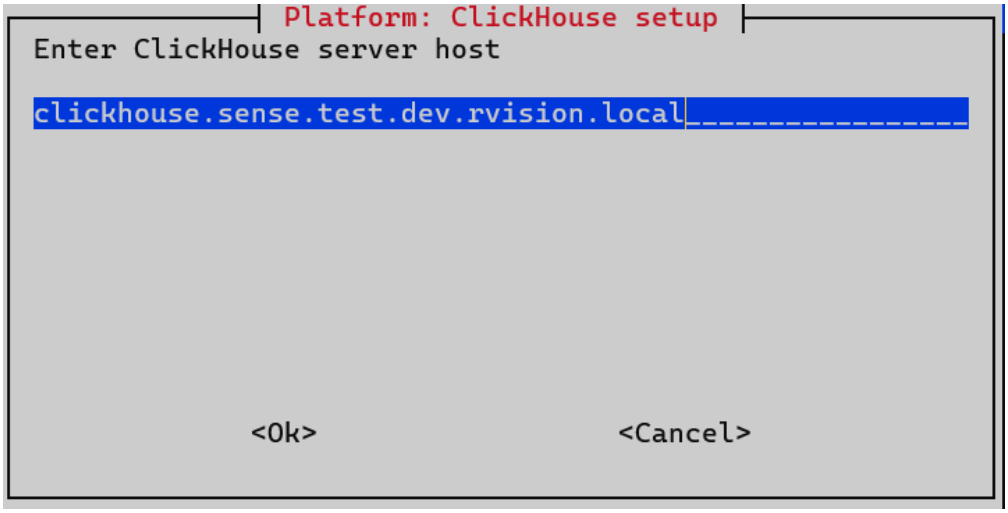


  - Абсолютный путь к TLS-сертификату центра сертификации. Например: `/etc/clickhouse-server/certs/ca.crt`.

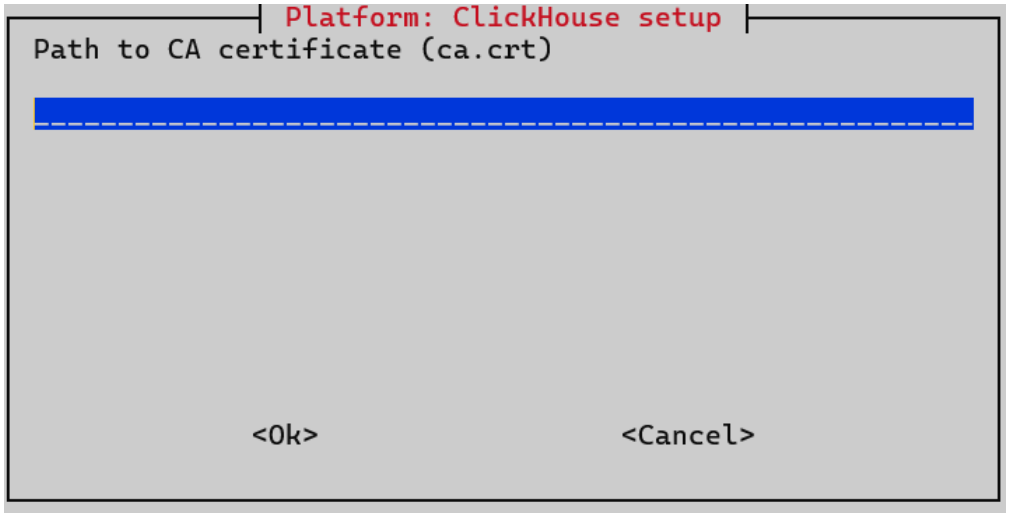


Для **внешнего расположения** укажите следующие параметры:

- Доменное имя сервера или балансировщика, если он настроен.

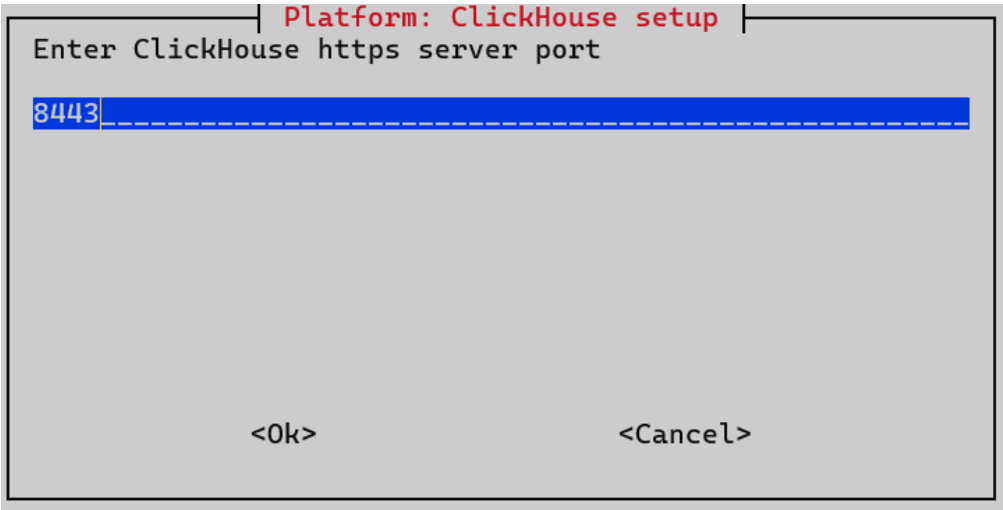


- Если было включено шифрование трафика, введите следующие данные:
  - Абсолютный путь к TLS-сертификату центра сертификации. Например: `/etc/clickhouse-server/certs/ca-certificate.crt`.

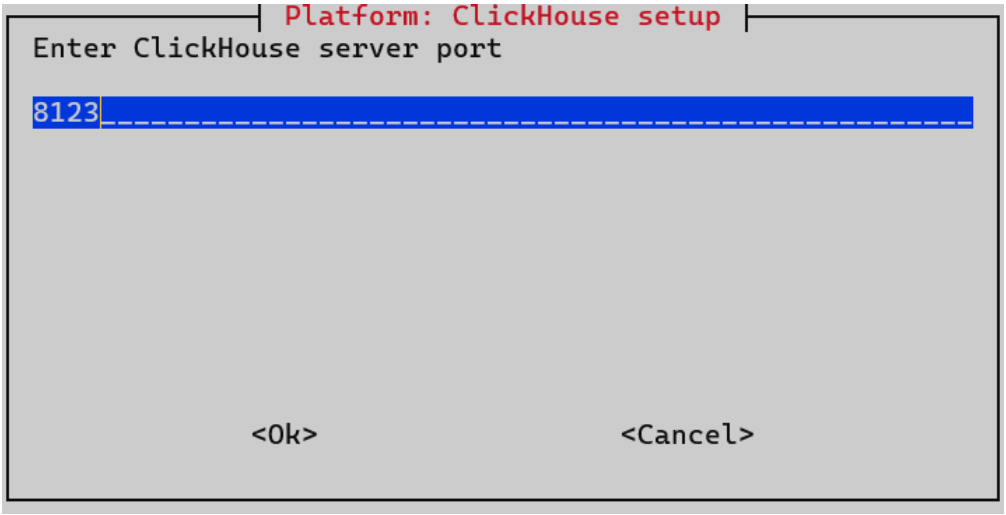


  - Порт для взаимодействия системы и ClickHouse по HTTPS, указанный при настройке

кластера ClickHouse. По умолчанию: 8443.



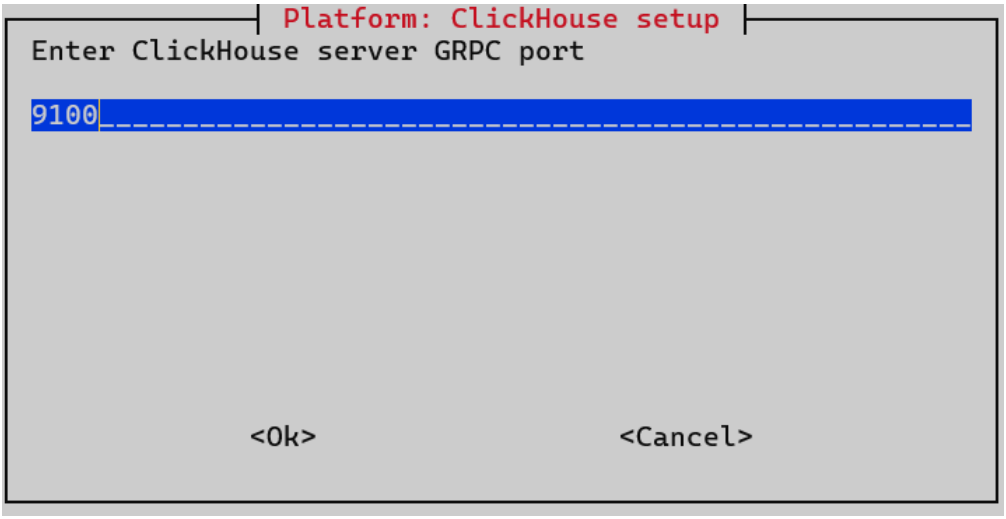
- Порт для взаимодействия системы и ClickHouse по HTTP, указанный при настройке кластера ClickHouse. По умолчанию: 8123.



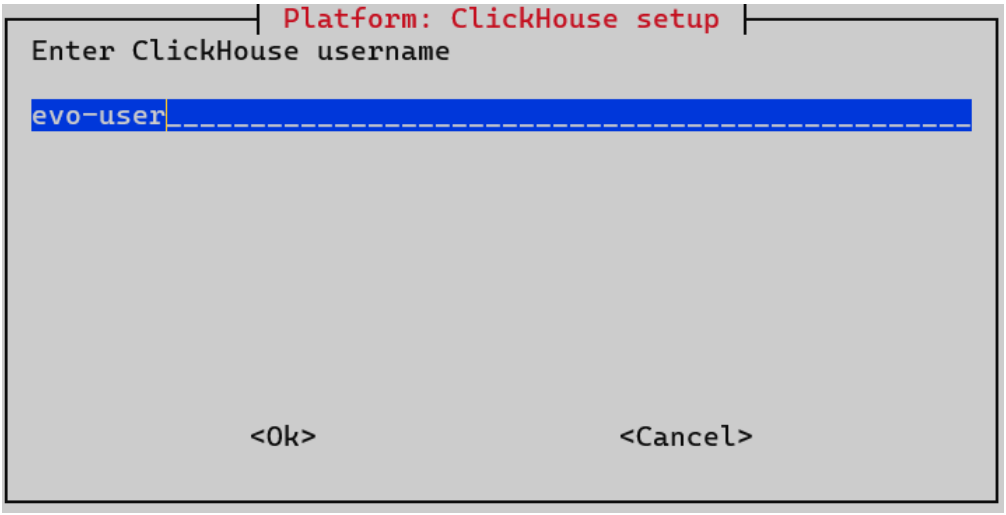
- Порт для взаимодействия системы и ClickHouse по gRPC, указанный при настройке кластера ClickHouse. По умолчанию: 9100.



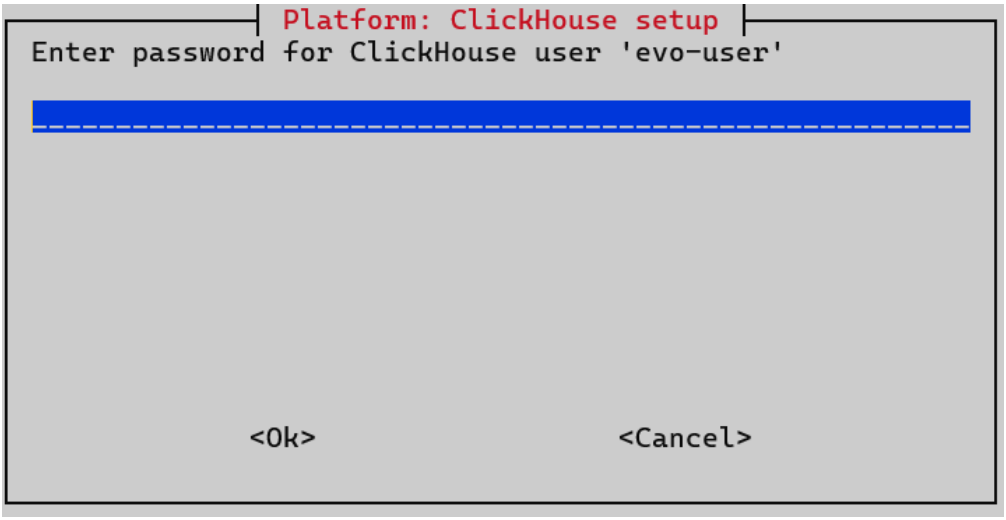
Если было включено шифрование трафика, gRPC-порт будет принимать только зашифрованный трафик.



- Имя системного пользователя.



- Пароль системного пользователя.





В качестве имени и пароля пользователя следует использовать имя и пароль системного пользователя или пользователя writer, созданных при развертывании нового или существующего кластера ClickHouse.

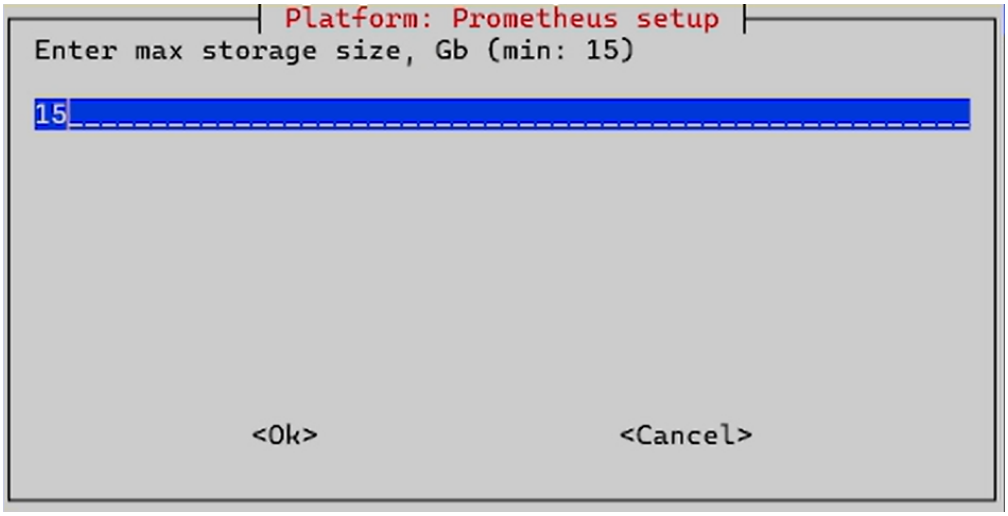
Настройка Prometheus

Prometheus — это система мониторинга, собирающая различные метрики работы кластера.

Задайте максимальный размер хранилища Prometheus на этапе **Prometheus setup**, не менее 15 ГБ.



Если в кластере не установлен провайдер хранилища или установлен local-path-provisioner, максимальный размер хранилища указывать не требуется.



Настройка Loki

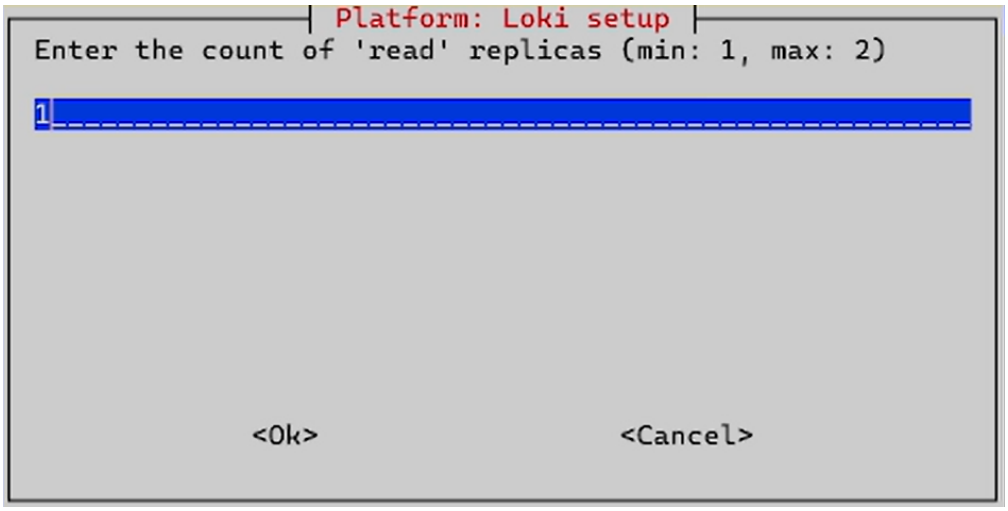
Loki — это система для хранения и просмотра логов.



Количество реплик требуется указывать, только если в кластере установлено более одного worker-узла.

Задайте параметры Loki на этапе **Loki setup**:

1. Задайте количество реплик хранилища для чтения логов.

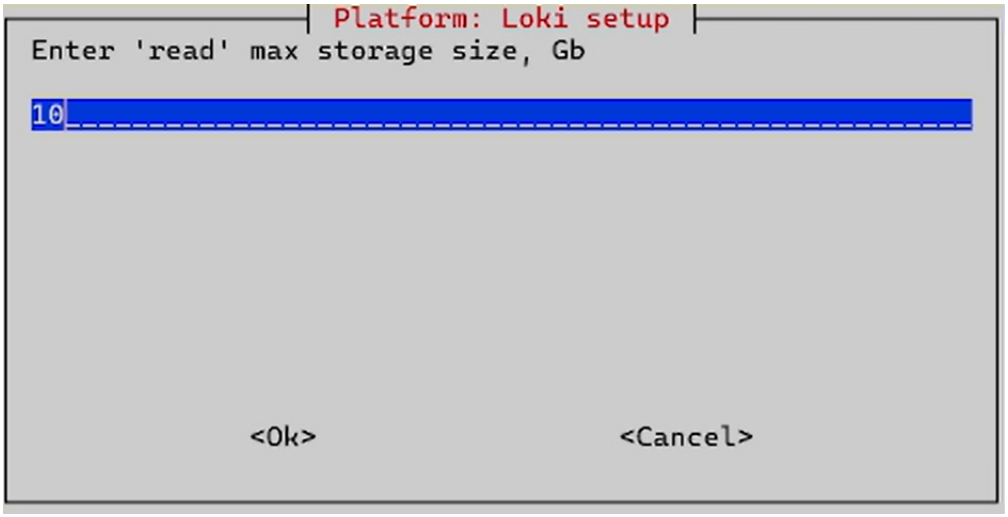


2. Задайте максимальный размер хранилища для чтения логов в ГБ.

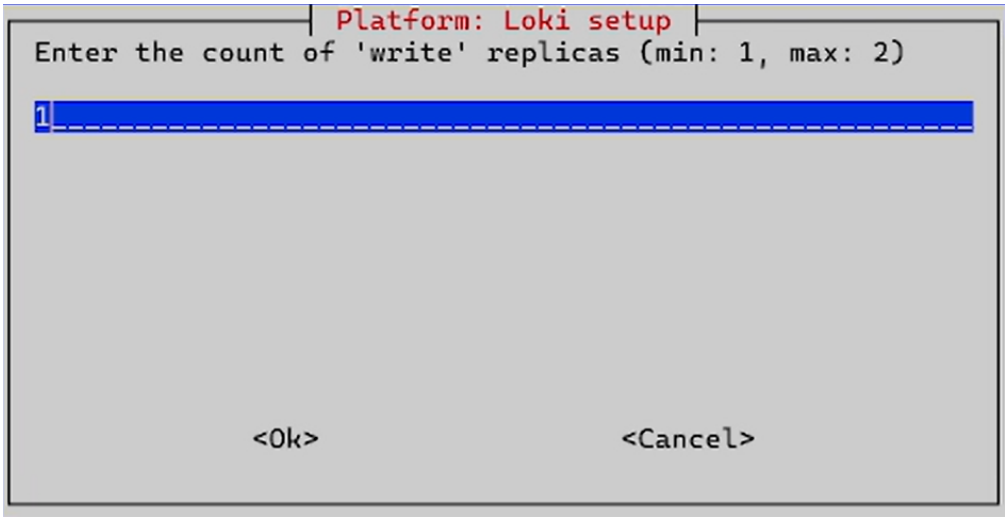


Если в кластере не установлен провайдер хранилища или установлен local-path-

provisioner, максимальный размер хранилища указывать не требуется.



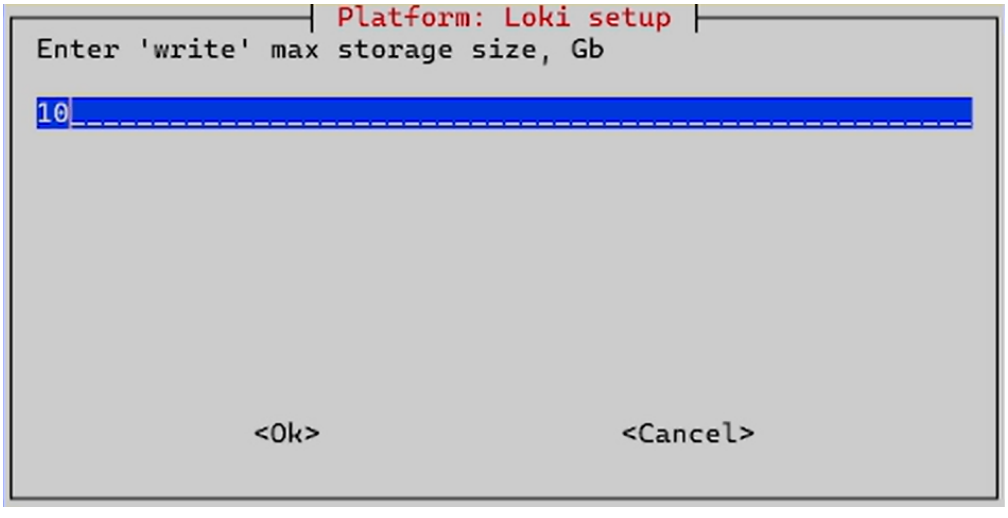
3. Задайте количество реплик хранилища для записи логов.



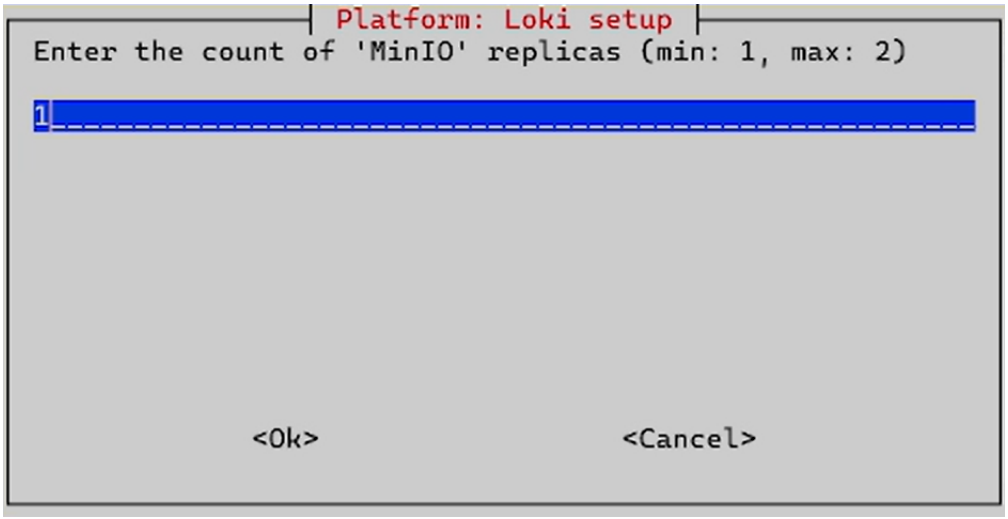
4. Задайте максимальный размер хранилища для записи логов в ГБ.



Если в кластере не установлен провайдер хранилища или установлен local-path-provisioner, максимальный размер хранилища указывать не требуется.

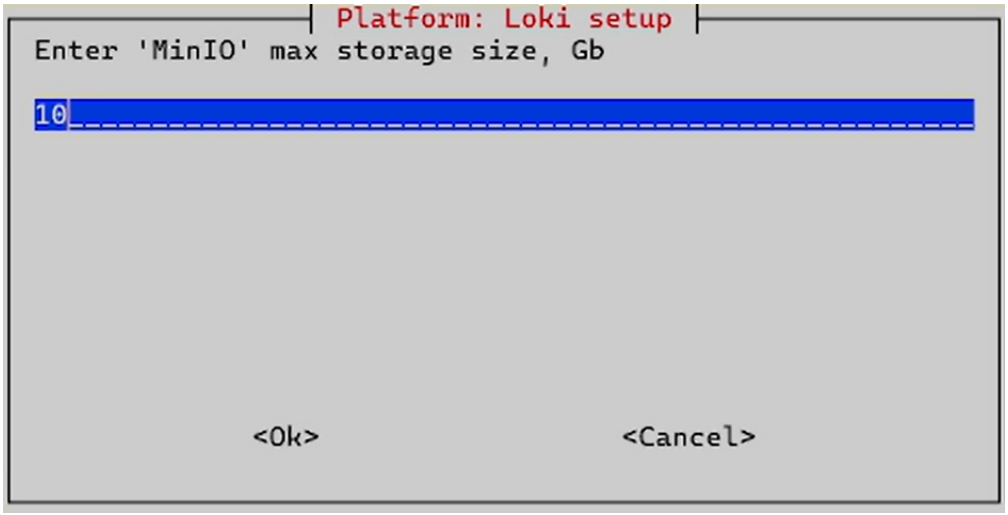


5. Задайте количество реплик MinIO.



6. Задайте максимальный размер хранилища MinIO в ГБ.

> Если в кластере не установлен провайдер хранилища или установлен local-path-provisioner, максимальный размер хранилища указывать не требуется.

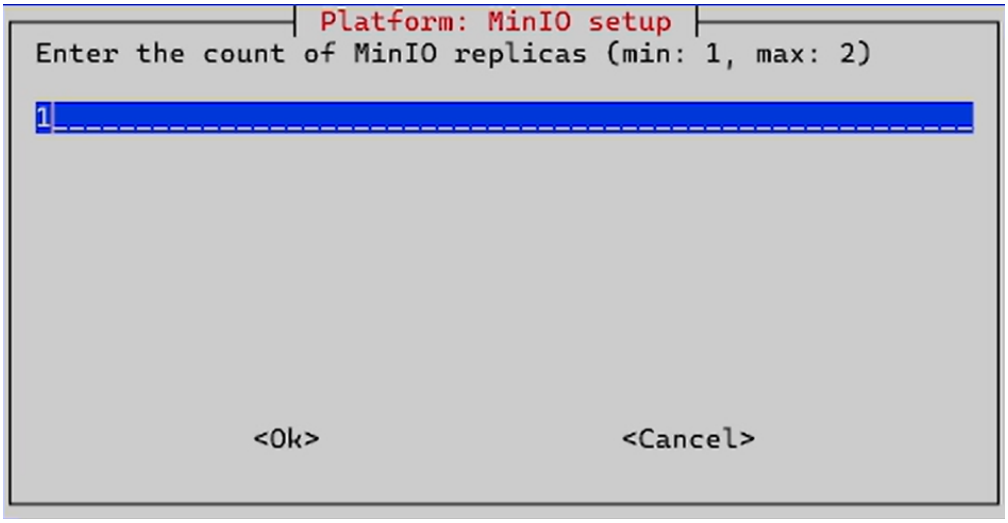


Настройка MinIO

MinIO — это система объектного хранения данных, которая используется для хранения файлов конфигураций, загруженных в систему файлов, а также логов сервисов и инфраструктуры.

На этапе **MinIO setup** укажите количество реплик MinIO.

> Количество реплик требуется указывать, только если в кластере установлено более одного worker-узла.

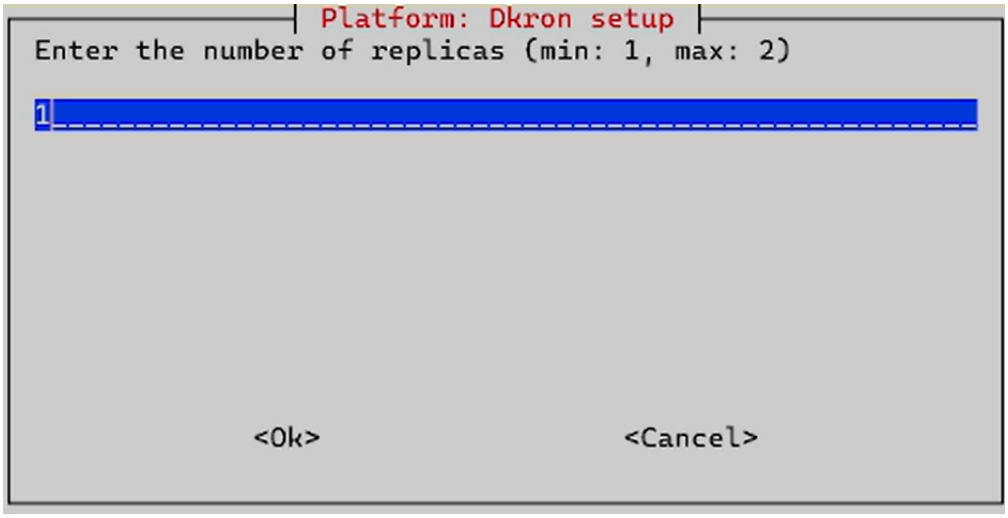


Настройка Dkron

Dkron — это распределенный планировщик задач.

Количество реплик Dkron требуется указывать, только если в кластере установлено более одного worker-узла. В ином случае этап настройки Dkron будет пропущен.

На этапе **Dkron setup** укажите количество реплик Dkron.



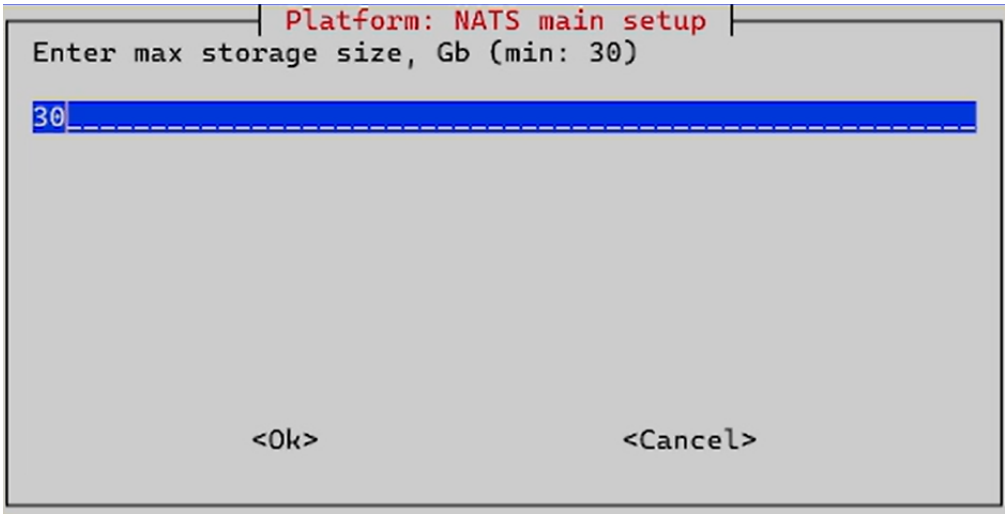
Настройка NATS

NATS — это брокер сообщений, который обеспечивает взаимодействие модулей платформы.

На этапе **NATS main setup** настройте основное хранилище NATS:

1. Задайте максимальный размер хранилища NATS, не менее 30 ГБ.

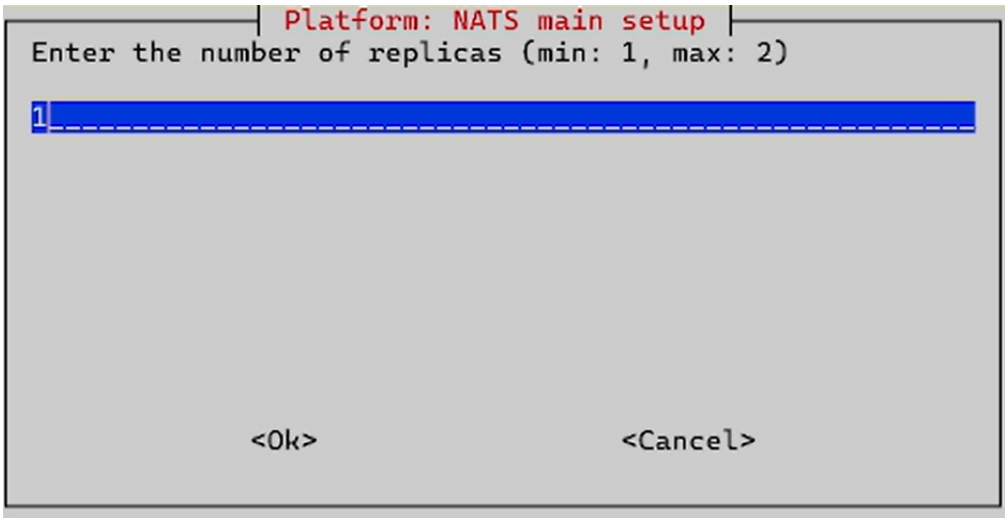
Если в кластере не установлен провайдер хранилища или установлен local-path-provisioner, максимальный размер хранилища указывать не требуется.



2. Укажите количество реплик NATS.



Количество реплик требуется указывать, только если в кластере установлено более одного worker-узла.

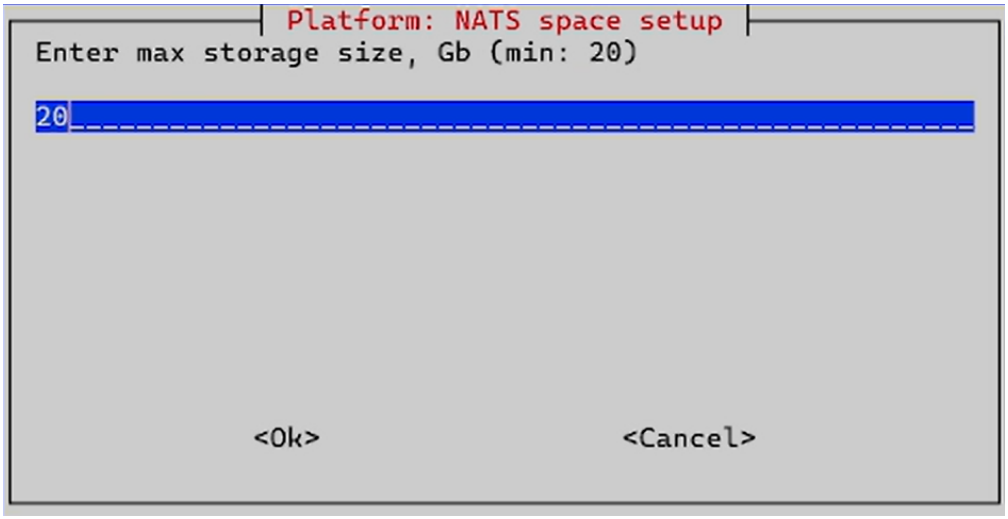


На этапе **NATS space setup** настройте хранилище NATS для управления пространствами:

1. Задайте максимальный размер хранилища NATS, не менее 20 ГБ.



Если в кластере не установлен провайдер хранилища или установлен local-path-provisioner, максимальный размер хранилища указывать не требуется.



2. Укажите название пространства, которое будет использовать NATS.

Идентификатор основного пространства (`main space ID`) формируется по шаблону:

```
space-main-<namespace>
```

Здесь:

- `<namespace>` — пространство имен, заданное на этапе **Kubernetes cluster namespace to install**.

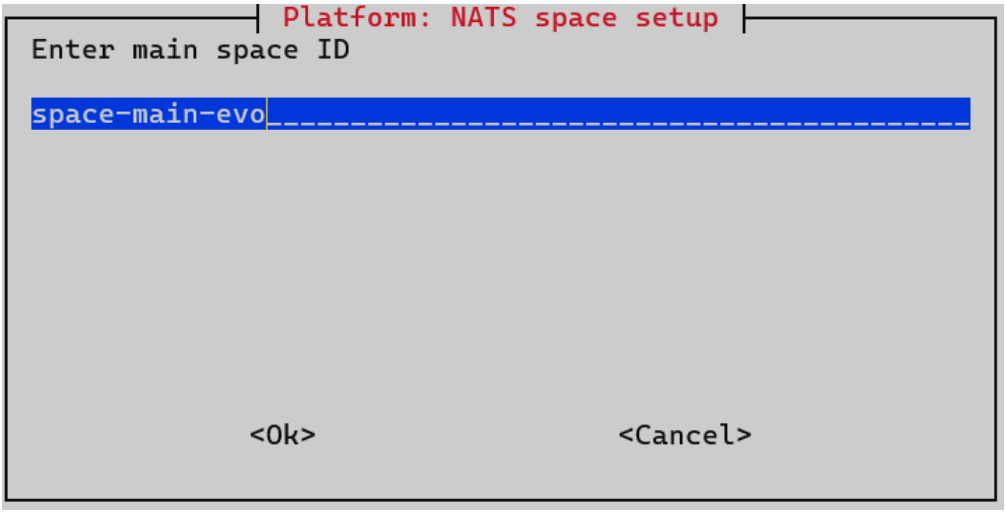


В большинстве случаев изменение идентификатора основного пространства не требуется.

Если идентификатор основного пространства требуется изменить, для него существуют следующие требования:

- Допустимые символы: прописные и строчные латинские буквы, цифры, знак дефиса.
- Идентификатор должен содержать не более 100 символов.

При развертывании нескольких экземпляров системы в одном кластере Kubernetes необходимо обеспечить уникальность идентификаторов основного пространства.

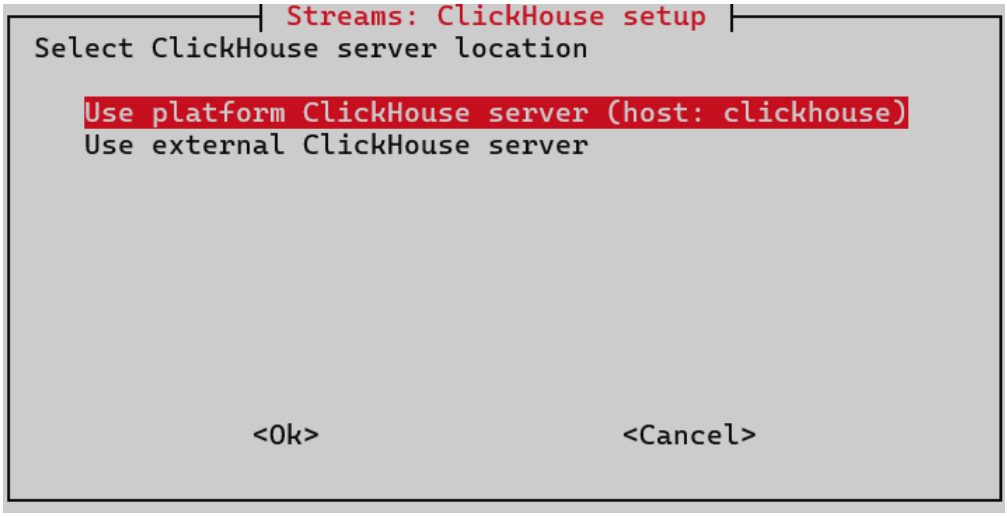


Настройка СУБД для работы с событиями ИБ

Для работы с событиями ИБ можно использовать ранее установленные инсталляции ClickHouse и PostgreSQL, общие для всех компонентов платформы, или использовать отдельные выделенные серверы.

Настройка ClickHouse для событий ИБ

На этапе **Streams: ClickHouse setup** выберите расположение ClickHouse для событий ИБ.



Доступные варианты:

- **Use platform ClickHouse server** — использование общего сервера ClickHouse, настроенного ранее.
- **Use external ClickHouse server** — внешнее расположение на выделенном сервере или в выделенном кластере. Для этого выделенный сервер или кластер должен быть предварительно настроен.

Для работы системы необходимо, чтобы в ClickHouse были настроены три пользователя с разным набором прав:

- **owner** — пользователь с расширенным набором прав. Используется для создания, обновления и удаления таблиц (хранилищ событий), а также получения служебной информации.
- **writer** — пользователь с правами на запись. Используется в коллекторах для записи событий в

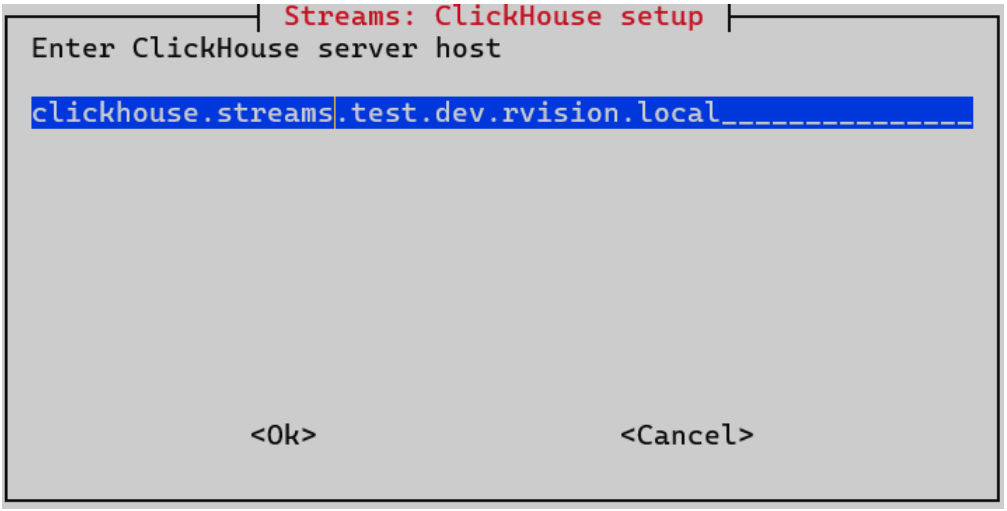
ClickHouse.

- **reader** — пользователь с правами только на чтение. Используется для выполнения поисковых запросов.

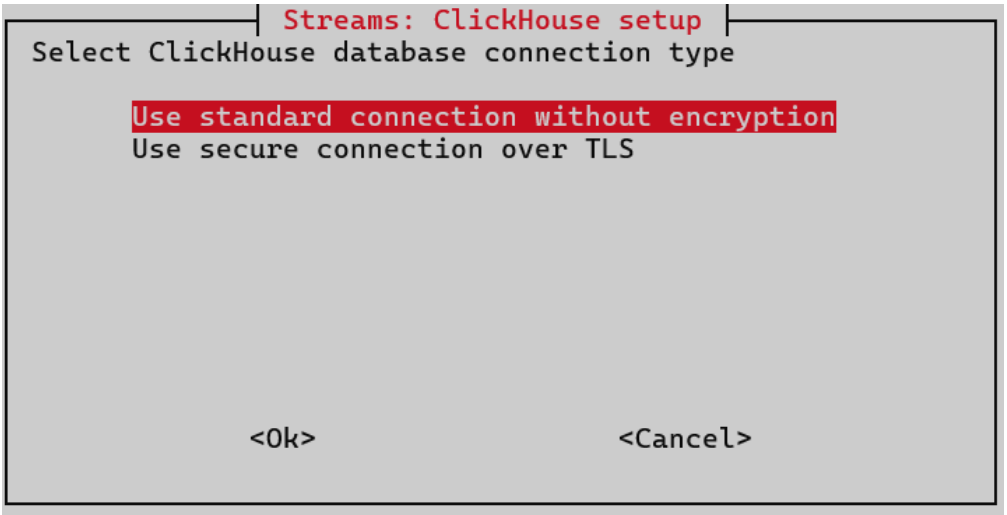
Вне зависимости от варианта размещения ClickHouse для событий ИБ следует указать пароли этих пользователей, заданные при развертывании кластера ClickHouse.

Для **внешнего расположения** укажите следующие параметры:

- Доменное имя сервера или балансировщика, если он настроен.

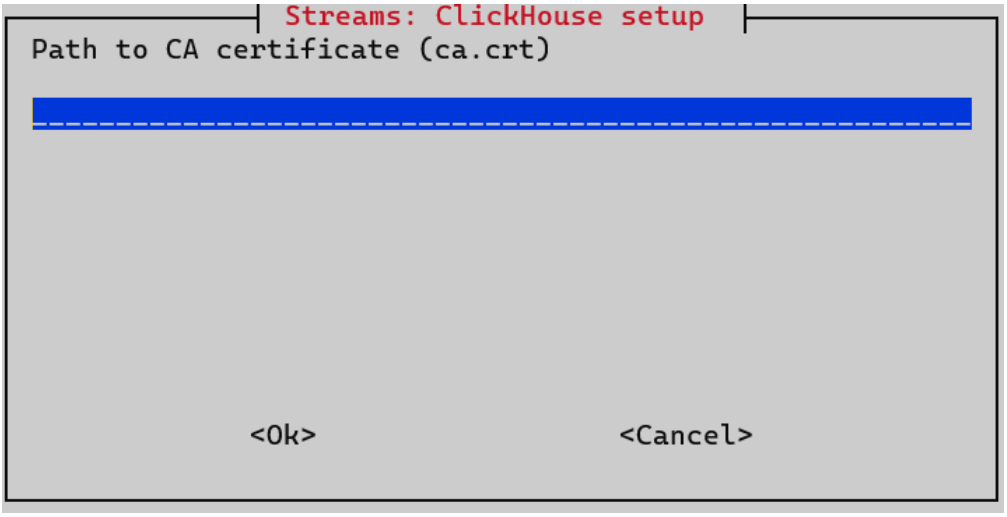


- Требуется ли включить шифрование трафика между системой и кластером ClickHouse:
  - **Use standard connection without encryption** — шифрование трафика не требуется.
  - **Use secure connection over TLS** — будет включено TLS-шифрование трафика.

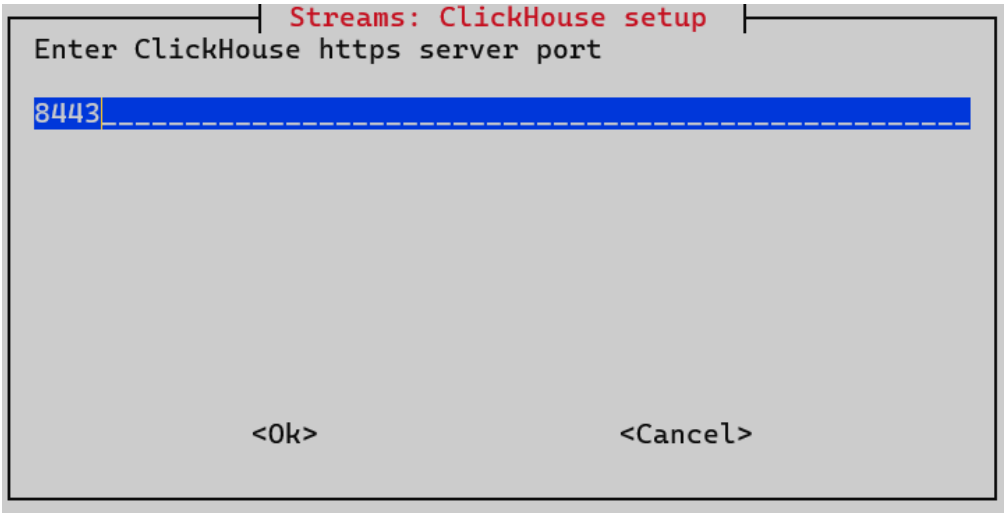


Если выбрана опция **Use secure connection over TLS**, потребуется ввести дополнительные данные:

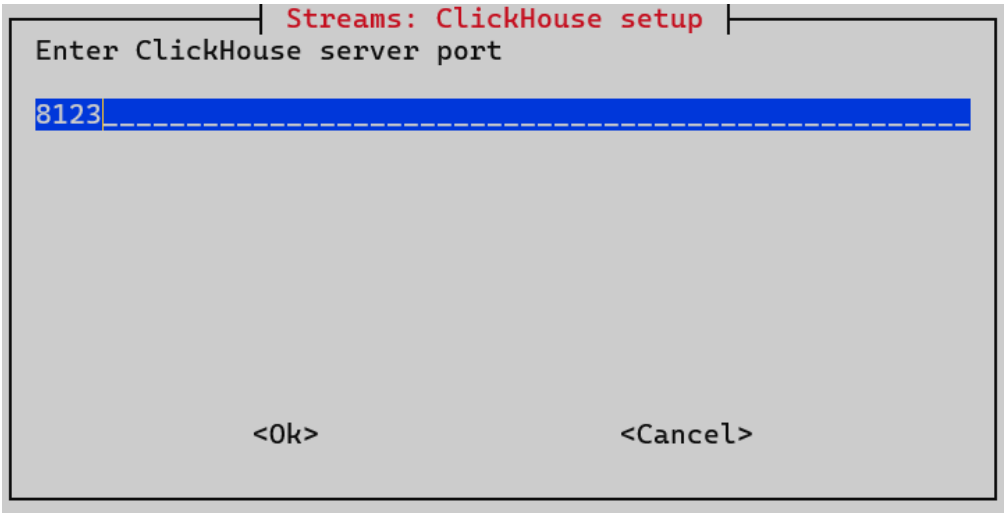
- Абсолютный путь к TLS-сертификату центра сертификации. Например: `/etc/clickhouse-server/certs/ca-certificate.crt`.



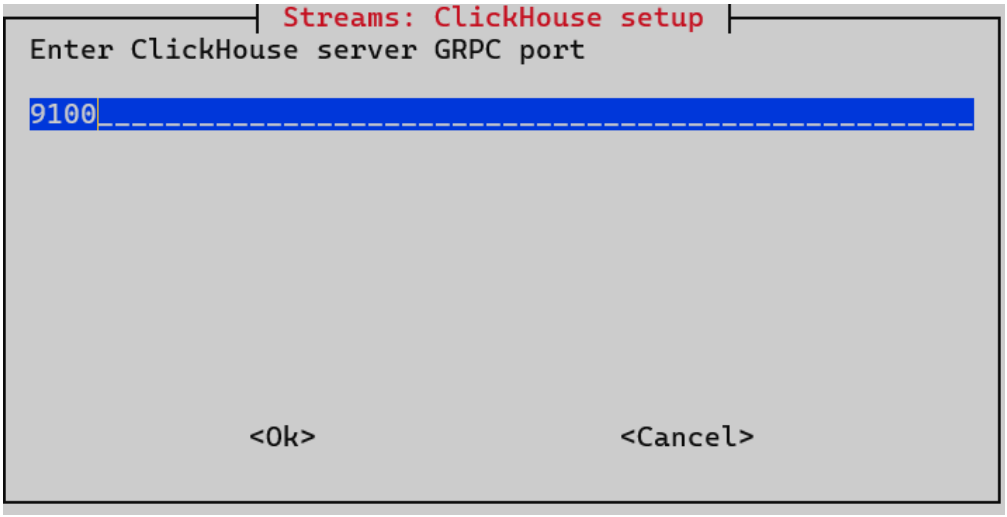
- Порт для взаимодействия системы и ClickHouse по HTTPS, указанный при настройке кластера ClickHouse. По умолчанию: 8443.



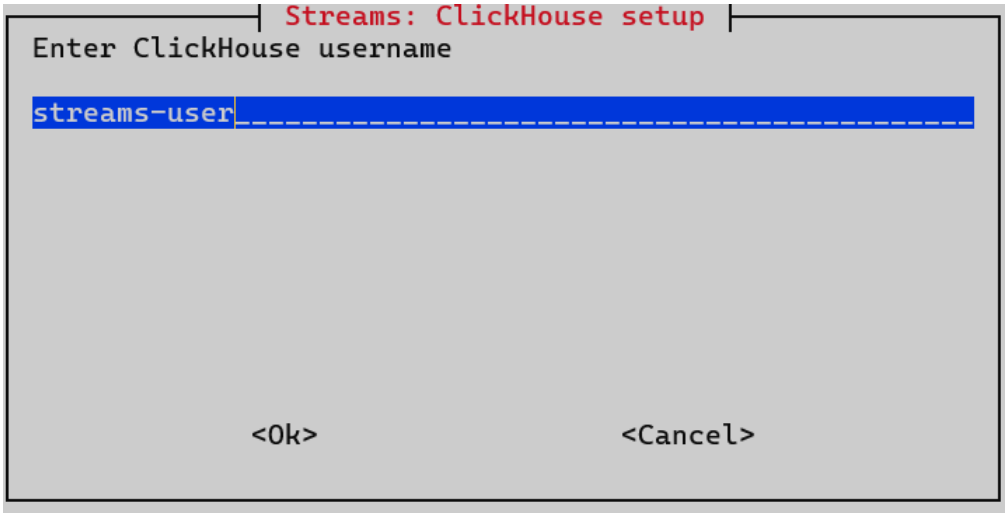
- Порт для взаимодействия системы и ClickHouse по HTTP, указанный при настройке кластера ClickHouse. По умолчанию: 8123.



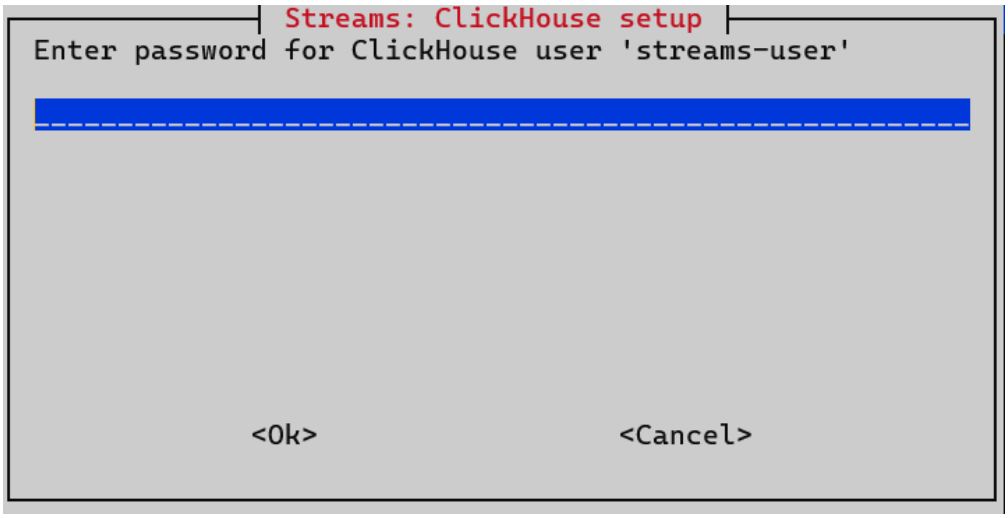
- Порт для взаимодействия системы и ClickHouse по gRPC, указанный при настройке кластера ClickHouse. По умолчанию: 9100.



- Имя системного пользователя.



- Пароль системного пользователя.

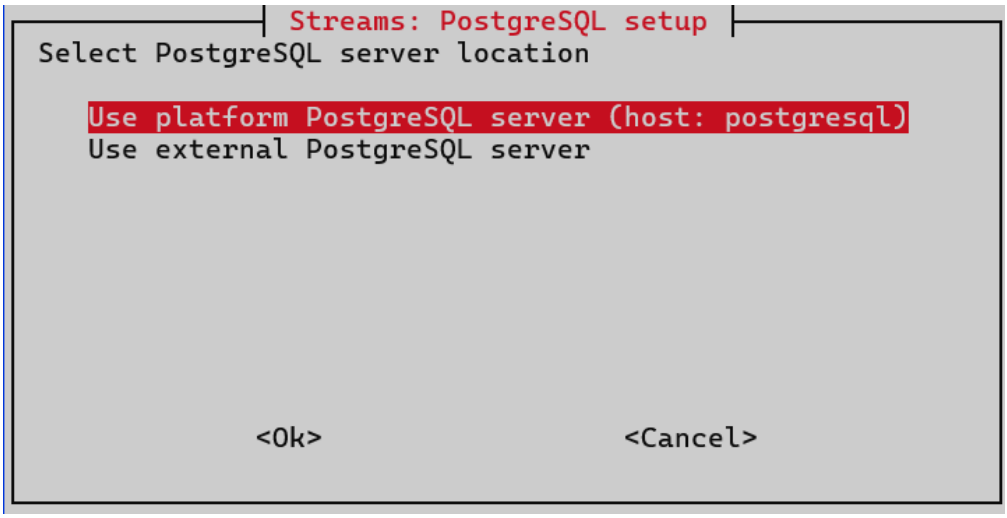




В качестве имени и пароля пользователя следует использовать имя и пароль системного пользователя или пользователя writer, созданных при развертывании нового или существующего кластера ClickHouse.

Настройка PostgreSQL для событий ИБ

На этапе **Streams: PostgreSQL setup** выберите расположение PostgreSQL для событий ИБ.



Доступные варианты:

- **Use platform PostgreSQL server** — использование общего сервера PostgreSQL, настроенного ранее.
- **Use external PostgreSQL server** — внешнее расположение на выделенном сервере.

Для расположения на выделенном сервере также следует задать:

- имя хоста или IP-адрес сервера PostgreSQL;
- порт сервера;
- имя базы данных;
- имя пользователя;
- пароль пользователя.

Завершение установки

Если установка прошла успешно, на экране отобразится соответствующее сообщение. Также на экран будет выведен URL-адрес для подключения к системе и учетные данные по умолчанию для входа в систему.

По завершении установки системы формируется комплект сертификатов, которые потребуются для установки сателлита.

Комплект сертификатов сохраняется в архив `$SETUP_ROOT_DIR/r-vision/nats-tls.tar.gz` (по умолчанию `/opt/r-vision/nats-tls.tar.gz`) на машине, с которой выполнялась установка.

Удаление файлов установщика

Вы можете удалить файлы и каталоги, которые необходимы только во время установки и не требуются для работы системы:

- run-файл установщика:

```
rm evo.platform_<version>.run
```

- Каталог распакованных файлов установки с подкаталогами, в том числе логи процесса установки из каталога `<installation_root>/r-vision/common/logs`:

  ```
  rm -r <installation_root>/r-vision
  ```

  Здесь:

  - `<installation_root>` — корневой каталог, в который выполнялась установка. По умолчанию: `/opt`.

- Архив дополнительного пакета установщика Kubespray:

  ```
  rm -f <kubespray_archive>
  ```

  Здесь:

  - `<kubespray_archive>` — архив установщика Kubespray, например, `kubespray-installer-redos.tar.gz`.

Авторизация в системе

Чтобы получить полное доменное имя (FQDN) системы, используйте следующую команду:

```
kubectl get secret "evo.streams.global" -n <namespace> -o
jsonpath="{.data.MODULE_DOMAIN}" | base64 -d
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

Чтобы выполнить вход в систему:

1. В адресной строке браузера введите полное доменное имя (FQDN) системы. На экране отобразится страница авторизации.
2. Укажите логин и пароль учетной записи пользователя и нажмите на кнопку **Войти**.



По умолчанию для первого входа в систему используются учетные данные **admin**/**admin**.



Установленная система содержит примеры сущностей, таких как база данных событий и элементы экспертизы. Список предустановленных сущностей приведен в

разделе *Начало работы с системой*.

Установка сателлитов

В данном разделе описан процесс установки сателлитов — экземпляров системы, развернутых в дополнительных удаленных кластерах.

Кластер Kubernetes, в котором разворачивается основной экземпляр системы, называется **центральным кластером**. Часть компонентов продукта может быть развернута в дополнительных удаленных кластерах — **кластерах сателлитов**. В отличие от основного экземпляра системы, сателлиты не обладают полным спектром функциональности продукта и могут работать только с теми компонентами, которые в них запущены.



С подробной информацией о сателлитах можно ознакомиться в разделе Управление пространствами.



Основной экземпляр системы обязательно должен быть установлен перед установкой сателлита.

Чтобы установить сателлит, выполните следующие действия:

1. Убедитесь, что окружение, в котором будет установлен сателлит, соответствует техническим требованиям.
2. Разверните кластер Kubernetes, отдельный от кластера основного экземпляра системы. Процесс развертывания аналогичен развертыванию центрального кластера.
3. Выполните подготовительные действия перед установкой сателлита: создайте каталог установки, настройте сетевой доступ к кластеру сателлита со стороны основного экземпляра системы и скачайте дистрибутив.
4. Установите сателлит.

Чтобы обновить сателлит, обратитесь к разделу Обновление системы. Процесс обновления сателлита аналогичен обновлению основного экземпляра системы.

Чтобы удалить сателлит, обратитесь к разделу Удаление системы. Процесс удаления сателлита аналогичен удалению основного экземпляра системы.

Технические требования к сателлитам

В данном разделе представлены минимальные технические требования для установки и функционирования сателлита системы.

- Программные требования
- Аппаратные требования

Программные требования

Поддерживаемые операционные системы

В случае сателлитов требования к операционной системе узлов кластера не предъявляются.

Требования к Kubernetes

- Минимальная версия Kubernetes: 1.24.
- Зарегистрированное полное доменное имя (FQDN) на каждом узле кластера.



Убедитесь, что FQDN узла разрешается в IP-адрес. Это можно проверить, например, с помощью утилиты nslookup:

```
nslookup <FQDN>
```

Здесь:

- `<FQDN>` — FQDN текущего узла.

Установка Kubernetes на узлы с адресами localhost.localdomain, localhost и 127.0.0.1 не предусмотрена.

- Поддерживаемая среда выполнения контейнеров (Container Runtime Interface, CRI): containerd, CRI-O.
- Установка сателлитов допускается на базе K0s.

Выбор среды выполнения имеет значение для автономной установки без Docker-registry в инфраструктуре.

Системные утилиты

Утилита	Минимальная версия	Наличие на узлах Kubernetes и АРМ администратора
ansible	2.9	Требуется.
bash	4.0	Требуется.
curl	Любая.	Требуется только на АРМ администратора.
diff	Любая.	Требуется только на АРМ администратора.
gunzip	Любая.	Требуется.
kubectl	Та же, что и версия Kubernetes, с отличием не более чем на один минор. Например, для Kubernetes 1.24 допустимы kubectl 1.23, 1.24 и 1.25.	Требуется только на АРМ администратора.
openssl	Любая.	Требуется только на АРМ администратора.

Утилита	Минимальная версия	Наличие на узлах Kubernetes и АРМ администратора
python	3.8	Требуется.
sed	Любая.	Требуется только на АРМ администратора.
tar	Любая.	Требуется только на АРМ администратора.

АРМ администратора — автоматизированное рабочее место администратора.



Если установка запускается непосредственно на узле кластера Kubernetes, то к нему применяются требования АРМ администратора.

Аппаратные требования

Требования для работы установщика

Свободное место на диске: от 10 ГБ.

Требования для узлов кластера Kubernetes



Приведены требования для минимальной инсталляции сателлита, при которой возможна его нестабильная работа. Чтобы получить рекомендации по конфигурации сателлита для вашей инсталляции, обратитесь в службу поддержки по адресу support@rvision.ru.

Возможны следующие варианты инсталляции сателлита:

- **Моно** — кластер Kubernetes из одного узла, выполняющего роль master-узла и worker-узла одновременно.
- **Распределенный** — кластер из нескольких узлов, содержащий как минимум один master-узел и один worker-узел.



Если при распределенной установке сателлита в кластере имеется только один worker-узел, то к нему применяются требования, аналогичные моно-инсталляции.

Ресурс	Моно	Распределенный, master-узел	Распределенный, worker-узел
CPU, ядер	8	4	6
RAM, ГБ	8	4	6
Место на диске, ГБ	100	100	100

Подготовка к установке сателлита

В данном разделе описаны подготовительные действия, которые необходимо выполнить перед установкой сателлита.

Подготовка состоит из следующих этапов:

1. Создание каталога установки.
2. Добавление TLS-сертификатов для подключения к основному экземпляру системы.
3. Настройка доступа к кластеру Kubernetes.
4. Настройка доступа к worker-узлам кластера.
5. Настройка сетевой связности между основным экземпляром системы и сателлитом.
6. Применение дополнительных настроек, если узлы кластера работают под управлением ОС Альт Сервер или Альт СП Сервер.
7. Скачивание архива с файлами для установки и проверка его целостности.

Создание каталога установки

Для работы установщика требуется распаковать содержимое дистрибутива в локальный каталог.



Если установка основного экземпляра системы и сателлита выполняется с одного АРМ администратора, то следует использовать для сателлита отдельный корневой каталог.

Например:

```
SETUP_ROOT_DIR=/opt/satellite ./evo.platform_<version>_satellite.run
```

Это позволит избежать смешивания установочных файлов основного экземпляра системы и сателлита в одном каталоге.

Далее в инструкции предполагается, что используется каталог по умолчанию.

По умолчанию распаковка выполняется в каталог `/opt`, например `/opt/r-vision`. Корневой каталог (`/opt` по умолчанию) можно изменить с помощью переменной `SETUP_ROOT_DIR`.

Таким образом:

- `/opt` — корневой каталог установщика;
- `/opt/r-vision` — каталог производителя.

Если пользователь не имеет права записи в каталог `/opt`, необходимо предварительно создать каталог производителя и назначить его владельцем пользователя, от имени которого будет выполняться установка.

Пример создания каталога установки

```
sudo mkdir -pv /opt/r-vision
sudo chown -R <user>:<group> /opt/r-vision
```

Здесь:

- `<user>` — имя пользователя;
- `<group>` — группа, в которую входит пользователь.

Добавление TLS-сертификатов для подключения к основному экземпляру системы

По завершении установки основного экземпляра системы создается архив TLS-сертификатов `$SETUP_ROOT_DIR/r-vision/nats-tls.tar.gz`. Этот архив необходимо скопировать в каталог установки сателлита `$SETUP_ROOT_DIR/r-vision` до начала его установки.



Каталоги установки (значения переменной `$SETUP_ROOT_DIR`) основного экземпляра системы и сателлита могут отличаться.

Настройка доступа к кластеру Kubernetes

До начала установки должен быть развернут кластер Kubernetes.

Установка может выполняться как с одного из master-узлов кластера Kubernetes, так и с машины администратора кластера. На машине, с которой будет выполняться установка, должны быть соблюдены следующие требования:

1. Установлена утилита ansible.
2. В системных переменных окружения указан путь к файлу `kubeconfig`.

Linux

В ОС Linux путь к файлу `kubeconfig` по умолчанию — `$HOME/.kube/config`. Вы можете изменить его с помощью переменной окружения `KUBECONFIG`:

```
export KUBECONFIG=<path_to_confg>
```

Здесь:

- `<path_to_config>` — путь к файлу `kubeconfig`.

Windows

В ОС Windows путь к файлу `kubeconfig` по умолчанию — `$%USERPROFILE%\.kube\config`. Вы можете изменить его с помощью переменной окружения `KUBECONFIG`:

```
$env:KUBECONFIG = <path_to_confg>
```

Здесь:

- `<path_to_config>` — путь к файлу `kubeconfig`.

Проверить доступ к кластеру можно, выполнив следующую команду:

```
kubectl get nodes
```

Если доступ к кластеру есть, отобразится список узлов кластера, например:

```
NAME                    STATUS   ROLES           AGE    VERSION   INTERNAL-IP
EXTERNAL-IP   OS-IMAGE
master01.company.local   Ready    control-plane   140d   v1.26.3   10.10.10.1
<none>        RED OS MUROM (7.3.2)
worker01.company.local   Ready    <none>          140d   v1.26.3   10.10.10.2
<none>        RED OS MUROM (7.3.2)
worker02.company.local   Ready    <none>          140d   v1.26.3   10.10.10.3
<none>        RED OS MUROM (7.3.2)
worker03.company.local   Ready    <none>          140d   v1.26.3   10.10.10.4
<none>        RED OS MUROM (7.3.2)
```

Настройка доступа к worker-узлам кластера

Для загрузки образов контейнеров на worker-узлы установщик использует Ansible. Необходимо настроить SSH-доступ с машины установщика до worker-узлов под единой учетной записью. Поддерживается доступ по ключам или паролю.



Если на master-узле кластера установлена ОС Альт Сервер или Альт СП Сервер и планируется доступ по паролю, установите утилиту sshpass на master-узле:

```
sudo apt-get install sshpass
```

Для настройки SSH-доступа выполните следующие действия на каждом worker-узле кластера:

1. Пользователю, от имени которого будет установлен кластер, необходимо выдать права суперпользователя (root). Для этого добавьте следующую строку в файл `/etc/sudoers`:

   ```
   <user> ALL=(ALL:ALL) NOPASSWD: ALL
   ```

   Здесь:

   - `<user>` — пользователь, от имени которого будет установлен кластер.

2. Настройте SSH-доступ по паролю или по ключу:
   - Если планируется доступ по паролю:
     a. Установите пакет sshpass:

     РЕД ОС

     ```
     sudo dnf install sshpass
     ```

Альт Сервер, Альт СП Сервер, Astra Linux

```
sudo apt-get install sshpass
```

b. Проверьте возможность подключения к узлу по SSH:

```
ssh <user>@<fqdn>
```

Здесь:

- `<user>` — пользователь, от имени которого будет установлен кластер.
- `<fqdn>` — полное доменное имя (FQDN) текущего узла.



Чтобы узнать FQDN текущего узла, выполните команду:

```
hostname -f
```

◦ Если планируется доступ по ключу:

a. Проверьте возможность подключения к узлу по SSH:

```
ssh <user>@<fqdn>
```

Здесь:

- `<user>` — пользователь, от имени которого будет установлен кластер.
- `<fqdn>` — полное доменное имя (FQDN) текущего узла.



Чтобы узнать FQDN текущего узла, выполните команду:

```
hostname -f
```

b. Если подключение по SSH-ключу не настроено, сгенерируйте ключи и добавьте открытый ключ на узел:

```
ssh-keygen
ssh-copy-id <user>@<fqdn>
```

Здесь:

- `<user>` — пользователь, от имени которого будет установлен кластер.
- `<fqdn>` — полное доменное имя (FQDN) текущего узла.

Настройка сетевой связности между основным экземпляром системы и сателлитом

Следует разрешить TCP-подключения со стороны кластера, в котором установлен основной экземпляр системы, ко всем узлам кластера сателлита.



Допустимые значения номера порта подключения: `30000–32767`. Значение по умолчанию: `30072`. Это значение может быть изменено в процессе установки сателлита.

Дополнительные настройки для Альт Сервер и Альт СП Сервер

Если узлы кластера работают под управлением ОС Альт Сервер или Альт СП Сервер, выполните следующие действия на master-узле:

1. Установите коллекцию ansible.posix:

   ```
   ansible-galaxy collection install ansible.posix
   ```

2. Скачайте пакеты libnewt52 и newt52 из репозитория Альт:

   ```
   wget
   http://ftp.altlinux.org/pub/distributions/ALTLinux/Sisyphus/x86_64/RPMS.classic/lib
   newt52-0.52.24-alt1.x86_64.rpm
   wget
   http://ftp.altlinux.org/pub/distributions/ALTLinux/Sisyphus/x86_64/RPMS.classic/new
   t52-0.52.24-alt1.x86_64.rpm
   ```

3. Установите пакеты libnewt52 и newt52:

   ```
   rpm -i libnewt52-0.52.24-alt1.x86_64.rpm
   rpm -i newt52-0.52.24-alt1.x86_64.rpm
   ```

4. Расширьте системную переменную `PATH`, чтобы утилиты, расположенные в каталоге `<setup_dir>/r-vision/common/bin`, были доступны из любой директории без указания полного пути:

   ```
   export PATH="$PATH:<setup_dir>/r-vision/common/bin"
   ```

   Здесь:

   - `<setup_dir>` — каталог, в который будет установлена система. По умолчанию: `/opt`.

Получение дистрибутива

Скачайте архив с установочными файлами `evo.platform_<version>_satellite.run` и проверьте его контрольную сумму. Ссылка на дистрибутив предоставляется по запросу в службу поддержки по

адресу support@rvision.ru.

```
# Получение дистрибутива.
curl -O
https://files.rvision.ru/platform/<random_part>/evo.platform_<version>_satellite.run

# Получение контрольной суммы.
curl -O
https://files.rvision.ru/platform/<random_part>/evo.platform_<version>_satellite.run.s
ha256

# Проверка контрольной суммы.
sha256sum -c evo.platform_<version>_satellite.run.sha256

# При успешной проверке контрольной суммы будет выведено следующее сообщение.
evo.platform_<version>_satellite.run: OK

# При нарушении контрольной суммы будет выведено следующее сообщение.
evo.platform_<version>_satellite.run: FAILED
```



При выявлении нарушения контрольной суммы выполнять установку **недопустимо**.

Необходимо убедиться, что скачивание run-файла дистрибутива было полностью завершено, и повторно проверить контрольную сумму.

При подтверждении нарушения контрольной суммы следует сообщить в техническую поддержку по адресу support@rvision.ru.

Процесс установки сателлита

В данном разделе описаны действия по выполнению новой установки сателлита.

Установка состоит из следующих этапов:

1. Запуск установщика
2. Настройка Ansible
3. Выбор пространства имен Kubernetes
4. Настройка NATS
5. Завершение установки
6. Настройка mTLS-шифрования между основным экземпляром и сателлитом
7. Подключение к пространству сателлита

Запуск установщика

Чтобы запустить установщик сателлита:

1. После скачивания run-файла перейдите в директорию его хранения и запустите его с помощью

bash:

```
bash ./evo.platform_<version>_satellite.run
```



Требуемая версия bash: 4.0 или выше.

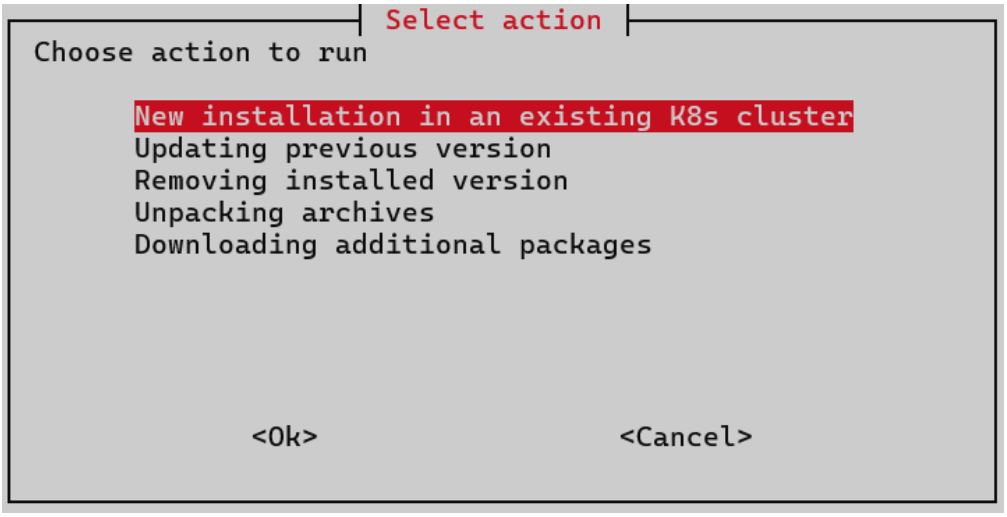
Или сделайте его исполняемым и запустите непосредственно:

```
chmod +x ./evo.platform_<version>_satellite.run
./evo.platform_<version>_satellite.run
```

Установщик выведет следующие сведения:

- версия системы и дата ее выпуска;
- путь к файлу хранения логов установки;
- результаты проверок:
  - удовлетворения системным требованиям;
  - наличия необходимых утилит;
  - целостности дистрибутива.

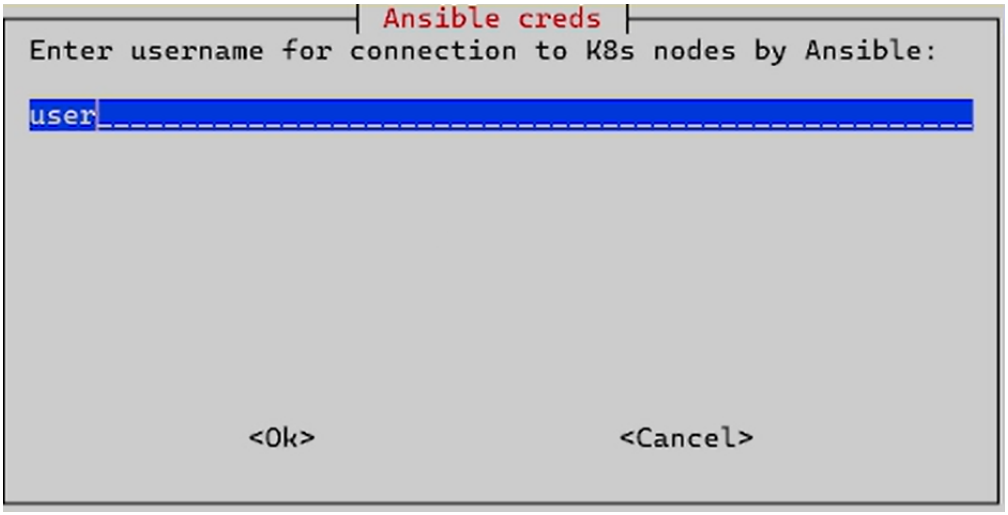
2. На этапе **Select action** выберите действие **New installation in an existing K8s cluster**:



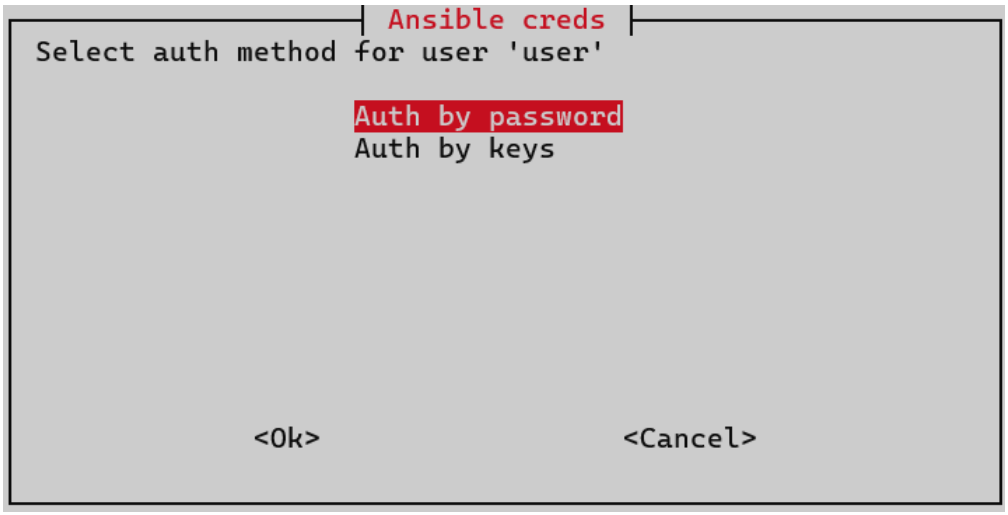
Настройка Ansible

Задайте параметры Ansible на этапе **Ansible creds**:

1. Задайте имя пользователя.



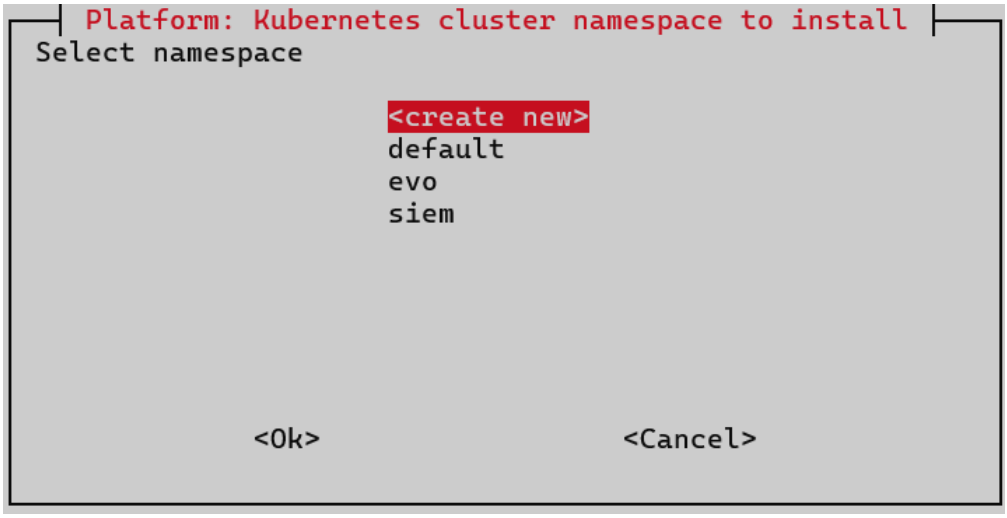
2. Выберите метод аутентификации: по паролю или через ключ.



3. Если выбрана аутентификация по паролю, задайте и подтвердите пароль пользователя.

Выбор пространства имен Kubernetes

Выберите пространство имен (namespace) Kubernetes, в которое будет установлен сателлит, на этапе **Kubernetes cluster namespace to install**.



Если требуется создать новое пространство имен, выберите опцию **create new** и введите название

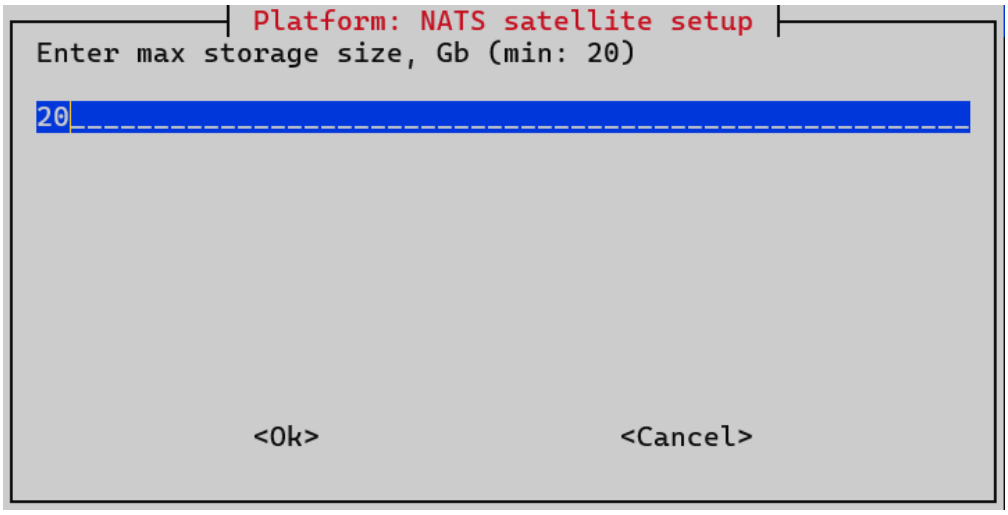
пространства.

Настройка NATS

На этапе **NATS satellite setup**:

1. Укажите максимальный размер хранилища NATS, не менее 20 ГБ.

   Если в кластере не установлен провайдер хранилища или установлен local-path-provisioner, максимальный размер хранилища указывать не требуется.

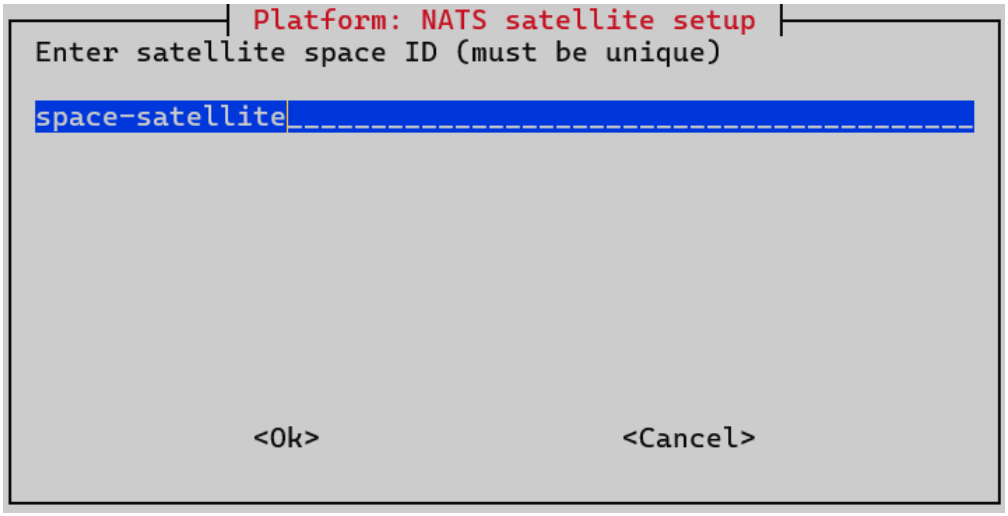


2. Укажите уникальный идентификатор пространства сателлита.

   С помощью одного дистрибутива можно установить несколько сателлитов, однако у каждого из них идентификатор пространства должен быть уникальным.

   Требования к идентификатору пространства сателлита:

   - Допустимые символы: прописные и строчные латинские буквы, цифры, знак дефиса.
   - Идентификатор должен содержать не более 100 символов.



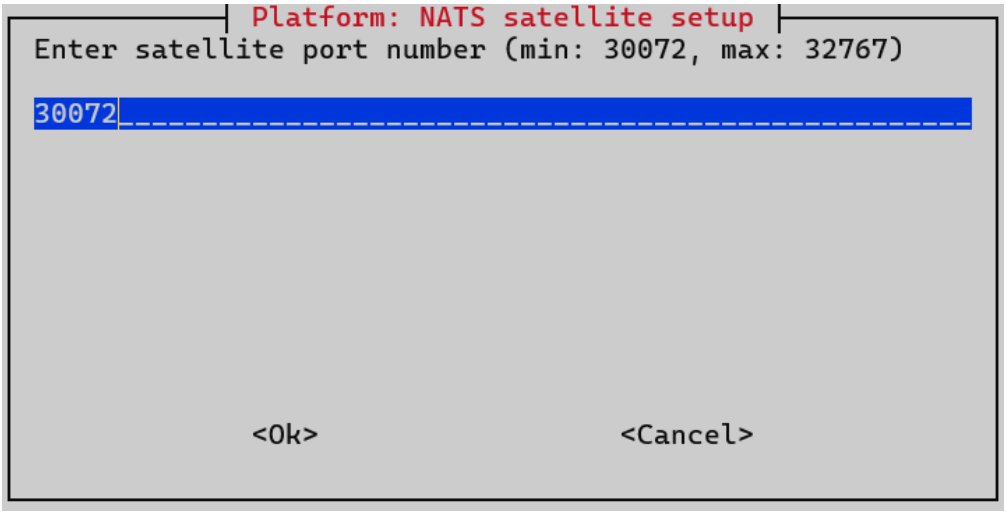
   Сохраните идентификатор пространства, он понадобится для подключения к

пространству сателлита из интерфейса системы.

3. Укажите номер TCP-порта, по которому основной экземпляр системы будет подключаться к сателлиту. Данный порт будет открыт на всех узлах кластера сателлита.



Допустимые значения номера порта подключения: `30000–32767`. Значение по умолчанию: `30072`. Сателлиты могут использовать одинаковые порты при условии установки в разные кластеры Kubernetes.





Сохраните номер порта, он понадобится для подключения к пространству сателлита из интерфейса системы.

Завершение установки

По завершении установки в терминале отобразится сообщение, содержащее идентификатор пространства и номер порта установленного сателлита, например:

```
To add this Satellite as a new space of main instance use values:
  space id: space-satellite
  space port: 30072
```

Настройка mTLS-шифрования между основным экземпляром и сателлитом

По умолчанию после установки основного экземпляра системы и сателлита шифрование соединения между ними отключено.

При необходимости вы можете настроить шифрование соединения по протоколу mTLS. Для этого:

1. Включите TLS-шифрование для пространств. Для этого:
   a. Подключитесь к терминалу master-узла кластера, в котором установлен основной экземпляр системы.
   b. Откройте ConfigMap `space-manager-extravars`:

```
kubectl -n <namespace> edit configmap space-manager-extravars
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

c. Установите для переменной `NATS_CONFIGURATOR_USE_LEAF_TLS_FOR_SPACES` значение `"true"`.

d. Сохраните изменения в ConfigMap.

e. Перезапустите Deployment `evo.space.space-manager`:

```
kubectl -n <namespace> rollout restart deployment evo.space.space-manager
```

2. Задайте параметры TLS в ConfigMap `nats-space-config`. Для этого:

a. Откройте ConfigMap `nats-space-config`:

```
kubectl edit -n <namespace> configmaps nats-space-config -o yaml
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

b. В блоке `leafnodes` добавьте блок `tls` со следующим содержимым:

```
"tls": {
  "ca_file": "/etc/nats/ca/ca.crt",
  "cert_file": "/etc/nats/tls/tls.crt",
  "key_file": "/etc/nats/tls/tls.key",
  "verify": true,
  "handshake_first": true
}
```

▼ *Пример измененного блока* `leafnodes`

```
"leafnodes": {
  "no_advertise": true,
  "port": 7422,
  "tls": {
    "ca_file": "/etc/nats/ca/ca.crt",
    "cert_file": "/etc/nats/tls/tls.crt",
    "key_file": "/etc/nats/tls/tls.key",
    "verify": true,
    "handshake_first": true
```

```
  }
},
```

c. Перезапустите StatefulSet `nats-space`:

```
kubectl -n <namespace> rollout restart statefulset nats-space
```

3. Повторите действия на шаге 2 в терминале master-узла кластера сателлита.

Подключение к пространству сателлита

Управление сателлитом осуществляется за счет запускаемых в нем сервисов через веб-интерфейс основного экземпляра системы. Чтобы подключиться к сателлиту, необходимо создать подключение к его пространству. Для этого:

1. Войдите в веб-интерфейс основного экземпляра системы.
2. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**.
3. Нажмите на кнопку **Создать** (┼). Отобразится окно создания подключения.
4. Введите идентификатор пространства сателлита в поле **ID**.
5. Введите название подключения. Название может быть любым.
6. В поле **IP или домен** введите IP-адрес или доменное имя (FQDN) одного из узлов кластера, в котором развернут сателлит.
7. В поле **Порт** введите номер TCP-порта, указанного на этапе **NATS satellite setup** установки сателлита.
8. При необходимости введите описание подключения к пространству.
9. Нажмите на кнопку **Создать**. Система создаст подключение к пространству и отобразит соответствующее уведомление. Подключение отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Пространства**.

У созданного подключения отобразится статус **Подключается**. По завершении подключения статус изменится на **Активен**.



Время на инициализацию подключения зависит загруженности кластеров и каналов связи и может достигать 5—10 минут.

Обновление системы

В данном разделе описаны действия по обновлению основного экземпляра системы или сателлита (далее — "экземпляр системы").

Обновление состоит из следующих этапов:

1. Подготовка к обновлению
2. Обновление системы

3. Действия после обновления
4. Авторизация в системе



Если к основному экземпляру системы подключены сателлиты, нужно обновить их после обновления основного экземпляра системы. Инструкция по обновлению сателлитов приведена в разделе Обновление сателлитов.

Подготовка к обновлению

Подготовительные действия перед обновлением



В процессе обновления системы базы данных и сервисы могут быть временно недоступны. Работоспособность системы будет автоматически восстановлена по завершении обновления.

Перед обновлением экземпляра системы до новой версии выполните подготовительные действия:

1. Минимизируйте операции в системе, связанные с изменениями в базах данных и NATS, такие как активный сбор событий или изменения конфигурации системы.
2. Проверьте соответствие программных и аппаратных требований для работы установщика новой версии экземпляра системы.
3. Убедитесь, что у вас установлена версия ClickHouse, поддерживаемая новой версией системы. Если ваша версия ClickHouse отличается от поддерживаемой, обновите ClickHouse до требуемой версии.
4. Скачайте архив с файлами для установки новой версии экземпляра системы и проверьте его целостность.

Смена доменного имени системы

При необходимости перед началом обновления вы можете изменить доменное имя, по которому доступен веб-интерфейс системы. Для этого:

1. Подключитесь к терминалу master-узла кластера Kubernetes, в котором установлен основной экземпляр системы.
2. Обновите доменное имя в секрете `evo.infra.global`, выполнив следующую команду:

   ```
   kubectl patch secret evo.infra.global -n <namespace>
   -p='{"stringData":{"FRONTEND_HOST": "<new_domain_name>","GRAFANA_EXTERNAL_URL":
   "https://<new_domain_name>/grafana"}}'
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<new_domain_name>` — новое доменное имя, например, `evo-new.company.local`.

   В результате успешного обновления секрета должно отобразиться следующее сообщение:

```
secret/evo.infra.global patched
```

3. Запустите обновление. После обновления система будет доступна по новому доменному имени.

Особенности обновления системы с определенных версий

При обновлении системы с определенных версий имеется ряд особенностей:

▼ *Особенности обновления системы с версии ниже 2.0.0*

В системе версии ниже 2.0.0 может возникать ошибка при просмотре логов коллектора в интерфейсе:

```
Не удалось получить логи коллектора.
connect ECONNREFUSED <...>
```

В логах пода `loki-gateway` отображается ошибка:

```
host not found in resolver "coredns.kube-system.svc.cluster.local." in
/etc/nginx/nginx.conf:33
```

Причина: чарт Loki по умолчанию использует имя `coredns` для конфигурации разрешения имен пода `loki-gateway`, однако в кластере может использоваться другое имя для сервиса DNS — например, `kube-dns`.

В системе версии 2.0.0 и выше данный параметр определяется автоматически, но при обновлении с более старой версии проблема может воспроизводиться, поэтому перед обновлением следует ее устранить.

Решение: нужно исправить конфигурацию сервиса `loki-gateway`. Для этого:

1. Получите фактическое имя DNS-сервиса:

   ```
   K8S_DNS_SERVICE="$(kubectl get service -n kube-system -l k8s-app=kube-dns -o
   'jsonpath={.items..metadata.name}')"
   ```

2. Получите конфигурацию `loki-gateway`:

   ```
   LOKI_CONFIG="$(get cm loki-gateway -n kube-prometheus-stack -o
   'jsonpath={.data.nginx\.conf}')"
   ```

3. Сравните имена DNS-сервисов:

   ```
   echo "K8s DNS: $K8S_DNS_SERVICE"
   echo "Loki DNS: $(echo "$LOKI_CONFIG" | grep resolver)"
   ```

Пример вывода имен DNS-сервисов

```
K8s DNS: kube-dns
Loki DNS: resolver coredns.kube-system.svc.cluster.local.;
```

Если в выводе имена сервисов разные, перейдите к шагу 4.

4. Откройте ConfigMap `loki-gateway`:

   ```
   KUBE_EDITOR=nano kubectl edit configmap/loki-gateway -n kube-prometheus-stack
   ```

   Замените имя DNS-сервиса `coredns` после `resolver` на значение переменной `K8S_DNS_SERVICE`. После исправления строка ConfigMap должна принять вид:

   ```
   resolver <dns_name>.kube-system.svc.cluster.local.;
   ```

   Здесь:

   - `<dns_name>` — имя DNS-сервиса Kubernetes, полученное на шаге 3, например, `kube-dns`.

5. Сохраните файл и перезапустите `loki-gateway`:

   ```
   kubectl rollout restart deployment/loki-gateway -n kube-prometheus-stack
   ```

▼ *Особенности обновления системы с версии ниже 2.3.0*

Начиная с версии 2.3.0 система поставляется на базе платформы R-Vision EVO.

Обновление системы с версий ниже 2.3.0 требует ручных действий.

Выключение коллекторов

Перед обновлением системы с версии ниже 2.3.0 необходимо выключить все работающие коллекторы.

После обновления все необходимые коллекторы и конвейеры нужно будет включить вручную.

Создание общей БД PostgreSQL

Сервисы платформы могут использовать как общую БД, так и собственные. В установщиках системы версий ниже 2.3.0 общая БД отсутствует. Необходимо создать ее вручную с помощью команды:

```
kubectl exec -n <namespace> sts/postgresql -- psql -U <username> -d postgres -c
'CREATE DATABASE evo;'
```

Здесь:

- <namespace> — название пространства имен, в котором установлена система.
- <username> — имя пользователя PostgreSQL, которое было указано при установке системы.



Имя пользователя PostgreSQL можно получить из секрета evo.siem.global, выполнив следующую команду:

```
kubectl -n <namespace> get secrets evo.siem.global -o
jsonpath='{.data.POSTGRES_USER}' | base64 -d
```

Здесь:

- <namespace> — название пространства имен, в котором установлена система.

Обновление NATS

Начиная с версии 2.2.0 в поставку системы входит NATS. При обновлении с 2.2.x необходимо предварительно удалить предыдущие helm-релизы NATS, чтобы избежать ошибок. Для этого:

1. Удостоверьтесь, что NATS развернут в необходимом пространстве имен:

```
helm list -n <namespace> | grep nats-
```

2. Удалите nats-main и nats-bridge при их наличии:

```
helm uninstall -n <namespace> nats-bridge
helm uninstall -n <namespace> nats-main
```

Здесь:

- <namespace> — название пространства имен, в котором установлена система.

Устранение проблем при обновлении

При обновлении системы установщик выполняет ряд специфичных действий по переносу конфигурационных параметров в платформу.

Если во время обновления системы произошел сбой, выполните следующие действия:

1. Удалите метаданные модулей, которые успели развернуться до сбоя:

```
kubectl get cm -n <namespace> -o name | grep release-info | xargs -n1 -t kubectl
delete -n <namespace>
```

2. Запустите обновление повторно без распаковки дистрибутива:

```
export PATH=$PATH:/opt/r-vision/common/bin
```

```
evoctl instance update -v
```

Добавление разрешений

При обновлении системы с версии ниже 2.3.0 в пользовательских ролях сбрасываются все разрешения. После обновления необходимо заново настроить разрешения ролей.

▼ *Особенности обновления системы с версии ниже 2.3.3*

Начиная с версии 2.3.3 названия схем активных списков и таблиц обогащения не могут содержать точки.

Перед обновлением системы с версии ниже 2.3.3 удостоверьтесь, что ни одна схема активного списка или таблица обогащения не содержит точки в названии. Если такие схемы или таблицы имеются, необходимо создать их новые версии без точек в названии, а старые версии удалить.

▼ *Особенности обновления системы с версии ниже 2.4.0*

После обновления системы с версии ниже 2.4.0 тип существующих точек входа HTTP Client изменится на HTTP Client_deprecated в связи с их функциональными обновлениями.

Для обновления точек входа HTTP Client:

1. Измените тип существующих точек входа HTTP Client_deprecated в конвейерах на обновленный тип без суффикса _deprecated.
2. Удалите устаревший тип точки входа из таблицы `entry_point_type` в базе данных коллекторов PostgreSQL. Для этого:
   a. Подключитесь к терминалу master-узла кластера.
   b. Подключитесь к PostgreSQL, выполнив следующую команду:

   ```
   kubectl exec -it -n <namespace> postgresql-0 sh
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   c. Подключитесь к базе данных коллекторов:

   ```
   psql -U postgres -d collector
   ```

   d. Удалите из таблицы `entry_point_type` устаревший тип точки входа:

   ```
   delete from "entry_point_type" where "name" like '%_deprecated';
   ```

▼ *Особенности обновления системы с версии ниже 2.5.0*

При обновлении системы с версии ниже 2.5.0 необходимо обновить ClickHouse до поддерживаемой версии.

▼ *Особенности обновления системы с версии ниже 2.6.0*

Обновление системы с версий ниже 2.6.0 требует ручных действий.

Сохранение настроек стримов в менеджере пространств

При обновлении системы с версии ниже 2.6.0 настройки стримов в менеджере пространств автоматически принимают значения по умолчанию.

Чтобы не потерять изменения, внесенные в настройки стримов, **перед обновлением системы** сохраните их в секрет `evo.space.global`. Для этого выполните следующую команду для каждой измененной переменной окружения:

```
kubectl patch secret evo.space.global -n evo-namespace
-p='{"stringData":{"<env_var>": "<value>"}}'
```

Здесь:

- `<env_var>` — переменная окружения, новое значение которой необходимо сохранить. Например: `NATS_CONFIGURATOR_GLOBAL_OBJECTS_OS_MAX_SIZE`.
- `<value>` — новое значение переменной окружения. Например: `512MB`.



Список переменных окружения с настройками стримов и их значения по умолчанию приведены в разделе Настройка стримов в менеджере пространств.

Обновление Microsoft Visual C++

Начиная с версии 2.6.0 агенты в ОС Windows совместимы с Microsoft Visual C++ 14.29 и выше. Если ваши агенты работают на хостах с ОС Windows, при обновлении системы с версии ниже 2.6.0 обновите Microsoft Visual C++ до версии не ниже 14.29.

Переход на новые точки входа R-Vision Endpoint

Начиная с версии 2.6.0 тип точек входа R-Vision Endpoint заменен на R-Vision Endpoint_deprecated в силу функционального устаревания, а R-Vision EVO Endpoint — на R-Vision Endpoint.

После обновления системы с версии ниже 2.6.0 рекомендуется перейти со старых точек входа R-Vision Endpoint_deprecated на новые — R-Vision Endpoint. Для этого:

1. Удалите агенты R-Vision Endpoint и установите системные агенты согласно инструкции в разделе Переход с агентов R-Vision Endpoint на системные агенты.
2. Измените тип существующих точек входа R-Vision Endpoint_deprecated в конвейерах на обновленный тип без суффикса _deprecated и укажите домен для подключения к шлюзу. Подробности приведены в описании точки входа R-Vision Endpoint.
3. Удалите устаревший тип точки входа из таблицы `entry_point_type` в базе данных коллекторов PostgreSQL. Для этого:
   a. Подключитесь к терминалу master-узла кластера.
   b. Подключитесь к PostgreSQL, выполнив следующую команду:

```
kubectl exec -it -n <namespace> postgresql-0 sh
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

c. Подключитесь к базе данных коллекторов:

```
psql -U postgres -d collector
```

d. Удалите из таблицы `entry_point_type` устаревший тип точки входа:

```
delete from "entry_point_type" where id = 'f7183617-e756-4fa9-9f59-
bd2c2529625c';
```

Включение поиска по нескольким хранилищам событий



Одновременный поиск по нескольким хранилищам, созданным до версии 2.4.0 и в версии 2.4.0 или выше, недоступен. Это связано с переходом на тип данных `DateTime64` в служебных полях моделей событий в версии 2.4.0. В таком случае необходимо ограничиться поиском только по тем хранилищам, которые были созданы либо в версии ниже 2.4.0, либо в версии 2.4.0 или выше.

Для работы поиска по нескольким хранилищам событий выполните процедуру создания недостающих удаленных (remote) и локальных таблиц хранилищ:

1. Создайте необходимые распределенные таблицы с помощью скрипта, приведенного после заголовка **Создать необходимые распределенные таблицы** в разделе Создание пользователей ClickHouse. Вы можете выполнить скрипт на любом экземпляре кластера ClickHouse.
2. Подключитесь к терминалу master-узла кластера.
3. Откройте ConfigMap сервиса `evo.streams.event-storage-manager`. Для этого выполните следующую команду:

```
kubectl -n <namespace> edit configmap evo.streams-event-storage-manager-env
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

4. Измените значение переменной `EVO_STREAMS_EVENT_STORAGE_MANAGER_RECREATE_EVENT_STORAGES` на `"true"`.



Пересоздание таблиц хранилищ событий занимает некоторое время. В процессе пересоздания liveness-пробы сервиса `evo.streams.event-storage-manager` могут завершиться с ошибкой, что приведет к циклическому

перезапуску этого сервиса. Во избежание такого поведения рекомендуется увеличить интервал между liveness-пробами, выполнив следующие действия:

1. Откройте конфигурацию контроллера Deployment `evo.streams.event-storage-manager`, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit deployment evo.streams.event-storage-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

2. Скопируйте и сохраните текущие значения параметров в секции `livenessProbe`.

   Пример содержимого секции `livenessProbe`

   ```
   livenessProbe:
     grpc:
       port: 3000
       service: ''
     initialDelaySeconds: 10
     timeoutSeconds: 3
     periodSeconds: 10
     successThreshold: 1
     failureThreshold: 3
   ```

3. Увеличьте значения параметров `timeoutSeconds`, `periodSeconds` и `failureThreshold`.
4. Сохраните изменения.

После перезапуска сервиса `evo.streams.event-storage-manager` рекомендуется восстановить настройки liveness-проб по умолчанию.

5. Удалите под сервиса `evo.streams.event-storage-manager`, выполнив следующую команду:

   ```
   kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.streams.event-storage-manager
   ```

   Под будет пересоздан автоматически.

6. Дождитесь, когда под `evo.streams.event-storage-manager` перейдет в статус `Running`. Инструкция по проверке статуса пода приведена в разделе Просмотр состояния подов кластера.
7. Когда под перейдет в статус `Running`, перейдите в веб-интерфейс системы и убедитесь, что поиск по хранилищам событий выполняется успешно и события поступают в хранилища.

8. Откройте ConfigMap сервиса `evo.streams.event-storage-manager`. Для этого выполните следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.streams-event-storage-manager-env
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

9. Измените значение переменной `EVO_STREAMS_EVENT_STORAGE_MANAGER_RECREATE_EVENT_STORAGES` обратно на `"false"`.
10. Если вы меняли интервал liveness-проб, восстановите интервал по умолчанию в конфигурации `evo.streams.event-storage-manager`. Ее можно открыть, выполнив следующую команду:

    ```
    kubectl -n <namespace> edit deployment evo.streams.event-storage-manager
    ```

    Здесь:

    - `<namespace>` — название пространства имен, в котором установлена система.

Восстановление работы сервисов оповещений

После обновления системы с версии ниже 2.6.0 сервисы оповещений в конвейерах могут перестать генерировать оповещения. Чтобы возобновить их генерацию, отключите и заново установите конфигурации всех конвейеров, которые содержат сервисы оповещений.

▼ *Особенности обновления системы с версии ниже 2.7.0*

При обновлении системы с версии ниже 2.7.0 необходимо обновить ClickHouse до поддерживаемой версии.

▼ *Особенности обновления системы с версии ниже 2.7.1*

Пересоздание системных активных списков

Начиная с версии 2.7.1 в активных списках изменена логика срока жизни записей, а также добавлен специальный тип данных для хранения VRL в записях. Теперь срок жизни записи можно задавать в секундах через поле `ttl` типа `integer` вместо поля `dueDate`. Из-за этого при обновлении системы с версии 2.7.0 системные активные списки могут начать работать некорректно.

Чтобы восстановить корректную работу системных активных списков, **перед обновлением системы с версии 2.7.0** их необходимо удалить. Для этого:

1. Удостоверьтесь, что системные активные списки не используются ни в одном из конвейеров. В противном случае удалите их из конвейеров.
2. Сбросьте тип системных активных списков на **Пользовательский**, запустив в СУБД PostgreSQL следующий скрипт:

   ```
   -- Сброс типа для белого и черного списков.
   UPDATE active_list.active_list
   ```

```
SET is_system = false
WHERE active_list_name IN ('system-whitelist', 'system-blacklist');

-- Сброс типа для схем белого и черного списков.
UPDATE expertise.expertise
SET object_type = 'custom'
WHERE name IN ('system-whitelist', 'system-blacklist');

-- Удаление скрипта добавления исключений.
DELETE FROM evo_streams_script.executed_script WHERE name =
'ScriptAddExceptionActiveList';
```

3. Удалите системные активные списки:
   a. Перейдите в раздел **Ресурсы → Активные списки**.
   b. Найдите активные списки system-whitelist и system-blacklist.
   c. Удостоверьтесь, что найденные активные списки имеют тип **Пользовательский**. В противном случае повторите шаг 2 текущей инструкции.
   d. Удалите активные списки system-whitelist и system-blacklist.
4. Удалите схемы системных активных списков:
   a. Перейдите в раздел **Экспертиза**.
   b. Найдите схемы активных списков system-whitelist и system-blacklist.
   c. Удостоверьтесь, что найденные схемы активных списков имеют тип создания **Пользовательский**. В противном случае повторите шаг 2 текущей инструкции.
   d. Удалите схемы активных списков system-whitelist и system-blacklist.



Если вы хотите импортировать в систему CSV-файлы с записями системных активных списков, которые были экспортированы в версии 2.7.0, их необходимо предварительно отредактировать и привести к актуальному формату. Для этого выполните следующие действия:

a. Замените поле `dueDate` на `ttl` и укажите актуальные сроки жизни для каждой записи в секундах. Поле `ttl` может принимать только целочисленные значения.

   Если требуется сделать записи бессрочными, задайте значение `0`.

b. Удалите поле `createdAt`.

Обновление настроек NATS Object Store

Начиная с версии 2.7.1 в системе исправлена проблема, из-за которой при сбросе метрик в глобальной шине могла очиститься очередь событий.

Теперь в новых коллекторах сброс метрик не будет приводить к очистке очереди в глобальной шине. Однако в существующих коллекторах данная проблема по-прежнему может наблюдаться. Чтобы исправить ее, необходимо изменить настройки объектного хранилища NATS Object Store.

Для этого **после обновления на версию 2.7.1 или выше** выполните следующие действия для центрального кластера и каждого кластера сателлита:

1. Подключитесь к терминалу master-узла кластера.
2. Откройте терминал контейнера пода `nats-space-box`:

   ```
   kubectl exec -n <namespace> -it deployment/nats-space-box -- sh
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

3. Создайте скрипт `script.sh`, например, с помощью команды:

   ```
   vi script.sh
   ```

   Содержимое скрипта `script.sh`

   ```
   filter="<object_id>"

   data="$(nats obj ls -n r-space-siem-collector-objects | grep "$filter")"
   set -- $data

   list="$@"

   changed=1
   while [ $changed -eq 1 ]; do
     changed=0
     new_list=""
     prev=""
     for item in $list; do
       if [ -z "$prev" ]; then
         prev="$item"
         continue
       fi
       prev_prefix="${prev:0:105}"
       curr_prefix="${item:0:105}"

       if [ "$curr_prefix" \< "$prev_prefix" ]; then
         new_list="$new_list $item"
         prev="$prev"
         changed=1
       elif [ "$curr_prefix" == "$prev_prefix" ]; then
         prev_suffix="${prev:105}"
         curr_suffix="${item:105}"
         prev_p2=$(echo "$prev_suffix" | cut -d '_' -f2)
         curr_p2=$(echo "$curr_suffix" | cut -d '_' -f2)
         if [[ "$prev_p2" =~ ^[0-9]+$ && "$curr_p2" =~ ^[0-9]+$ && "$curr_p2" \<
   "$prev_p2" ]]; then
   ```

```
        new_list="$new_list $item"
        prev="$prev"
        changed=1
      else
        new_list="$new_list $prev"
        prev="$item"
      fi
    else
      new_list="$new_list $prev"
      prev="$item"
    fi
  done

  new_list="$new_list $prev"
  list="$new_list"
done


for oldKey in $list; do
  suffix="${oldKey:105}"

  p1=$(echo "$suffix" | cut -d'_' -f1)
  p2=$(echo "$suffix" | cut -d'_' -f2)
  p3=$(echo "$suffix" | cut -d'_' -f3)

  if [[ -z "$p1" && "$p2" =~ ^[0-9]+$ ]]; then
    suffix=""
    if [[ "$p3" == "mongodb/mongodb.checkpoints.json" ]]; then
suffix="/mongodb.checkpoints.json"; fi
    if [[ "$p3" == "sftp/checkpoints.json" ]];            then
suffix="/checkpoints.json"        ; fi
    if [[ "$p3" == "smb/checkpoints.json" ]];             then
suffix="/checkpoints.json"        ; fi
    if [[ "$p3" == "database/checkpoints.json" ]];        then
suffix="/checkpoints.json"        ; fi
    if [[ "$p3" == "http" ]];                             then
suffix="/http_checkpoints.json"   ; fi

    if [[ -n "$suffix" ]]; then
      prefix="${oldKey:0:105}"
      newKey=$(echo "$prefix""$suffix")
      oldConnectionId=$(echo "$oldKey" | cut -d'/' -f2)
      newConnectionId=$(echo "$newKey" | cut -d'/' -f2)

      echo ">>>>> rename object key in bucket r-space-siem-collector-objects"
      echo "  old key: $oldKey"
      echo "  new key: $newKey"
      echo "  get obj: $(nats object get r-space-siem-collector-objects "$oldKey"
-f -O ./tmp.json)"

      content=$(cat tmp.json)
```

```
      contentIndex=$((${#oldConnectionId}+18))
      contentSuffix="${content:$contentIndex}"
      newContent="{\"connection_id\":\"""$newConnectionId""$contentSuffix"
      echo "$newContent" > ./tmp_new.json

      echo "  put obj: $(nats object put r-space-siem-collector-objects --name
"$newKey" -f ./tmp_new.json)"
      echo "  del obj: $(nats object del r-space-siem-collector-objects "$oldKey"
-f)"
    fi
  fi
done
```

Здесь:

- `<object_id>` — идентификатор коллектора или конвейера, для которого необходимо изменить настройки NATS Object Store.

  При необходимости вы можете указать несколько идентификаторов, разделяя их вертикальной чертой. Например: `7f901c19-451f-698e-a5f4-6ea53f074bb6|e7bebc60-41cf-8bf3-68b1-382332a74f57`.

  Вы также можете не указывать идентификаторы, оставив значение пустым. Тогда скрипт отработает для всех коллекторов и конвейеров в кластере.

4. Обновите настройки NATS Object Store. Для этого выполните следующие действия для каждого коллектора в кластере:

   a. Отключите конфигурации всех конвейеров в коллекторе.

   b. Выключите коллектор.

   c. Отредактируйте скрипт `script.sh`, добавленный ранее, указав идентификатор коллектора вместо `<object_id>`.

   d. Запустите скрипт, например, с помощью команды:

   ```
   sh script.sh
   ```

   Для коллектора будут обновлены настройки NATS Object Store.

   e. Включите коллектор и установите конфигурации его конвейеров.



Вы также можете обновить настройки сразу всех коллекторов в кластере. Для этого:

a. Отключите конфигурации всех конвейеров во всех коллекторах кластера.

b. Выключите все коллекторы в кластере.

c. Укажите пустое значение вместо `<object_id>` в скрипте `script.sh`.

d. Запустите скрипт, например, с помощью команды:

```
sh script.sh
```

e. Включите нужные коллекторы и установите конфигурации их конвейеров.

Однако данный вариант не рекомендуется, поскольку при отключении всех коллекторов может произойти большая пауза в сборе событий.

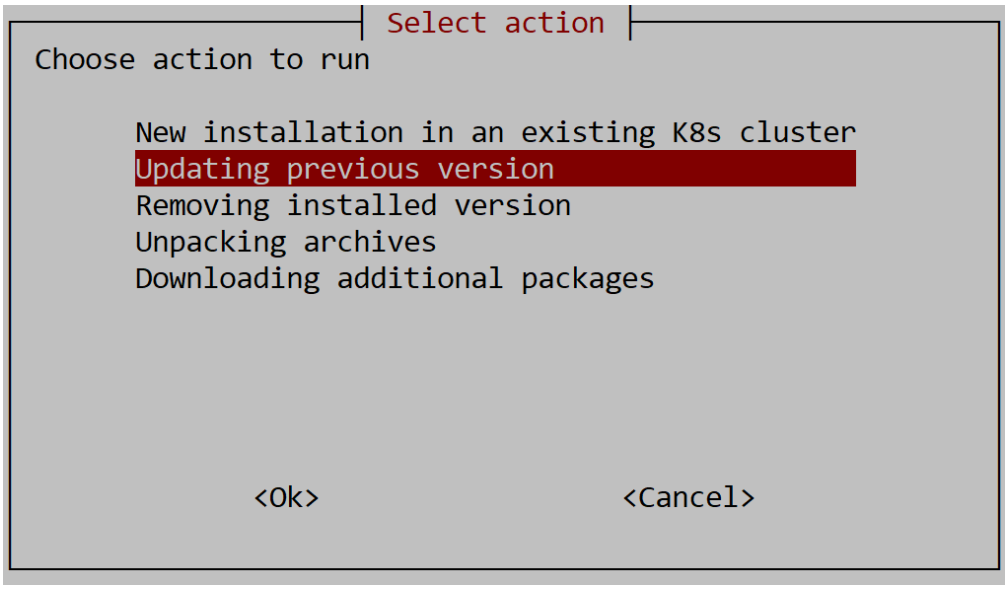


После выполнения действий временный скрипт `script.sh` можно не удалять вручную, поскольку он будет удален автоматически при перезапуске пода.

Обновление системы

Чтобы обновить экземпляр системы до новой версии:

1. Запустите установщик и выберите режим установки **Updating previous version**.



При выборе обновления выполняется распаковка пакетов в каталог продукта и управление передается скрипту установщика.



Если версии ClickHouse, PostgreSQL и/или NATS, установленные в кластере, не соответствуют техническим требованиям, при запуске обновления системы установщик попытается обновить их автоматически.

Если версии компонентов значительно устарели, автоматическое обновление будет недоступно и отобразится сообщение с ключевыми словами `Please perform a manual update`. Пример для PostgreSQL:

```
Please perform a manual update postgresql to required version >=
15.0.0.
```

В таком случае выполните ручное обновление необходимых компонентов согласно инструкциям из раздела Ручное обновление компонентов системы и

перезапустите обновление системы.

Автоматическое обновление доступно для ClickHouse и PostgreSQL, только если они установлены **в кластере Kubernetes**. Обновление **вынесенных** инсталляций ClickHouse и PostgreSQL администратор кластера должен выполнять самостоятельно.

2. Выберите пространство имен, в котором установлен экземпляр системы.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

3. Задайте параметры Ansible:
   a. Имя пользователя.
   b. Метод аутентификации: через ключ или по паролю.
   c. Пароль: задайте и подтвердите пароль пользователя.

В целевом пространстве имен будут обновлены все чарты модуля Streams.

После обновления на экран будет выведен URL-адрес для подключения к системе.

Обновление сателлитов

После обновления основного экземпляра системы необходимо обновить все подключенные к нему сателлиты. Чтобы обновить сателлиты, выполните следующие действия в каждом из них:

1. Скопируйте архив TLS-сертификатов `$SETUP_ROOT_DIR/r-vision/nats-tls.tar.gz` из основного экземпляра системы в каталог `$SETUP_ROOT_DIR/r-vision` сателлита.



Каталоги установки (значения переменной `$SETUP_ROOT_DIR`) основного экземпляра системы и сателлита могут отличаться.

2. Обновите сателлит с помощью установщика аналогично основному экземпляру системы.

Действия после обновления

После обновления выполните следующие действия:

- Создание недостающих пользователей и таблиц ClickHouse
- Перезапуск коллекторов и конвейеров

Создание недостающих пользователей и таблиц ClickHouse

После обновления системы необходимо создать необходимых пользователей и недостающие таблицы в ClickHouse, если они отсутствуют. Для этого выполните следующий скрипт на любом экземпляре кластера ClickHouse:

```
-- Создать пользователя owner и выдать ему необходимые права.
CREATE USER IF NOT EXISTS owner ON CLUSTER '{cluster}' IDENTIFIED WITH sha256_password
```

```
BY '<OwnerPassword>';
GRANT ON CLUSTER '{cluster}' ALL on default.* TO 'owner';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.asynchronous_metrics TO 'owner';

-- Создать пользователя writer и выдать ему необходимые права.
CREATE USER IF NOT EXISTS writer ON CLUSTER '{cluster}' IDENTIFIED WITH
sha256_password BY '<WriterPassword>';
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' INSERT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER UPDATE ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER DELETE ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.asynchronous_metrics TO 'writer';

-- Создать пользователя reader и выдать ему необходимые права.
CREATE USER IF NOT EXISTS reader ON CLUSTER '{cluster}' IDENTIFIED WITH
sha256_password BY '<ReaderPassword>';
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'reader';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.asynchronous_metrics TO 'reader';

-- Создать необходимые распределенные таблицы.
CREATE TABLE IF NOT EXISTS default.disks ON CLUSTER '{cluster}' AS system.disks ENGINE
```

```
= Distributed('{cluster}', system, disks);
CREATE TABLE IF NOT EXISTS default.storage_policies ON CLUSTER '{cluster}' AS
system.storage_policies ENGINE = Distributed('{cluster}', system, storage_policies);
CREATE TABLE IF NOT EXISTS default.parts ON CLUSTER '{cluster}' AS system.parts ENGINE
= Distributed('{cluster}', system, parts);
CREATE TABLE IF NOT EXISTS default.query_log ON CLUSTER '{cluster}' AS
system.query_log ENGINE = Distributed('{cluster}', system, query_log);
CREATE TABLE IF NOT EXISTS default.columns ON CLUSTER '{cluster}' AS system.columns
ENGINE = Distributed('{cluster}', system, columns);
CREATE TABLE IF NOT EXISTS default.parts_columns ON CLUSTER '{cluster}' AS
system.parts_columns ENGINE = Distributed('{cluster}', system, parts_columns);
CREATE TABLE IF NOT EXISTS default.tables ON CLUSTER '{cluster}' AS system.tables
ENGINE = Distributed('{cluster}', system, tables);
CREATE TABLE IF NOT EXISTS default.clusters ON CLUSTER '{cluster}' AS system.clusters
ENGINE = Distributed('{cluster}', system, clusters);
CREATE TABLE IF NOT EXISTS default.asynchronous_metrics ON CLUSTER '{cluster}' AS
system.asynchronous_metrics ENGINE = Distributed('{cluster}', system,
asynchronous_metrics);
```

Здесь `<OwnerPassword>`, `<WriterPassword>` и `<ReaderPassword>` — это пароли пользователей owner, writer и reader соответственно.



Необходимо указывать пароли, которые были использованы на этапах настройки общесистемного экземпляра ClickHouse и ClickHouse для событий ИБ при установке системы.

Перезапуск коллекторов и конвейеров

После обновления рекомендуется выключить все работающие коллекторы и конвейеры и включить их заново, чтобы обновить образы коллекторов и конфигурации конвейеров.

Авторизация в системе

Чтобы выполнить вход в систему:

1. В адресной строке браузера введите URL-адрес, выведенный на экран после обновления системы. На экране отобразится страница авторизации.
2. Укажите логин и пароль учетной записи пользователя и нажмите на кнопку **Войти**.



Если вы обновляли сателлиты, после обновления может потребоваться перезапуск выполняющихся на них коллекторов и сервисов из интерфейса основного экземпляра системы.

Ручное обновление компонентов системы

В данном разделе приведены инструкции по ручному обновлению инфраструктурных компонентов системы: СУБД ClickHouse и PostgreSQL и брокера сообщений NATS.



Если версии компонентов значительно устарели, при запуске обновления системы

отобразится сообщение с ключевыми словами `Please perform a manual update`. Пример для PostgreSQL:

```
Please perform a manual update postgresql to required version >=
15.0.0.
```

Содержание раздела:

- Обновление ClickHouse
- Обновление PostgreSQL
- Обновление NATS

Обновление ClickHouse

Порядок ручного обновления ClickHouse различается в зависимости от расположения инсталляции ClickHouse.

Вынесенная инсталляция ClickHouse

Действия выполняются на каждом узле кластера ClickHouse.

Чтобы обновить ClickHouse в выделенном кластере:

1. Скачайте RPM-пакеты clickhouse-client, clickhouse-common-static и clickhouse-server нужной версии из официального репозитория ClickHouse.
2. Запустите обновление:

   ```
   dnf install clickhouse-client-<package_version>.rpm clickhouse-common-
   static-<package_version>.rpm clickhouse-server-<package_version>.rpm
   ```

   Здесь:

   - `<package_version>` — версия скачанных пакетов, например, `25.8.1.5101`.
3. Перезапустите сервис ClickHouse:

   ```
   systemctl restart clickhouse-server
   ```

После обновления кластера ClickHouse системным пользователям owner, writer и reader необходимо заново выдать требуемые права с помощью скрипта, приведенного ниже. Выполните скрипт на любом узле кластера ClickHouse:

```
-- Выдать необходимые права пользователю owner.
GRANT ON CLUSTER '{cluster}' ALL on default.* TO 'owner';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'owner';
```

```
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies
TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO
'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO
'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO
'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO
'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON
system.asynchronous_metrics TO 'owner';

-- Выдать необходимые права пользователю writer.
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' INSERT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER UPDATE ON default.* TO
'writer';
GRANT ON CLUSTER '{cluster}' ALTER DELETE ON default.* TO
'writer';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies
TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO
'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO
'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO
'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO
'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO
'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON
system.asynchronous_metrics TO 'writer';

-- Выдать необходимые права пользователю reader.
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'reader';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies
TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'reader';
```

```
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO
'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO
'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO
'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.tables TO
'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.clusters TO
'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON
system.asynchronous_metrics TO 'reader';
```

ClickHouse в кластере Kubernetes

Чтобы обновить ClickHouse в кластере Kubernetes:

1. Скачайте архив установщика новой версии системы, если еще не сделали этого.
2. Загрузите образ новой версии ClickHouse на узлы кластера, выполнив следующую команду на узле, где находится установщик:

   ```
   evoctl instance upload -m evo.infra -g Clickhouse
   ```

3. Перейдите в каталог с helm-чартом ClickHouse, выполнив следующую команду:

   ```
   cd /opt/r-vision/infra/helm/clickhouse
   ```

4. Получите параметры текущей установленной версии ClickHouse, выполнив следующие команды:

   ```
   NS=<namespace>
   ADMIN_PASSWORD=$(kubectl get secret -n "$NS" clickhouse -o
   jsonpath="{.data.admin-password}" | base64 -d)
   SYNC_PASSWORD=$(helm get values -n "$NS" clickhouse -o json | jq -r
   '.clusterSettings.password // ""')
   PVC_SIZE=$(helm get values -n "$NS" clickhouse -o json | jq -r
   '.persistence.size // ""')
   REPLICAS=$(helm get values -n "$NS" clickhouse -o json | jq -r
   '.replicaCount // ""')
   SHARDS=$(helm get values -n "$NS" clickhouse -o json | jq -r '.shards //
   ""')
   ```

   Здесь:

   - <namespace> — название пространства имен, в котором установлена система.

     Вы можете получить название пространства имен, в котором

установлена система, с помощью утилиты evoctl.

5. Проверьте возможность запуска обновления с помощью опции `--dry-run`.



Команда ниже только проверяет возможность запуска обновления, но не выполняет само обновление.

```
helm upgrade -n "$NS" clickhouse ./ -f ./custom_values.yaml \
    --set auth.password="${ADMIN_PASSWORD:?}" \
    --set shards="${SHARDS:?}" \
    --set replicaCount="${REPLICAS:?}" \
    --set persistence.size="${PVC_SIZE:?}" \
    --set clusterSettings.password="${SYNC_PASSWORD:?}" \
    --set resources.requests.cpu=<resources_requests_cpu> \
    --set resources.requests.memory=<resources_requests_memory> \
    --set resources.limits.cpu=<resources_limits_cpu> \
    --set resources.limits.memory=<resources_limits_memory> \
    --dry-run
```

Здесь и далее:

- `<resources_requests_cpu>` — минимальные требуемые ресурсы процессора. По умолчанию `300m` (30% времени одного ядра).
- `<resources_requests_memory>` — минимальный требуемый объем ОЗУ. По умолчанию `1200Mi` (1200 МиБ).
- `<resources_limits_cpu>` — максимальное допустимое потребление ресурсов процессора. По умолчанию `300m` (30% времени одного ядра).
- `<resources_limits_memory>` — максимальное допустимое потребление ОЗУ. По умолчанию `1200Mi` (1200 МиБ).

На следующий шаг можно переходить, только если при выполнении команды выше не возникло ошибок.

Могут возникнуть ошибки следующего вида:

```
bash: <env_name>: parameter null or not set
```



Здесь:

- `<env_name>` — имя переменной окружения, значение которой получить не удалось: например, `ADMIN_PASSWORD`.

Если ошибка возникла, выполните следующие действия для каждой из переменных с ошибкой. Ниже приведен пример действий для переменной `ADMIN_PASSWORD`:

a. Выполните команду для получения значения переменной вручную:

```
kubectl get secret -n "$NS" clickhouse -o
jsonpath="{.data.admin-password}" | base64 -d
```

b. Присвойте переменной результат выполнения предыдущей команды:

```
ADMIN_PASSWORD=<admin_password>
```

Здесь:

- `<admin_password>` — пароль администратора ClickHouse, полученный на предыдущем шаге.

c. Получив значения всех переменных, повторите команду проверки возможности обновления:

```
helm upgrade -n "$NS" clickhouse ./ -f
./custom_values.yaml \
    --set auth.password="${ADMIN_PASSWORD:?}" \
    --set shards="${SHARDS:?}" \
    --set replicaCount="${REPLICAS:?}" \
    --set persistence.size="${PVC_SIZE:?}" \
    --set clusterSettings.password="${SYNC_PASSWORD:?}" \
    --set resources.requests.cpu=<resources_requests_cpu>
\
    --set
resources.requests.memory=<resources_requests_memory> \
    --set resources.limits.cpu=<resources_limits_cpu> \
    --set
resources.limits.memory=<resources_limits_memory> \
    --dry-run
```

d. Если повторная проверка возможности обновления не выявила ошибок, переходите к следующему шагу. В противном случае обратитесь в службу поддержки по адресу support@rvision.ru.

6. Запустите обновление ClickHouse:

```
helm upgrade -n "$NS" clickhouse ./ -f ./custom_values.yaml \
    --set auth.password="${ADMIN_PASSWORD:?}" \
    --set shards="${SHARDS:?}" \
    --set replicaCount="${REPLICAS:?}" \
    --set persistence.size="${PVC_SIZE:?}" \
    --set clusterSettings.password="${SYNC_PASSWORD:?}" \
    --set resources.requests.cpu=<resources_requests_cpu> \
    --set resources.requests.memory=<resources_requests_memory> \
    --set resources.limits.cpu=<resources_limits_cpu> \
```

```
    --set resources.limits.memory=<resources_limits_memory>
```

7. Убедитесь, что обновление выполнилось успешно. Для этого запустите мониторинг подов ClickHouse, выполнив следующую команду:

   ```
   kubectl get pod -n "$NS" -l app.kubernetes.io/instance=clickhouse --watch
   ```

   После выполнения команды состояние каждого пода начнет выводиться в консоль в реальном времени. При успешном обновлении все поды будут иметь статус `Running`.

   Чтобы прекратить наблюдение за состоянием подов, нажмите на сочетание клавиш CTRL+C.

Обновление PostgreSQL

Порядок ручного обновления PostgreSQL различается в зависимости от расположения инсталляции PostgreSQL.

PostgreSQL в кластере Kubernetes



При обновлении PostgreSQL до новой мажорной версии (например, с 14 до 15) требуется **перенос данных между версиями**: создание полного дампа всех баз данных в предыдущей версии PostgreSQL и восстановление из него в новой версии.

Следуйте приведенным ниже инструкциям очень внимательно, поскольку **ошибочные действия могут привести к потере данных**.

1. Выведите подробный список всех баз данных, выполнив следующие команды:

   ```
   NS=<namespace>
   kubectl exec -n "$NS" sts/postgresql -- psql -U postgres -c '\l+'
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

     Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

2. Сохраните отобразившийся список. Он потребуется вам для контроля восстановления из дампа в конце процедуры обновления.
3. Оцените суммарный объем дискового пространства, занимаемый всеми базами данных PostgreSQL, выполнив следующую команду:

   ```
   kubectl exec -n "$NS" sts/postgresql -- psql -U postgres -c 'SELECT
   ```

```
pg_size_pretty(SUM(pg_database_size(pg_database.datname))) AS "TotalSize"
FROM pg_database;'
```

4. Создайте каталог для дампа на диске, на котором свободного пространства не меньше, чем размер баз данных, оцененный на предыдущем шаге:

   ```
   DUMP_DIR=<path_to_dump_dir>
   mkdir -pv "$DUMP_DIR"
   ```

   Здесь:

   - `<path_to_dump_dir>` — путь к каталогу для дампа.

5. Создайте дамп, выполнив следующую команду:

   ```
   kubectl exec -n "$NS" sts/postgresql -- pg_dumpall -U postgres --clean --if
   -exists --load-via-partition-root --quote-all-identifiers --no-password >
   "$DUMP_DIR"/fulldump.pgdump
   ```

6. Выполните базовую проверку корректности дампа.

   Если хотя бы одна из нижеследующих проверок не будет пройдена, это значит, что дамп некорректен. Убедитесь, что в каталоге для дампа достаточно свободного места, и создайте дамп заново.

   Критерием полной корректности дампа является только успешное восстановление из него.

   a. Убедитесь, что размер резервной копии составляет не менее 200 КБ, выполнив следующую команду:

      ```
      du -hs "$DUMP_DIR"/fulldump.pgdump
      ```

   b. Проверьте заголовок файла дампа, выполнив следующую команду:

      ```
      head "$DUMP_DIR"/fulldump.pgdump
      ```

      Заголовок файла дампа должен содержать следующую строку:

      ```
      PostgreSQL database cluster dump
      ```

   c. Проверьте конец файла дампа, выполнив следующую команду:

      ```
      tail "$DUMP_DIR"/fulldump.pgdump
      ```

Конец файла дампа должен содержать следующие строки:

```
PostgreSQL database dump complete
PostgreSQL database cluster dump complete
```

7. Скачайте архив установщика новой версии системы, если еще не сделали этого.
8. Загрузите образ новой версии PostgreSQL на узлы кластера, выполнив следующую команду на узле, где находится установщик:

```
evoctl instance upload -m evo.infra -g PostgreSQL
```

9. Перейдите в каталог с helm-чартом PostgreSQL, выполнив следующую команду:

```
cd /opt/r-vision/infra/helm/postgresql
```

10. Получите параметры текущей установленной версии PostgreSQL, выполнив следующие команды:

```
POSTGRES_USER=$(helm get values -n "$NS" postgresql -o json | jq -r
'.global.postgresql.auth.username // ""')
POSTGRES_PASSWORD=$(helm get values -n "$NS" postgresql -o json | jq -r
'.global.postgresql.auth.password // ""')

# Имя базы данных может быть определено либо в Helm Values, либо в секрете
Kubernetes.
# Если оно не определено ни в одном из этих элементов конфигурации,
используется имя базы данных по умолчанию "evo".
POSTGRES_DBNAME=$(helm get values -n "$NS" postgresql -o json | jq -r
'.global.postgresql.auth.database // ""')
[ "$POSTGRES_DBNAME" ] || POSTGRES_DBNAME=$(kubectl get secret -n "$NS"
evo.infra.global -o jsonpath="{.data.POSTGRES_DBNAME}" | base64 -d)
[ "$POSTGRES_DBNAME" ] || POSTGRES_DBNAME=evo

# Проверяется минимальный размер PVC для PostgreSQL. Если он составлял менее
8 ГиБ, при обновлении он устанавливается в 8 ГиБ.
PVC_SIZE=$(kubectl get pvc -n "$NS" -l app.kubernetes.io/instance=postgresql
-o jsonpath='{.items[0].spec.resources.requests.storage}')
[ "${PVC_SIZE%Gi}" -ge 8 ] || PVC_SIZE="8Gi"
```

11. Проверьте возможность запуска обновления с помощью опции `--dry-run`.



Команда ниже только проверяет возможность запуска обновления, но не выполняет само обновление.

```
helm upgrade -i -n "$NS" postgresql ./ -f ./custom_values.yaml \
```

```
  --set primary.persistence.size="${PVC_SIZE:?}" \
  --set global.postgresql.auth.username="${POSTGRES_USER:?}" \
  --set global.postgresql.auth.database="${POSTGRES_DBNAME:?}" \
  --set-literal global.postgresql.auth.password="${POSTGRES_PASSWORD:?}" \
  --set-literal
global.postgresql.auth.postgresPassword="${POSTGRES_PASSWORD:?}" \
  --set primary.resources.requests.cpu=<primary_resources_requests_cpu> \
  --set
primary.resources.requests.memory=<primary_resources_requests_memory> \
  --set primary.resources.limits.cpu=<primary_resources_limits_cpu> \
  --set primary.resources.limits.memory=<primary_resources_limits_memory> \
  --dry-run
```

Здесь и далее:

- `<primary_resources_requests_cpu>` — минимальные требуемые ресурсы процессора. По умолчанию `100m` (10% времени одного ядра).
- `<primary_resources_requests_memory>` — минимальный требуемый объем ОЗУ. По умолчанию `128Mi` (128 МиБ).
- `<primary_resources_limits_cpu>` — максимальное допустимое потребление ресурсов процессора. По умолчанию `1` (100% времени одного ядра).
- `<primary_resources_limits_memory>` — максимальное допустимое потребление ОЗУ. По умолчанию `1024Mi` (1024 МиБ).



На следующий шаг можно переходить, только если при выполнении команды выше не возникло ошибок.

Могут возникнуть ошибки следующего вида:

```
bash: <env_name>: parameter null or not set
```

Здесь:

- `<env_name>` — имя переменной окружения, значение которой получить не удалось: например, `POSTGRES_USER`.

Если ошибка возникла, выполните следующие действия для каждой из переменных с ошибкой. Ниже приведен пример действий для переменной `POSTGRES_USER`:

a. Выполните команду для получения значения переменной вручную:

```
helm get values -n "$NS" postgresql -o json | jq -r
'.global.postgresql.auth.username // ""'
```

b. Присвойте переменной результат выполнения предыдущей команды:

```
POSTGRES_USER=<postgres_user>
```

Здесь:

- `<postgres_user>` — имя системного пользователя PostgreSQL, полученное на предыдущем шаге.

c. Получив значения всех переменных, повторите команду проверки возможности обновления:

```
helm upgrade -i -n "$NS" postgresql ./ -f
./custom_values.yaml \
  --set primary.persistence.size="${PVC_SIZE:?}" \
  --set
global.postgresql.auth.username="${POSTGRES_USER:?}" \
  --set
global.postgresql.auth.database="${POSTGRES_DBNAME:?}" \
  --set-literal
global.postgresql.auth.password="${POSTGRES_PASSWORD:?}"
\
  --set-literal
global.postgresql.auth.postgresPassword="${POSTGRES_PASSW
ORD:?}" \
  --set
primary.resources.requests.cpu=<primary_resources_request
s_cpu> \
  --set
primary.resources.requests.memory=<primary_resources_requ
ests_memory> \
  --set
primary.resources.limits.cpu=<primary_resources_limits_cp
u> \
  --set
primary.resources.limits.memory=<primary_resources_limits
_memory> \
  --dry-run
```

d. Если повторная проверка возможности обновления не выявила ошибок, переходите к следующему шагу. В противном случае обратитесь в службу поддержки по адресу support@rvision.ru.

12. Удалите предыдущий установленный релиз PostgreSQL и его ресурсы PVC:

```
helm uninstall -n "$NS" postgresql --wait --debug
kubectl delete pvc -n "$NS" -l app.kubernetes.io/instance=postgresql --wait
```

13. Установите новую версию PostgreSQL:

```
helm upgrade -i -n "$NS" postgresql ./ -f ./custom_values.yaml \
  --set primary.persistence.size="${PVC_SIZE:?}" \
  --set global.postgresql.auth.username="${POSTGRES_USER:?}" \
  --set global.postgresql.auth.database="${POSTGRES_DBNAME:?}" \
  --set-literal global.postgresql.auth.password="${POSTGRES_PASSWORD:?}" \
  --set-literal
global.postgresql.auth.postgresPassword="${POSTGRES_PASSWORD:?}" \
  --set primary.resources.requests.cpu=<primary_resources_requests_cpu> \
  --set
primary.resources.requests.memory=<primary_resources_requests_memory> \
  --set primary.resources.limits.cpu=<primary_resources_limits_cpu> \
  --set primary.resources.limits.memory=<primary_resources_limits_memory>
```

14. Убедитесь, что обновление выполнилось успешно. Для этого запустите мониторинг подов PostgreSQL, выполнив следующую команду:

    ```
    kubectl get pod -n "$NS" -l app.kubernetes.io/instance=postgresql --watch
    ```

    После выполнения команды состояние каждого пода начнет выводиться в консоль в реальном времени. При успешном обновлении все поды будут иметь статус `Running`.

    Чтобы прекратить наблюдение за состоянием подов, нажмите на сочетание клавиш CTRL+C.

15. Восстановите данные из дампа, выполнив следующую команду:

    ```
    kubectl exec -n "$NS" -i sts/postgresql -- psql -U postgres -f - <
    "$DUMP_DIR"/fulldump.pgdump
    ```

16. Выведите подробный список всех баз данных, выполнив следующую команду:

    ```
    kubectl exec -n "$NS" sts/postgresql -- psql -U postgres -c '\l+'
    ```

    Убедитесь, что список соответствует списку баз данных до обновления, полученному на шагах 1—2.

Вынесенная инсталляция PostgreSQL

Для обновления вынесенной инсталляции PostgreSQL воспользуйтесь руководством в официальной документации PostgreSQL.

Обновление NATS



В системе установлены несколько экземпляров NATS, с каждым из которых связан отдельный helm-чарт:

- **nats-bridge**;
- **nats-main**;
- **nats-space**.

Каждый из этих чартов необходимо обновить отдельно.

Чтобы обновить брокер сообщений NATS, выполните следующие действия:

1. Скачайте архив установщика новой версии системы, если еще не сделали этого.
2. Загрузите образ новой версии NATS на узлы кластера, выполнив следующую команду на узле, где находится установщик:

```
evoctl instance upload -m evo.infra -g Nats
```

3. Создайте переменную для названия пространства имен, в котором установлена система, если ранее не сделали этого:

```
NS=<namespace>
```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

4. Обновите чарты экземпляров NATS, для которых это требуется:
   - Обновление чарта nats-bridge
   - Обновление чарта nats-main
   - Обновление чарта nats-space

Обновление чарта nats-bridge

Чтобы обновить чарт `nats-bridge`, выполните следующие действия:

1. Перейдите в каталог с чартом `nats-bridge`, выполнив следующую команду:

```
cd /opt/r-vision/infra/helm/nats-bridge
```

2. Проверьте возможность запуска обновления с помощью опции `--dry-run`.



Команда ниже только проверяет возможность запуска обновления, но не выполняет само обновление.

```
helm upgrade -i -n "$NS" nats-bridge ./ -f ./custom_values.yaml \
  --set container.merge.resources.requests.cpu=<bridge_resources_requests_cpu> \
  --set
container.merge.resources.requests.memory=<bridge_resources_requests_memory> \
```

```
  --set container.merge.resources.limits.cpu=<bridge_resources_limits_cpu> \
  --set container.merge.resources.limits.memory=<bridge_resources_limits_memory> \
  --set container.env.GOMEMLIMIT=<bridge_gomemlimit> \
  --dry-run
```

Здесь и далее:

- `<bridge_resources_requests_cpu>` — минимальные требуемые ресурсы процессора для `nats-bridge`. По умолчанию `250m` (25% времени одного ядра).
- `<bridge_resources_requests_memory>` — минимальный требуемый объем ОЗУ для `nats-bridge`. По умолчанию `128Mi` (128 МиБ).
- `<bridge_resources_limits_cpu>` — максимальное допустимое потребление ресурсов процессора для `nats-bridge`. По умолчанию `4` (100% времени 4 ядер).
- `<bridge_resources_limits_memory>` — максимальное допустимое потребление ОЗУ для `nats-bridge`. По умолчанию `8192Mi` (8192 МиБ).
- `<bridge_gomemlimit>` — переменная окружения Go, задающая лимит памяти для `nats-bridge`. По умолчанию `6144MiB` (6144 МиБ).



На следующий шаг можно переходить, только если при выполнении команды выше не возникло ошибок. В противном случае обратитесь в службу поддержки по адресу support@rvision.ru.

3. Установите новую версию NATS для `nats-bridge`:

```
helm upgrade -i -n "$NS" nats-bridge ./ -f ./custom_values.yaml \
  --set container.merge.resources.requests.cpu=<bridge_resources_requests_cpu> \
  --set
container.merge.resources.requests.memory=<bridge_resources_requests_memory> \
  --set container.merge.resources.limits.cpu=<bridge_resources_limits_cpu> \
  --set container.merge.resources.limits.memory=<bridge_resources_limits_memory> \
  --set container.env.GOMEMLIMIT=<bridge_gomemlimit>
```



Если при обновлении произойдет ошибка, связанная с изменением структуры чарта, удалите предыдущий релиз чарта:

```
helm uninstall <namespace> nats-bridge
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Затем запустите установку новой версии NATS повторно.

4. Убедитесь, что обновление выполнилось успешно. Для этого запустите мониторинг подов `nats-bridge`, выполнив следующую команду:

```
kubectl get pod -n "$NS" -l app.kubernetes.io/instance=nats-bridge --watch
```

После выполнения команды состояние каждого пода начнет выводиться в консоль в реальном времени. При успешном обновлении все поды будут иметь статус `Running`.

Чтобы прекратить наблюдение за состоянием подов, нажмите на сочетание клавиш CTRL+C.

Обновление чарта nats-main

Чтобы обновить чарт `nats-main`, выполните следующие действия:

1. Перейдите в каталог с чартом `nats-main`, выполнив следующую команду:

```
cd /opt/r-vision/infra/helm/nats-main
```

2. Проверьте, что размер PVC для `nats-main` составляет не менее 60 ГиБ, выполнив следующие команды:

```
PVC_SIZE_OLD=$(helm get values -n "$NS" nats-main -o json | jq -r
'.config.jetstream.fileStore.pvc.size // ""')

PVC_SIZE=$(kubectl get pvc -n "$NS" -l app.kubernetes.io/instance=nats-main -o
jsonpath='{.items[0].spec.resources.requests.storage}')

[ "${PVC_SIZE%Gi}" -ge 60 ] || PVC_SIZE="60Gi"

echo "PVC_SIZE_OLD $PVC_SIZE_OLD"
echo "PVC_SIZE     $PVC_SIZE"
```

3. Если значения переменных `PVC_SIZE_OLD` и `PVC_SIZE` различаются, это означает, что размер PVC для `nats-main` — менее 60 ГиБ и его необходимо увеличить для корректной работы системы. Для этого выполните следующие команды:

```
[ "${PVC_SIZE_OLD:?}" = "${PVC_SIZE:?}" ] || {
  for _pvc in $(kubectl get pvc -n "$NS"  -l app.kubernetes.io/instance=nats-main
-o name); do
    kubectl patch -n "$NS" "$_pvc" -p
"{\"spec\":{\"resources\":{\"requests\":{\"storage\":\"$PVC_SIZE\"}}}}"
  done
  kubectl patch secret -n "$NS" evo.infra.global
-p="{\"stringData\":{\"NATS_STORAGE_SIZE\":\"$PVC_SIZE\"}}"
  helm uninstall -n "$NS" nats-main
}
```

4. Проверьте возможность запуска обновления с помощью опции `--dry-run`.



Команда ниже только проверяет возможность запуска обновления, но не выполняет само обновление.

```
helm upgrade -i -n "$NS" nats-main ./ -f ./custom_values.yaml \
  --set container.merge.resources.requests.cpu=<main_resources_requests_cpu> \
  --set container.merge.resources.requests.memory=<main_resources_requests_memory>
\
  --set container.merge.resources.limits.cpu=<main_resources_limits_cpu> \
  --set container.merge.resources.limits.memory=<main_resources_limits_memory> \
  --set container.env.GOMEMLIMIT=<main_gomemlimit> \
  --dry-run
```

Здесь и далее:

- `<main_resources_requests_cpu>` — минимально требуемые ресурсы процессора для `nats-main`. По умолчанию `500m` (50% времени одного ядра).
- `<main_resources_requests_memory>` — минимально требуемый объем ОЗУ для `nats-main`. По умолчанию `1024Mi` (1024 МиБ).
- `<main_resources_limits_cpu>` — максимально допустимое потребление ресурсов процессора для `nats-main`. По умолчанию `4` (100% времени 4 ядер).
- `<main_resources_limits_memory>` — максимально допустимое потребление ОЗУ для `nats-main`. По умолчанию `8192Mi` (8192 МиБ).
- `<main_gomemlimit>` — переменная окружения Go, задающая лимит памяти для `nats-main`. По умолчанию `6144MiB` (6144 МиБ).



На следующий шаг можно переходить, только если при выполнении команды выше не возникло ошибок. В противном случае обратитесь в службу поддержки по адресу support@rvision.ru.

5. Установите новую версию NATS для `nats-main`:

```
helm upgrade -i -n "$NS" nats-main ./ -f ./custom_values.yaml \
  --set container.merge.resources.requests.cpu=<main_resources_requests_cpu> \
  --set container.merge.resources.requests.memory=<main_resources_requests_memory>
\
  --set container.merge.resources.limits.cpu=<main_resources_limits_cpu> \
  --set container.merge.resources.limits.memory=<main_resources_limits_memory> \
  --set container.env.GOMEMLIMIT=<main_gomemlimit>
```



Если при обновлении произойдет ошибка, связанная с изменением структуры чарта, удалите предыдущий релиз чарта:

```
helm uninstall <namespace> nats-main
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Затем запустите установку новой версии NATS повторно.

6. Убедитесь, что обновление выполнилось успешно. Для этого запустите мониторинг подов `nats-main`, выполнив следующую команду:

```
kubectl get pod -n "$NS" -l app.kubernetes.io/instance=nats-main --watch
```

После выполнения команды состояние каждого пода начнет выводиться в консоль в реальном времени. При успешном обновлении все поды будут иметь статус `Running`.

Чтобы прекратить наблюдение за состоянием подов, нажмите на сочетание клавиш CTRL+C.

Обновление чарта nats-space

Чтобы обновить чарт `nats-space`, выполните следующие действия:

1. Перейдите в каталог с чартом `nats-space`, выполнив следующую команду:

```
cd /opt/r-vision/infra/helm/nats-space
```

2. Создайте переменную, хранящую размер PVC для `nats-space`:

```
PVC_SIZE=$(helm get values -n "$NS" nats-space -o json | jq -r
'.config.jetstream.fileStore.pvc.size // ""')
```

3. Создайте переменную, хранящую идентификатор пространства NATS:

```
SPACE_ID=$(kubectl get cm -n "$NS" space-controller-extravars -o
jsonpath='{.data.SPACE_ID}')
```

4. Проверьте возможность запуска обновления с помощью опции `--dry-run`.



Команда ниже только проверяет возможность запуска обновления, но не выполняет само обновление.

```
helm upgrade -i -n "$NS" nats-space ./ -f ./custom_values.yaml \
  --set config.jetstream.fileStore.pvc.size="${PVC_SIZE:?}" \
  --set config.jetstream.merge.domain="${SPACE_ID:?}" \
  --set container.merge.resources.requests.cpu=<space_resources_requests_cpu> \
  --set container.merge.resources.requests.memory=<space_resources_requests_memory>
\
  --set container.merge.resources.limits.cpu=<space_resources_limits_cpu> \
```

```
  --set container.merge.resources.limits.memory=<space_resources_limits_memory> \
  --set container.env.GOMEMLIMIT=<space_gomemlimit> \
  --dry-run
```

Здесь и далее:

- `<space_resources_requests_cpu>` — минимальные требуемые ресурсы процессора для `nats-space`. По умолчанию `250m` (25% времени одного ядра).
- `<space_resources_requests_memory>` — минимальный требуемый объем ОЗУ для `nats-space`. По умолчанию `1024Mi` (1024 МиБ).
- `<space_resources_limits_cpu>` — максимальное допустимое потребление ресурсов процессора для `nats-space`. По умолчанию `4` (100% времени 4 ядер).
- `<space_resources_limits_memory>` — максимальное допустимое потребление ОЗУ для `nats-space`. По умолчанию `8192Mi` (8192 МиБ).
- `<space_gomemlimit>` — переменная окружения Go, задающая лимит памяти для `nats-space`. По умолчанию `6144MiB` (6144 МиБ).



На следующий шаг можно переходить, только если при выполнении команды выше не возникло ошибок. В противном случае обратитесь в службу поддержки по адресу support@rvision.ru.

5. Установите новую версию NATS для `nats-space`:

```
helm upgrade -i -n "$NS" nats-space ./ -f ./custom_values.yaml \
  --set config.jetstream.fileStore.pvc.size="${PVC_SIZE:?}" \
  --set config.jetstream.merge.domain="${SPACE_ID:?}" \
  --set container.merge.resources.requests.cpu=<space_resources_requests_cpu> \
  --set container.merge.resources.requests.memory=<space_resources_requests_memory>
\
  --set container.merge.resources.limits.cpu=<space_resources_limits_cpu> \
  --set container.merge.resources.limits.memory=<space_resources_limits_memory> \
  --set container.env.GOMEMLIMIT=<space_gomemlimit>
```



Если при обновлении произойдет ошибка, связанная с изменением структуры чарта, удалите предыдущий релиз чарта:

```
helm uninstall <namespace> nats-space
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Затем запустите установку новой версии NATS повторно.

6. Убедитесь, что обновление выполнилось успешно. Для этого запустите мониторинг подов `nats-space`, выполнив следующую команду:

```
kubectl get pod -n "$NS" -l app.kubernetes.io/instance=nats-space --watch
```

После выполнения команды состояние каждого пода начнет выводиться в консоль в реальном времени. При успешном обновлении все поды будут иметь статус `Running`.

Чтобы прекратить наблюдение за состоянием подов, нажмите на сочетание клавиш CTRL+C.

Переход с агентов R-Vision Endpoint на системные агенты

Менеджер агентов системы не поддерживает работу с агентами R-Vision Endpoint. Если вы используете R-Vision Endpoint, необходимо перейти с агентов R-Vision Endpoint на системные агенты. Для этого:

1. Удалите агенты R-Vision Endpoint.
2. Установите новые агенты.

Удаление агентов R-Vision Endpoint

Удаление агента R-Vision Endpoint отличается в зависимости от используемой операционной системы:

- Удаление агента R-Vision Endpoint в ОС Linux
- Удаление агента R-Vision Endpoint в ОС Windows
- Удаление агента R-Vision Endpoint в macOS

Удаление агента R-Vision Endpoint в ОС Linux

Удаление агента R-Vision Endpoint в ОС Linux зависит от используемого дистрибутива:

- Для дистрибутивов семейства Debian:
  1. Удалите агент с помощью команды:

     ```
     sudo dpkg -r rpoint-agent
     ```

  2. Удалите директорию установки агента. Директория установки по умолчанию: `/var/rpoint`.
- Для дистрибутивов семейства RedHat:
  1. Удалите агент с помощью команды:

     ```
     sudo rpm -e rpoint-agent
     ```

  2. Удалите директорию установки агента. Директория установки по умолчанию: `/var/rpoint`.

Удаление агента R-Vision Endpoint в ОС Windows

Чтобы удалить агент R-Vision Endpoint в ОС Windows:

1. Перейдите в раздел **Установка и удаление программ** в параметрах системы.
2. Найдите R-Vision Endpoint в списке установленных программ.
3. Выберите программу R-Vision Endpoint и нажмите на кнопку **Удалить**.
4. Удалите папку установки агента. Папка установки по умолчанию: `C:\Program Files (x86)\rpoint`.

Удаление агента R-Vision Endpoint в macOS

Чтобы удалить агент R-Vision Endpoint в macOS:

1. Отключите автоматический запуск службы агента:

   ```
   launchctl unload -w /Library/LaunchDaemons/ru.rvision.endpoint.plist
   ```

2. Удалите файл службы агента `/Library/LaunchDaemons/ru.rvision.endpoint.plist`.
3. Удалите директорию `/Library/StartupItems/WAZUH/`.
4. Удалите директорию `/Library/RPoint`.
5. Удалите директорию `Applications/R-Vision Endpoint.app`.

Установка новых агентов

Завершив удаление агентов R-Vision Endpoint, установите новые агенты согласно разделу Развертывание агентов.

Развертывание агентов

В данном разделе приведены инструкции по установке и первичной настройке агентов, а также технические требования к хостам, на которых должны работать агенты.

Чтобы установить агенты, выполните следующие действия:

1. Убедитесь, что хост, на котором будет установлен агент, соответствует техническим требованиям.
2. Если планируется установка агента в систему, развернутую из золотого образа (golden image), выполните подготовительные действия. В противном случае перейдите к следующему шагу.
3. Установите агент.
4. Настройте конфигурацию агента.
5. Подключите агент к кластеру.
   - Если требуется подключить агент к кластеру без шифрования, обратитесь к разделу Подключение агента к системе.
   - Если требуется настроить TLS-шифрование подключения, обратитесь к разделу Подключение агента с шифрованием.

При необходимости выполнения определенных операций над агентами, обратитесь к разделу Работа с агентами на хосте.

Чтобы обновить установленный агент, обратитесь к разделу Обновление агента.

Если в процессе установки или обновления агента возникнут неполадки, обратитесь к разделу Устранение проблем при установке или обновлении агентов.

Чтобы удалить агент, обратитесь к разделу Удаление агента.

Технические требования к хосту агента

В данном разделе представлены минимальные технические требования к хосту для функционирования агента.

- Аппаратные требования
- Программные требования

Аппаратные требования

- Для установки:
  - Объем ОЗУ: 1 ГБ.
  - Свободное место на диске: 10 ГБ.
- Для работы агента:
  - CPU: не менее 2 ядер, с минимальной нагрузкой.
  - Объем ОЗУ: не менее 2 ГБ.
  - Свободное место на диске: достаточное для временного хранения данных в случае сбоя передачи в NATS.

Программные требования

Поддерживаемые операционные системы

Поддерживаемые дистрибутивы Linux

Семейство	Операционная система	Версия
Debian	Astra Linux	1.7 "Смоленск" 1.7.6 "Смоленск" 1.8 "Смоленск" 1.8.1 "Воронеж"
	Debian	10 11 12
	Ubuntu	18 20 22 24
RedHat	AlmaLinux	8
	CentOS	7 8.2 9
	Rocky Linux	8.9 9 9.5
	Red Hat Enterprise Linux	8.4 9.4
	РОСА Хром Сервер	2021.1
	SberLinux	9.3
	MosOS	Arbat 15.5
	РЕД ОС	7.3.4 Муром

Поддерживаемые версии Windows

Операционная система	Версия
Windows	Server 2012 R2 Standard Server 2016 Standard Server 2019 Standard Server 2022 Standard 10 Pro 11 Pro

Поддерживаемые версии macOS

Операционная система	Версия
macOS	11 12 13 14.5 14.6 14.6.1 14.7.2 14.7.6 15.1 15.2

Требования к Microsoft Visual C++ Redistributable

Минимальная версия Microsoft Visual C++ Redistributable: 14.29.

Подготовка к установке агента из золотого образа

Агенты могут устанавливаться из золотых образов операционных систем.

Золотой образ (golden image) представляет собой "снимок" системы, предназначенный для создания дополнительных компонентов инфраструктуры на его основе. Золотой образ содержит все настройки и приложения, необходимые для быстрого развертывания системы и начала ее работы, в том числе и агент.



За более подробными сведениями о золотых образах можно обратиться к официальной документации Red Hat.

Перед установкой агента в систему, развернутую из золотого образа, необходимо подготовить ее. Для этого:

1. Перейдите в подкаталог `etc` каталога установки агента.

   Пути до подкаталога `etc` по умолчанию:

   - для ОС Linux: `/opt/rvision/agent/etc/`;
   - для ОС Windows: `C:\Program Files (x86)\RVision\Agent\etc\`;
   - для macOS: `/Library/R-Vision/agent/etc/`.

2. Если в подкаталоге `etc` содержится файл `.installation_id`, удалите его.

   В противном случае агенты на разных хостах, созданные из одного золотого образа, будут обладать одинаковым идентификатором. Это может привести к неожиданным последствиям, например, к проблемам регистрации агентов в системе и их некорректному функционированию.

Установка агента

Установка агента отличается в зависимости от используемой операционной системы на хосте:

- Установка агента в ОС Linux
- Установка агента в ОС Windows
- Установка агента в macOS



- Перед установкой агента ознакомьтесь с техническими требованиями к хосту.
- Для установки агента необходимо обладать правами администратора или суперпользователя (root).

Установка агента в ОС Linux

Чтобы установить агент на хосте с ОС Linux:

1. При необходимости укажите путь установки и путь хранения логов агента в переменных окружения `EVO_AGENT_HOME` и `EVO_AGENT_LOGS`:

   ```
   export EVO_AGENT_HOME=<install_path>
   export EVO_AGENT_LOGS=<log_path>
   ```

   Здесь:

   - `<install_path>` — полный путь установки агента.
   - `<log_path>` — полный путь хранения логов.

   По умолчанию установлены следующие пути:

   - путь установки: `/opt/rvision/agent`;

- путь хранения логов: `/opt/rvision/agent/logs`.

2. Загрузите и запустите установочный пакет.

   Команда запуска зависит от семейства используемого дистрибутива Linux.

   Семейство Debian

   ```
   sudo -E dpkg -i <installer_path>
   ```

   При возникновении ошибок зависимости выполните следующую команду для их исправления:

   ```
   sudo apt-get install -f
   ```

   Семейство RedHat

   ```
   sudo -E rpm -U <installer_path>
   ```

   При возникновении ошибок зависимости выполните следующую команду для их исправления:

   ```
   sudo yum install -y
   ```

   Здесь:

   - `<installer_path>` — полный путь до установщика.

Установка агента в ОС Windows



Для установки агента в ОС Windows необходимо наличие установленного компонента Microsoft Visual C++ Redistributable. Поддерживаемые версии Microsoft Visual C++ Redistributable приведены в разделе Технические требования к хосту агента.

Установщик последней версии Microsoft Visual C++ Redistributable можно скачать с официального сайта Microsoft.



Установка агента в ОС Windows может выполняться из установщиков следующего вида:

- `evo-agent-<build_no>.d-<bit_depth>.msi` — используется для установки только агента. В этом случае необходимо установить компонент Microsoft Visual C++ Redistributable вручную.
- `evo-agent-<build_no>.s-<bit_depth>.msi` — используется для установки агента

вместе с Microsoft Visual C++ Redistributable.

Здесь:

- `<build_no>` — номер сборки агента, например: `3.1.0.16940`.
- `<bit_depth>` — разрядность архитектуры процессора на хосте: `x86` или `x64`.

Чтобы установить агент на хосте с ОС Windows:

1. При необходимости укажите путь установки агента в переменной окружения `RVISION_AGENT_INSTALL_PATH`.

   По умолчанию агент устанавливается в папку `C:\Program Files (x86)\RVision\Agent`.

2. Загрузите и запустите установочный пакет с помощью команды:

   ```
   msiexec /i <installer_path> /L*V '<log_path>'
   ```

   Здесь:

   - `<installer_path>` — полный путь до установщика.
   - `<log_path>` — полный путь хранения логов.

   Для корректной установки агента требуется запускать установочный пакет с правами администратора. Если появляется сообщение об отсутствии DLL-файлов, скачайте их из актуальной библиотеки Microsoft Visual C++ Redistributable и установите их вручную. Агент применит установленные DLL-файлы для продолжения установки.

Установка агента в macOS

Чтобы установить агент на хосте с macOS, загрузите и запустите установочный пакет с помощью команды:

```
sudo installer -pkg <installer_path> -target /
```

Здесь:

- `<installer_path>` — полный путь до установщика.



Пути установки агента и хранения логов в macOS установлены по умолчанию и не могут быть изменены:

- путь установки: `/Library/R-Vision/agent`;
- путь хранения логов: `/Library/R-Vision/agent/logs`.

Настройка конфигурации агента

Конфигурация агента настраивается в файле `rpoint.conf`.

Файл конфигурации `rpoint.conf` создается автоматически при установке агента и хранится в подкаталоге `etc` каталога установки.



Пути хранения файла конфигурации по умолчанию:

- для ОС Linux: `/opt/rvision/agent/etc/`;
- для ОС Windows: `C:\Program Files (x86)\RVision\Agent\etc\`;
- для macOS: `/Library/R-Vision/agent/etc/`.

Структура файла конфигурации

Файл конфигурации `rpoint.conf` содержит следующие настройки:

- `log` — настройки логирования.
- `nats` — настройки подключения к локальному NATS-серверу.
- `custom_deny_exports` — дополнительные значения `deny_exports`.
- `custom_deny_imports` — дополнительные значения `deny_imports`.
- `default_deny_exports_enabled` — признак использования значений `deny_exports` по умолчанию.
- `default_deny_imports_enabled` — признак использования значений `deny_imports` по умолчанию.
- `nats_server` — параметры запуска локального NATS-сервера.
- `inventory` — настройки сбора инвентарной информации.
- `messages_sender` — настройки модуля отправки сообщений.
- `go_memory_limit` — ограничение на потребление ОЗУ брокером сообщений NATS.
- `restart_nats_on_isolation` — признак перезапуска модуля `nats-server` при изменении настроек изоляции.
- `isolation_monitoring_interval` — интервал проверки изоляции.
- `agent_id` — идентификатор агента.
- `agent_id_generation_method` — способ автоматической генерации идентификатора агента.
- `gateway_discovery_enabled` — возможность переподключения агента к другому шлюзу.
- `limits` — ограничения на потребление ОЗУ агентом.

▼ *Пример содержимого файла конфигурации* `rpoint.conf`

```
{
  "log": {
    "file_path": "logs/rpoint.log",
    "level": "info",
    "rotation_size": 10485760,
    "target_file_name": "rpoint_%N.log",
    "max_messages_queue_size": 10000
```

```
  },
  "nats": {
    "local": {
      "connection": {
        "url": "tls://127.0.0.1:4111"
      }
    }
  },
  "custom_deny_exports": [],
  "custom_deny_imports": [],
  "default_deny_exports_enabled": true,
  "default_deny_imports_enabled": true,
  "nats_server": {
    "host": "127.0.0.1",
    "port": 4111,
    "jetstream": {
      "store_dir": "./nats",
      "max_memory_store": 1073741824,
      "max_file_store": 10737418240
    },
    "leafnodes": {
        "remotes": [
            {
              "url": "nats://worker01.rpoint.local:32000"
            }
        ]
    },
    "log_file": "logs/nats.log",
    "log_size_limit": 10485760,
    "logfile_max_num": 10
  },
  "inventory": {
    "monitoring_interval": 86400
  },
  "messages_sender": [
    {
      "name": "ib_events",
      "read_stream": "local-ib-events",
      "read_subject": "local.e.ib.>",
      "max_bytes": 100000000,
      "should_log_ack": false,
      "max_parallel_messages": 10
    },
    {
      "name": "audit_events",
      "read_stream": "local-agent-audit",
      "read_subject": "local.events.siem.audit.v1.SystemAuditEvent",
      "max_bytes": 50000000,
      "should_log_ack": false,
      "ack_timeout": 3000,
      "headers": {
```

```
        "Nats-Expected-Stream": "events-siem-audit-system-audit"
      },
      "max_parallel_messages": 10
    }
  ],
  "go_memory_limit": "1GiB",
  "restart_nats_on_isolation": true,
  "isolation_monitoring_interval": 1.0,
  "agent_id": "7b976b06-ff10-4266-80c2-ce9fbe66f939",
  "gateway_discovery_enabled": false
}
```

Секция `log`

В данной секции задаются настройки логирования.



Рекомендуется оставлять данную секцию без изменений.

Параметр	Описание
`file_path`	Путь к файлу логирования на устройстве.
`level`	Уровень логирования. Доступные значения: • `fatal` — записи о критических ошибках в работе ПО, приводящих к аварийному завершению работы. • `error` — записи уровня `fatal`, а также ошибки, нарушающие работу функционала. • `warning` — записи уровня `error`, а также предупреждения, не приводящие к нарушению работы функционала, но требующие внимания. • `info` — записи уровня `warning`, а также информационные сообщения о штатной работе ПО. • `debug` — записи уровня `info`, а также отладочная информация для разработчиков. • `trace` — записи уровня `debug`, а также наиболее подробная отладочная информация для разработчиков.
`rotation_size`	Размер файла в байтах, по достижении которого выполняется ротация логов. Задается, если требуется ротация по размеру файла. Не может задаваться одновременно с параметром `time_rotation`.

Параметр	Описание
`time_rotation`	Период ротации логов. Задается, если требуется ротация по времени. Указывается в формате JSON-объекта со следующими полями: • `hours` — часы; • `minutes` — минуты; • `seconds` — секунды. Не может задаваться одновременно с параметром `rotation_size`.
`target_file_name`	Шаблон имени файла после ротации логов. Если в параметре задан шаблон, содержащий символы `%N`, то вместо этих символов в каждом файле логов будет добавлен порядковый номер ротации. Например, для шаблона `rpoint_%N.log` итоговые файлы будут иметь названия `rpoint_1.log`, `rpoint_2.log`, `rpoint_3.log`.
`max_messages_queue_size`	Максимальное количество сообщений логов, которые будут сохранены в кэше при сбое работы процесса rpoint-launcher.

▼ *Пример секции* `log`

```
"log": {
  "file_path": "logs/rpoint.log",
  "level": "info",
  "rotation_size": 10485760,
  "target_file_name": "rpoint_%N.log",
  "max_messages_queue_size": 10000
}
```

Секция `nats`

В данной секции задаются настройки подключения агента к локальному NATS-серверу для обмена сообщениями с менеджером агентов.

Параметр	Описание
`local`	Конфигурация подключения к локальному NATS-серверу. Указывается в формате JSON-объекта `connection` с полем `url`. В поле `url` задается URL-адрес локального NATS-сервера в формате `tls://127.0.0.1:<port>`. Здесь: • `<port>` — порт подключения к серверу. Порт должен совпадать со значением `port` в параметре `nats_server`.

▼ *Пример секции* `nats`

```
"nats": {
  "local": {
    "connection": {
      "url": "tls://127.0.0.1:4111"
    }
  }
}
```

Параметр `custom_deny_exports`

Параметр `custom_deny_exports` позволяет задать дополнительные значения параметра `deny_exports` при генерации файла конфигурации `nats-leafnode.conf`.



- Файл `nats-leafnode.conf` содержит настройки подключения к менеджеру агентов и генерируется автоматически при запуске агента.
- В параметре `deny_exports` файла `nats-leafnode.conf` указываются стримы NATS, запрещенные к экспорту из NATS-сервера агента в шлюз.

▼ *Пример параметра* `custom_deny_exports`

```
"custom_deny_exports": []
```

Параметр `custom_deny_imports`

Параметр `custom_deny_imports` позволяет задать дополнительные значения параметра `deny_imports` при генерации файла конфигурации `nats-leafnode.conf`.



- Файл `nats-leafnode.conf` содержит настройки подключения к менеджеру агентов и генерируется автоматически при запуске агента.
- В параметре `deny_imports` файла `nats-leafnode.conf` указываются стримы NATS, запрещенные к импорту в NATS-сервер агента из шлюза.

▼ *Пример параметра* `custom_deny_imports`

```
"custom_deny_imports": []
```

Параметр `default_deny_exports_enabled`

Параметр `default_deny_exports_enabled` определяет, нужно ли использовать значения параметра `deny_exports` по умолчанию при генерации файла конфигурации `nats-leafnode.conf`.



- Файл `nats-leafnode.conf` содержит настройки подключения к менеджеру агентов и генерируется автоматически при запуске агента.
- В параметре `deny_exports` файла `nats-leafnode.conf` указываются стримы NATS, запрещенные к экспорту из NATS-сервера агента в шлюз.

Параметр может принимать одно из следующих значений:

- `true` — в `deny_exports` будут добавлены значения по умолчанию, а также дополнительные значения из параметра `custom_deny_exports`.
- `false` — в `deny_exports` будут добавлены только дополнительные значения из параметра `custom_deny_exports`.

▼ *Значения* `deny_exports` *по умолчанию*

```
"deny_exports": [
    "$KV.local-configs.>",
    "$KV.command-executor-schedule.>",
    "$KV.local-modules-status.>",
    "local-module-rpc.>",
    "local-ep-to-manager.>",
    "local.>",
    "$JS.ACK.local-agent-audit.>",
    "$JS.ACK.local-ib-events.>",
    "$JS.ACK.local-module-rpc.>"
]
```

▼ *Пример параметра* `default_deny_exports_enabled`

```
"default_deny_exports_enabled": true
```



- Если параметр не указан, то агент автоматически воспринимает его установленным в значении `true`.
- Вы можете изменить значение параметра `default_deny_exports_enabled` при запуске агента с помощью следующих атрибутов:
  - `--default-deny-exports-disable` — меняет значение параметра на `false`.
  - `--default-deny-exports-enable` — меняет значение параметра на `true`.

Параметр `default_deny_imports_enabled`

Параметр `default_deny_imports_enabled` определяет, нужно ли использовать значения параметра `deny_imports` по умолчанию при генерации файла конфигурации `nats-leafnode.conf`.



- Файл `nats-leafnode.conf` содержит настройки подключения к менеджеру агентов и генерируется автоматически при запуске агента.
- В параметре `deny_imports` файла `nats-leafnode.conf` указываются стримы NATS, запрещенные к импорту в NATS-сервер агента из шлюза.

Параметр может принимать одно из следующих значений:

- `true` — в `deny_imports` будут добавлены значения по умолчанию, а также дополнительные значения из параметра `custom_deny_imports`.
- `false` — в `deny_imports` будут добавлены только дополнительные значения из параметра

`custom_deny_imports`.

▼ *Значения* `deny_imports` *по умолчанию*

```
"deny_imports": [
    "$KV.local-configs.>",
    "$KV.command-executor-schedule.>",
    "$KV.local-modules-status.>",
    "local-module-rpc.>",
    "local-ep-to-manager.>",
    "local.>",
    "e.ib.>",
    "ep-to-agents.>",
    "$JS.ACK.local-agent-audit.>",
    "$JS.ACK.local-ib-events.>",
    "$JS.ACK.local-module-rpc.>",
    "$JS.ACK.ep-to-agents.>"
]
```

▼ *Пример параметра* `default_deny_imports_enabled`

```
"default_deny_imports_enabled": true
```



- Если параметр не указан, то агент автоматически воспринимает его установленным в значении `true`.
- Вы можете изменить значение параметра `default_deny_imports_enabled` при запуске агента с помощью следующих атрибутов:
  - `--default-deny-imports-disable` — меняет значение параметра на `false`.
  - `--default-deny-imports-enable` — меняет значение параметра на `true`.

Секция `nats_server`

В данной секции задаются параметры запуска локального NATS-сервера.

Параметр	Описание
`host`	IP-адрес или полное доменное имя (FQDN) локального NATS-сервера. Не меняйте значение данного параметра.
`port`	Порт локального NATS-сервера. Значение должно совпадать с портом, указанным в значении `local` в параметре `nats`.

Параметр	Описание
`jetstream`	Параметры хранения данных брокера сообщений NATS JetStream: • `store_dir` — путь к папке, в которой хранятся данные. • `max_memory_store` — максимальный объем занимаемой памяти в байтах. • `max_file_store` — максимальный объем файла на дисковом пространстве в байтах. Не меняйте значение данного параметра.
`leafnodes`	Настройки шлюзов для подключения. Указываются в виде JSON-объекта `remotes` с полем `url`, в котором задается URL-адрес для подключения к шлюзу. Для каждого шлюза без TLS-шифрования в поле `url` указывается URL-адрес в формате `nats://<addr>:<port>`. Для каждого шлюза с TLS-шифрованием адрес указывается в формате `tls://<addr>:<port>`. Здесь: • `<addr>` — адрес подключения к шлюзу. • `<port>` — порт подключения к шлюзу. Подробности о подключении агентов к шлюзам приведены в разделах Подключение агента к системе и Настройка шифрования между агентом и шлюзом.
`log_file`	Путь к файлу с логами NATS. Не меняйте значение данного параметра.
`log_size_limit`	Максимальный размер файла с логами NATS в байтах. При достижении данного размера будет создан новый файл.
`logfile_max_num`	Максимальное количество файлов с логами NATS. При достижении данного количества будет произведена ротация логов.

▼ *Пример секции* `nats_server`

```
"nats_server": {
  "host": "127.0.0.1",
  "port": 4111,
  "jetstream": {
    "store_dir": "./nats",
    "max_memory_store": 1073741824,
    "max_file_store": 10737418240
  },
  "leafnodes": {
      "remotes": [
```

```
            {
              "url": "nats://worker01.rpoint.local:32000"
            }
        ]
    },
    "log_file": "logs/nats.log",
    "log_size_limit": 10485760,
    "logfile_max_num": 10
}
```

Секция `inventory`

В данной секции задаются настройки сбора инвентарной информации об агенте.



Рекомендуется оставлять данную секцию без изменений.

Параметр	Описание
`monitoring_interval`	Интервал сбора инвентарной информации в секундах.

▼ *Пример секции* `inventory`

```
"inventory": {
  "monitoring_interval": 86400
}
```

Секция `messages_sender`

В данной секции задаются параметры модуля отправки сообщений.



Не меняйте значения параметров в данной секции.

Настройки задаются в виде массива параметров для каждой очереди сообщений. Параметры очереди представлены в таблице ниже.

Параметр	Описание
`name`	Название канала NATS для передачи сообщений.
`read_stream`	Название потока NATS для чтения и передачи сообщений.
`read_subject`	Название темы NATS для чтения и передачи сообщений.
`max_bytes`	Максимальный размер очереди в байтах, из которой происходит чтение событий. Если параметр не задан, очередь не ограничена.

Параметр	Описание
`should_log_ack`	Признак, нужно ли логировать сообщения для данной очереди. Следует соблюдать осторожность при активации данного параметра. При большом количестве событий в очереди возможно переполнение логов неинформативными сообщениями.
`max_parralel_messages`	Максимальное количество сообщений, которое можно отправлять параллельно, для данной очереди.
`headers`	Заголовки, которые нужно добавлять к сообщению при публикации в стрим. Заголовки задаются в виде массива параметров `Nats-Expected-Stream`, где указывается ожидаемый стрим получения.

▼ *Пример секции* `messages_sender`

```
"messages_sender": [
  {
    "name": "ib_events",
    "read_stream": "local-ib-events",
    "read_subject": "local.e.ib.>",
    "max_bytes": 100000000,
    "should_log_ack": false,
    "max_parallel_messages": 10
  },
  {
    "name": "audit_events",
    "read_stream": "local-agent-audit",
    "read_subject": "local.events.siem.audit.v1.SystemAuditEvent",
    "max_bytes": 50000000,
    "should_log_ack": false,
    "ack_timeout": 3000,
    "headers": {
      "Nats-Expected-Stream": "events-siem-audit-system-audit"
    },
    "max_parallel_messages": 10
  }
]
```

Параметр `go_memory_limit`

Данный параметр позволяет ограничивать максимальный размер оперативной памяти, потребляемой брокером сообщений NATS. Задаваемое ограничение указывается с единицами измерения.



Рекомендуется оставлять данный параметр без изменений.

▼ *Пример параметра* `go_memory_limit`

```
"go_memory_limit": "1GiB"
```

В представленном примере задано ограничение на потребление ОЗУ в 1 ГиБ.

Параметр `restart_nats_on_isolation`

Данный параметр определяет, нужно ли перезапускать модуль `nats-server` при установке или снятии изоляции хоста.



Параметр рекомендуется включать только в macOS в связи с особенностями работы NATS.

▼ *Пример параметра* `restart_nats_on_isolation`

```
"restart_nats_on_isolation": true,
```

Параметр `isolation_monitoring_interval`

При включении изоляции хоста автоматически активируется система мониторинга, который проверяет состояние изоляции. Если произойдет попытка выключить изоляцию, то система мониторинга включит изоляцию обратно.

Данный параметр позволяет указать интервал между проверками изоляции хоста системой мониторинга в секундах.



Параметр работает только в macOS.

▼ *Пример параметра* `isolation_monitoring_interval`

```
"isolation_monitoring_interval": 1.0
```

Параметры `agent_id` и `agent_id_generation_method`

По умолчанию идентификатор агента генерируется автоматически на основе идентификатора устройства, где он устанавливается. Это соответствует установке параметра `agent_id_generation_method` со значением `machineId`.

При необходимости вы можете изменить способ генерации агента с помощью одного из следующих параметров:

- `agent_id` — позволяет указать идентификатор агента вручную в формате UUID.

  ▼ *Пример параметра* `agent_id`

  ```
  "agent_id": "7b976b06-ff10-4266-80c2-ce9fbe66f939"
  ```

- `agent_id_generation_method` — позволяет указать, каким способом должен генерироваться идентификатор агента. Доступные значения:

- `machineId` — идентификатор агента будет сгенерирован по методу UUIDv5 на основе пространства имен OID и идентификатора устройства.
- `mac` — идентификатор агента будет сгенерирован по методу UUIDv5 на основе пространства имен OID и первого MAC-адреса хоста.
- `random` — будет сгенерирован случайный идентификатор агента в формате UUID.

▼ *Пример параметра* `agent_id_generation_method`

```
"agent_id_generation_method": "machineId"
```

Алгоритм генерации идентификатора агента

Сгенерированный идентификатор агента сохраняется в файле `.agent_id` в подкаталоге `etc` каталога установки.



Пути хранения файла идентификатора по умолчанию:

- для ОС Linux: `/opt/rvision/agent/etc/`;
- для ОС Windows: `C:\Program Files (x86)\RVision\Agent\etc\`;
- для macOS: `/Library/R-Vision/agent/etc/`.

Генерация идентификатора происходит при запуске агента и выполняется по следующему алгоритму:

1. Сканируется подкаталог `etc` на наличие файла хранения идентификатора `.agent_id`.
   - Если файл найден и в нем содержится идентификатор агента в формате UUID, то идентификатор уже сгенерирован и повторная генерация не требуется.
   - В противном случае — переход к шагу 2.
2. Сканируется подкаталог `etc` на наличие файла конфигурации с настройками подключения к менеджеру `nats-leafnode.conf`.
   - Если файл найден и в нем содержится идентификатор агента в формате UUID, то идентификатор уже сгенерирован и повторная генерация не требуется.
   - В противном случае — переход к шагу 3.
3. Сканируется файл конфигурации агента `rpoint.conf`. Способ генерации идентификатора зависит от заданных параметров `agent_id` и `agent_id_generation_method`. В таблице ниже представлены возможные соотношения параметров и итоговый способ генерации идентификатора.

agent_id	agent_id_generation_method	Как генерируется идентификатор
Указано значение в формате UUID	Значение не указано или указано любое значение	Берется значение `agent_id`
Значение не указано или указано не в формате UUID	Значение не указано или указано некорректное значение	Используется автоматический способ генерации по `machineId`

agent_id	agent_id_generation_method	Как генерируется идентификатор
Значение не указано или указано не в формате UUID	Указано любое корректное значение	Используется тот способ генерации, который указан в `agent_id_generation_method`

4. Сгенерированный идентификатор агента сохраняется в файле `.agent_id` в подкаталоге `etc`.

Параметр `gateway_discovery_enabled`

Параметр `gateway_discovery_enabled` управляет возможностью автоматического переключения агента между шлюзами, если его шлюз остановлен.

Параметр может принимать одно из следующих значений:

- `true` — агент может автоматически переключаться между шлюзами. Подробные сведения о переключении между шлюзами приведены в разделе Алгоритм переключения агента между шлюзами.
- `false` — агент всегда будет подключаться только к тому шлюзу, который указан в секции `nats_server` файла `rpoint.conf`. В таком случае при остановке шлюза агент будет переходить в статус **Офлайн**.

▼ *Пример параметра* `gateway_discovery_enabled`

```
"gateway_discovery_enabled": false
```



- Если параметр не указан, то агент автоматически воспринимает его установленным в значении `true`.
- Вы можете изменить значение параметра `gateway_discovery_enabled` при запуске агента с помощью следующих атрибутов:
  - `--gateway-discovery-disable` — меняет значение параметра на `false`.
  - `--gateway-discovery-enable` — меняет значение параметра на `true`.

Алгоритм переключения агента между шлюзами

Если параметр `gateway_discovery_enabled` не задан в конфигурации агента или установлен в значение `true`, агент может переключаться между шлюзами по следующему алгоритму:

1. При подключении к кластеру системы агент получает информацию обо всех шлюзах и обновляет данную информацию по мере ее изменения.
2. Если происходит остановка шлюза агента, а сам агент в этот момент находится в статусе **Онлайн**, он автоматически повторно подключается к кластеру через любой работающий шлюз и остается в статусе **Онлайн**.

   Если при остановке шлюза агент не находится в статусе **Онлайн**, то при попытке запуска агента механизм переключения между шлюзами не сработает и агент не будет запущен.

Секция `limits`

В данной секции задаются ограничения на потребление ОЗУ агентом.



Если секция отсутствует, то применяются значения по умолчанию.

Параметр	Описание
`exceed_limit_timeout_seconds`	Временной интервал в секундах, за который определяется превышение лимита. Допустимые значения: от `15` до `1800`. Значение по умолчанию: `15`. Например, если установлен интервал в 15 секунд, то превышение лимита фиксируется, если в течение 15 секунд агент потребляет больше ОЗУ, чем установлено ограничениями.
`total`	Настройки лимитов потребления агента в целом: • `memory_percent` — предел потребления ОЗУ суммарно всеми процессами агента в процентах от всего доступного объема ОЗУ. Допустимые значения: от `10` до `100`. Значение по умолчанию: `30` • `memory_mb` — предел потребления ОЗУ суммарно всеми процессами агента в МБ. Допустимые значения: `100` и выше. Значение по умолчанию: `6000`.

Параметр	Описание
`modules`	Настройки лимитов потребления агента по модулям. Представляет собой массив JSON-объектов со следующими полями: • `name` — имя модуля агента. Допустимые значения: ◦ `active-response`; ◦ `file-reader`; ◦ `endpoint-security`; ◦ `object-access`; ◦ `process-creation`; ◦ `service-viewer`; ◦ `login-logoff`; ◦ `messages-sender`; ◦ `nats-server`. • `memory_percent` — предел потребления ОЗУ конкретным процессом агента в модуле в процентах от всего доступного объема ОЗУ. Допустимые значения: от `10` до `100`. Значение по умолчанию: `30`. • `memory_mb` — предел потребления ОЗУ конкретным процессом агента в модуле в МБ. Допустимые значения: `100` и выше. Значение по умолчанию: `500`. • `priority` — целое число, определяющее приоритет отключения процессов. Чем меньше значение, тем выше приоритет. Первыми отключаются процессы с наивысшим приоритетом. Возможные значения: `1` и выше. Значение по умолчанию: `5`.

▼ *Информация о модулях агента*

Под модулями агента подразумеваются процессы, реализующие ту или иную его функциональность:

- `active-response` — выполнение специальных действий для быстрого реагирования на инциденты или угрозы.
- `file-reader` — чтение журналов ОС. Настраивается в параметре **Чтение файлов** в конфигурации группы агентов.
- `endpoint-security` — работа Endpoint Security Framework в macOS для чтения событий **Вход/выход пользователей**, **Создание процессов** и **Управление пользователями**.

- `object-access` — управление доступом пользователей к объектам. Настраивается в параметре **Доступ к объектам** в конфигурации группы агентов.
- `process-creation` — анализ создания процессов. Настраивается в параметре **Создание процессов** в конфигурации группы агентов.
- `service-viewer` — отслеживание сервисов. Настраивается в параметре **Добавление или изменение сервисов** в конфигурации группы агентов.
- `login-logoff` — анализ входов и выходов пользователей. Настраивается в параметре **Вход/выход пользователей** в конфигурации группы агентов.
- `messages-sender` — отправка сообщений в шлюзы.
- `nats-server` — работа локального NATS-сервера на агенте.

▼ *Пример секции* `limits`

```
"limits": {
  "exceed_limit_timeout_seconds": 15,
  "total": {
    "memory_percent": 50,
    "memory_mb": 6000
  },
  "modules": [
    {
      "name": "active-response",
      "memory_percent": 20,
      "memory_mb": 500,
      "priority": 1
    },
    {
      "name": "file-reader",
      "memory_percent": 20,
      "memory_mb": 500,
      "priority": 2
    },
    {
      "name": "endpoint-security",
      "memory_percent": 20,
      "memory_mb": 500,
      "priority": 3
    },
    {
      "name": "object-access",
      "memory_percent": 20,
      "memory_mb": 500,
      "priority": 3
    },
    {
      "name": "process-creation",
      "memory_percent": 20,
      "memory_mb": 500,
      "priority": 3
```

```
    },
    {
      "name": "service-viewer",
      "memory_percent": 20,
      "memory_mb": 500,
      "priority": 3
    },
    {
      "name": "login-logoff",
      "memory_percent": 20,
      "memory_mb": 500,
      "priority": 3
    },
    {
      "name": "messages-sender",
      "memory_percent": 20,
      "memory_mb": 1000,
      "priority": 5
    },
    {
      "name": "nats-server",
      "memory_percent": 50,
      "memory_mb": 4000,
      "priority": 6
    },
    {
      "name": "launcher",
      "memory_percent": 50,
      "memory_mb": 100,
      "priority": 7
    }
  ]
}
```

Подключение агента к системе

Чтобы подключить агент к основному экземпляру системы или сателлиту, выполните следующие действия:

1. При необходимости подключитесь к пространству сателлита.
2. Создайте и настройте шлюз.
3. Подключите агент к шлюзу.



В данной инструкции рассматривается подключение агента без шифрования соединения. Если необходимо настроить агент с шифрованием, обратитесь к разделу Настройка шифрования между агентом и шлюзом.



Все агенты в системе должны быть подключены к кластеру либо с шифрованием, либо без шифрования. В ситуациях, когда для одной части шлюзов настроено TLS-

шифрование, а для другой — нет, корректное подключение агентов к кластеру не гарантируется.

Подключение к пространству сателлита

Для подключения агентов к системе используются шлюзы. Шлюзы могут располагаться как в основном экземпляре системы, так и в сателлитах.

Если для соединения с агентом планируется использовать шлюз, расположенный в сателлите, необходимо заранее подключиться к кластеру этого сателлита.

Управление сателлитом осуществляется за счет запускаемых в нем сервисов через веб-интерфейс основного экземпляра системы. Чтобы подключиться к сателлиту, необходимо создать подключение к его пространству. Для этого:

1. Войдите в веб-интерфейс основного экземпляра системы.
2. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**.
3. Нажмите на кнопку **Создать** (┼). Отобразится окно создания подключения.
4. Введите идентификатор пространства сателлита в поле **ID**.
5. Введите название подключения. Название может быть любым.
6. В поле **IP или домен** введите IP-адрес или доменное имя (FQDN) одного из узлов кластера, в котором развернут сателлит.
7. В поле **Порт** введите номер TCP-порта, указанного на этапе **NATS satellite setup** установки сателлита.
8. При необходимости введите описание подключения к пространству.
9. Нажмите на кнопку **Создать**. Система создаст подключение к пространству и отобразит соответствующее уведомление. Подключение отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Пространства**.

У созданного подключения отобразится статус **Подключается**. По завершении подключения статус изменится на **Активен**.



Время на инициализацию подключения зависит загруженности кластеров и каналов связи и может достигать 5—10 минут.

Создание и настройка шлюза

Чтобы настроить шлюз:

1. Создайте шлюз. При создании укажите следующие параметры:
   - Пространство, к которому необходимо подключиться:
     - Если необходимо настроить вынесенный шлюз в сателлите, выберите пространство, созданное на шаге Подключение к пространству сателлита.
     - Если необходимо настроить шлюз в основном экземпляре системы, выберите **Центральное пространство**.

- IP- или FQDN-адрес для подключения к шлюзу.

  ▼ *Особенности задания адресов для шлюза*

  Для подключения к шлюзу необходимо использовать адрес узла кластера, где создается шлюз. Так, например:

  - Если шлюз создается в основном экземпляре системы, то используйте IP-адрес любого узла центрального кластера.
  - Если шлюз создается в сателлите, используйте IP-адреса любого узла кластера сателлита.

  FQDN-адрес можно использовать, только если в инфраструктуре существует DNS-сервер и администратор настроил прямую зону (forward zone). Проверить, настроена ли такая зона, можно с помощью утилиты nslookup:

  ```
  nslookup <domain_name>
  ```

  Здесь:

  - `<domain_name>` — доменное имя.

  Обратные DNS-запросы для определения FQDN-адресов по IP-адресам возможны, если на DNS-сервере настроена обратная зона (reverse zone). В ней хранятся записи, сопоставляющие IP-адреса с доменными именами. Проверить, настроена ли такая зона, можно также с помощью утилиты nslookup:

  ```
  nslookup <IP_address>
  ```

  Здесь:

  - `<IP_address>` — IP-адрес узла.

  Если обратная зона не настроена, свяжитесь с сетевым администратором.

  Если ваш компьютер имеет доступ к кластеру с установленным файлом конфигурации `kubeconfig`, вы можете получить сведения обо всех IP-адресах узлов кластера. Для сбора IP-адресов всех узлов в кластере и их вывода в виде строки используйте команду:

  ```
  kubectl get node -o json
  --output='jsonpath={.items[*].status.addresses[0].address}'
  ```

- Порт для подключения к шлюзу. Допустимые значения: 30 000—32 767.

  Порт должен быть уникальным в рамках инсталляции системы.

- Максимальный размер буфера в МБ для хранения событий ИБ.



Эффективный размер буфера будет равен 80% от указанного значения.

2. Запустите шлюз. Шлюз станет доступен для подключения к нему агентов.

Подключение агента к шлюзу

Чтобы подключить агент к шлюзу:

1. Установите агент, если он еще не установлен.

   Если агент уже установлен и подключен к другому кластеру, удалите папку `nats` на хосте агента для его корректной работы. Способ удаления зависит от используемой ОС на хосте.

   Linux

   ```
   rm -rf "<nats_path>"
   ```

   Windows

   Используйте командную оболочку PowerShell:

   ```
   Remove-Item -Path "<nats_path>" -Recurse
   ```

   macOS

   ```
   rm -rf "<nats_path>"
   ```

   Здесь:

   - `<nats_path>` — путь к папке `nats` на хосте.

2. Настройте сетевой доступ к шлюзу по необходимому порту. Для этого задайте настройки подключения к шлюзу в параметре `leafnodes` в секции `nats_server` файла конфигурации `rpoint.conf`.

   ▼ *Особенности задания параметра* `leafnodes`

   Для каждого шлюза без TLS-шифрования в поле `url` указывается URL-адрес в формате `nats://<addr>:<port>`.

   Для каждого шлюза с TLS-шифрованием адрес указывается в формате `tls://<addr>:<port>`.

   Здесь:

   - `<addr>` — адрес подключения к шлюзу.
   - `<port>` — порт подключения к шлюзу.

   Например:

```
"leafnodes": {
  "remotes": [
    {
      "url": "nats://worker01.rpoint.local:32000"
    }
  ]
}
```

3. Запустите агент.



При необходимости вы можете пропустить шаг 2 и запустить агент сразу с необходимыми настройками подключения к шлюзу. Для этого к команде запуска необходимо добавить параметр `--gateway-url`. Например:

Linux

Для всех дистрибутивов, кроме Astra Linux и РОСА Хром Сервер:

- Если переменная окружения `EVO_AGENT_HOME` не задана:

```
sudo /opt/rvision/agent/bin/rpoint-launcher start
--gateway-url=nats://<addr>:<port>
```

- Если переменная окружения `EVO_AGENT_HOME` задана:

```
sudo -E ${EVO_AGENT_HOME}/bin/rpoint-launcher start
--gateway-url=nats://<addr>:<port>
```

Windows

- Если переменная `RVISION_AGENT_INSTALL_PATH` не задана:

```
& 'C:\Program Files (x86)\RVision\Agent\bin\rpoint-
launcher.exe' start --gateway-url=nats://<addr>:<port>
```

- Если переменная `RVISION_AGENT_INSTALL_PATH` задана:

```
& "${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe"
start --gateway-url=nats://<addr>:<port>
```

macOS

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher start
--gateway-url=nats://<addr>:<port>
```

Здесь:

- `<addr>` — адрес подключения к шлюзу.
- `<port>` — порт подключения к шлюзу.

Если настройка была выполнена корректно, агент появится в разделе **Агенты** → **Агенты** веб-интерфейса системы.

Подключение агента с шифрованием

Шифрование соединения обеспечивает безопасную передачу данных между агентами и шлюзами. Шифрование осуществляется через протокол TLS (Transport Layer Security).

Настройка подключения агента к системе с шифрованием данных состоит из следующих этапов:

1. Добавление TLS-сертификата на агенте
2. Настройка шифрования между агентом и шлюзом



Все агенты в системе должны быть подключены к кластеру либо с шифрованием, либо без шифрования. В ситуациях, когда для одной части шлюзов настроено TLS-шифрование, а для другой — нет, корректное подключение агентов к кластеру не гарантируется.

Добавление TLS-сертификата на агенте

Для шифрования соединения между агентами и шлюзами необходимо использовать TLS-сертификат.

TLS-сертификат — это цифровой документ, позволяющий использовать протокол безопасного соединения TLS (Transport Layer Security). Наличие TLS-сертификата предоставляет следующие гарантии:

- Ресурс, на котором расположен агент, является подлинным и не принадлежит злоумышленникам.
- Передаваемая информация защищена от злоумышленников благодаря установке зашифрованного соединения между агентом и шлюзом.

Чтобы добавить TLS-сертификат на агент:

1. Сгенерируйте TLS-сертификат.
2. Установите корневой сертификат на компьютере агента.



Добавление сертификатов выполняется после установки агента.

Генерация TLS-сертификата



В представленной инструкции в качестве примера используются файлы со следующими названиями:

- `ca.key` — файл приватного ключа корневого сертификата;
- `ca.crt` — файл корневого сертификата;
- `server.ext` — файл конфигурации TLS-сертификата;
- `server.key` — файл приватного ключа TLS-сертификата;
- `server.csr` — файл запроса на выпуск TLS-сертификата;
- `server.crt` — файл TLS-сертификата.

Чтобы сгенерировать TLS-сертификат, выполните на компьютере с установленным агентом следующие действия:

1. Создайте файл `server.ext` со следующей конфигурацией TLS-сертификата:

```
authorityKeyIdentifier=keyid,issuer

basicConstraints=CA:FALSE

keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment

subjectAltName = @alt_names

[alt_names]
DNS.1 = <domain_name_1>
...
DNS.N = <domain_name_N>
```

   Здесь:

   - `<domain_name_1>`, ..., `<domain_name_N>` — доменные имена шлюзов, с которыми необходимо настроить безопасное соединение.

   ▼ *Пример конфигурации TLS-сертификата*

```
authorityKeyIdentifier=keyid,issuer

basicConstraints=CA:FALSE

keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment

subjectAltName = @alt_names

[alt_names]
DNS.1 = worker01.rpoint.local
DNS.2 = worker02.rpoint.local
DNS.3 = worker03.rpoint.local
```

2. Создайте приватный ключ `ca.key` для корневого сертификата:

```
openssl genrsa -out ca.key 4096
```

3. Создайте самоподписанный корневой сертификат `ca.crt`:

```
openssl req -x509 -new -nodes -key ca.key -sha256 -days 3650 -out ca.crt
```

4. Создайте приватный ключ `server.key` для TLS-сертификата:

```
openssl genrsa -out server.key 2048
```

5. Создайте запрос на выпуск сертификата `server.csr`:

```
openssl req -new -key server.key -out server.csr
```

6. Подпишите запрос `server.csr` с помощью корневого сертификата `ca.crt`:

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
server.crt -days 825 -sha256 -extfile server.ext
```

   В результате будет сгенерирован файл TLS-сертификата `server.crt`.

Установка корневого сертификата на компьютере с агентом



В представленной инструкции в качестве примера используется файл корневого сертификата `ca.crt`.

Для TLS-шифрования соединения между агентом и шлюзом необходимо установить корневой сертификат, созданный на этапе Генерация TLS-сертификата, на компьютере агента.

Процесс установки корневого сертификата отличается в зависимости от используемой ОС:

- Установка корневого сертификата в ОС Linux
- Установка корневого сертификата в ОС Windows
- Установка корневого сертификата в macOS

Установка корневого сертификата в ОС Linux



В инструкции для установки сертификата в ОС Linux используется пакет ca-certificates:

- Пакет для Debian-подобных дистрибутивов
- Пакет для RedHat-подобных дистрибутивов

Чтобы установить корневой сертификат в ОС Linux:

- Для Debian-подобных дистрибутивов:
  1. Скопируйте файл корневого сертификата в каталог `/usr/local/share/ca-certificates/`:

     ```
     cp <path_to_cert> /usr/local/share/ca-certificates/
     ```

     Здесь:

     - `<path_to_cert>` — полный путь до файла корневого сертификата `ca.crt`.

  2. Выполните команду:

     ```
     update-ca-certificates
     ```

     Корневой сертификат будет добавлен в список доверенных.

- Для RedHat-подобных дистрибутивов:
  1. Скопируйте файл корневого сертификата в каталог `/etc/pki/ca-trust/source/anchors/`:

     ```
     cp <path_to_cert> /etc/pki/ca-trust/source/anchors/
     ```

     Здесь:

     - `<path_to_cert>` — полный путь до файла корневого сертификата `ca.crt`.

  2. Выполните команду:

     ```
     update-ca-trust
     ```

     Корневой сертификат будет добавлен в список доверенных.

Установка корневого сертификата в ОС Windows

Чтобы установить корневой сертификат в ОС Windows:

1. Откройте диспетчер сертификатов. Для этого выполните следующие действия:
   a. Нажмите сочетание клавиш WIN+R. Откроется окно **Выполнить**.
   b. Введите в окне `certlm.msc` и нажмите **ОК**.
2. Перейдите в раздел **Доверенные корневые центры сертификации** → **Сертификаты** с помощью дерева разделов в диспетчере сертификатов.
3. Выполните команду **Все задачи** → **Импорт** в контекстном меню раздела **Доверенные корневые центры сертификации** → **Сертификаты** в дереве разделов. Откроется мастер импорта сертификатов.
4. Нажмите **Далее** на стартовой странице мастера.
5. Выберите файл корневого сертификата `ca.crt` на странице выбора сертификата и нажмите

Далее.

6. Удостоверьтесь, что на странице выбора хранилища выбран вариант **Поместить все сертификаты в следующее хранилище** и выбрано хранилище **Доверенные корневые центры сертификации**. Нажмите **Далее**.
7. Проверьте корректность указанных параметров на странице завершения импорта и нажмите **Готово**. Корневой сертификат будет добавлен в список доверенных.



Вы также можете установить корневой сертификат следующим способом:

1. Дважды нажмите на файл сертификата `ca.crt`. Откроется окно свойств сертификата.
2. На вкладке **Общее** нажмите на кнопку **Установить сертификат**. Откроется мастер импорта сертификатов.
3. На стартовой странице мастера выберите расположение хранилища **Локальный компьютер** и нажмите **Далее**.
4. На странице выбора хранилища выполните следующие действия:
   a. Выберите вариант **Поместить все сертификаты в следующее хранилище**.
   b. Выберите хранилище **Доверенные корневые центры сертификации**.
   c. Нажмите **Далее**.
5. Проверьте корректность указанных параметров на странице завершения импорта и нажмите **Готово**. Корневой сертификат будет добавлен в список доверенных.

Установка корневого сертификата в macOS

Чтобы установить корневой сертификат в macOS:

1. Откройте приложение **Связка ключей**.
2. Откройте приложение **Finder** и найдите в нем файл корневого сертификата `ca.crt`. Перетащите найденный файл в окно приложения **Связка ключей**.
3. Перейдите на вкладку **Сертификаты** в приложении **Связка ключей**.
4. Дважды нажмите на добавленный корневой сертификат. Откроется окно сведений о сертификате.
5. Раскройте группу параметров **Доверие** и для каждого параметра выберите вариант **Всегда доверять**. Корневой сертификат будет добавлен в список доверенных.



Вы также можете установить корневой сертификат в macOS с помощью команды:

```
sudo security add-trusted-cert -d -r trustRoot -k
/Library/Keychains/System.keychain <path_to_cert>
```

Здесь:

- `<path_to_cert>` — полный путь до файла корневого сертификата `ca.crt`.

Настройка шифрования между агентом и шлюзом

Чтобы настроить шифрование соединения между агентом и шлюзом:

1. При необходимости подключитесь к пространству сателлита.
2. Создайте секрет.
3. Создайте и настройте шлюз.
4. Подключите агент к шлюзу с шифрованием.

Подключение к пространству сателлита

Для подключения агентов к системе используются шлюзы. Шлюзы могут располагаться как в основном экземпляре системы, так и в сателлитах.

Если для соединения с агентом планируется использовать шлюз, расположенный в сателлите, необходимо заранее подключиться к кластеру этого сателлита.

Управление сателлитом осуществляется за счет запускаемых в нем сервисов через веб-интерфейс основного экземпляра системы. Чтобы подключиться к сателлиту, необходимо создать подключение к его пространству. Для этого:

1. Войдите в веб-интерфейс основного экземпляра системы.
2. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**.
3. Нажмите на кнопку **Создать** (┼). Отобразится окно создания подключения.
4. Введите идентификатор пространства сателлита в поле **ID**.
5. Введите название подключения. Название может быть любым.
6. В поле **IP или домен** введите IP-адрес или доменное имя (FQDN) одного из узлов кластера, в котором развернут сателлит.
7. В поле **Порт** введите номер TCP-порта, указанного на этапе **NATS satellite setup** установки сателлита.
8. При необходимости введите описание подключения к пространству.
9. Нажмите на кнопку **Создать**. Система создаст подключение к пространству и отобразит соответствующее уведомление. Подключение отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Пространства**.

У созданного подключения отобразится статус **Подключается**. По завершении подключения статус изменится на **Активен**.



Время на инициализацию подключения зависит загруженности кластеров и каналов связи и может достигать 5—10 минут.

Создание секрета

Чтобы создать секрет с подготовленным TLS-сертификатом:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания секрета.

3. Введите название секрета.
4. При необходимости введите описание секрета.
5. Выберите из выпадающего списка тип секрета **Контейнер PKCS**.
6. Задайте настройки выбранного типа секрета:
   a. Загрузите файл сертификата и файл приватного ключа, подготовленные на этапе Добавление TLS-сертификата на агенте.
   b. Оставьте поле **Пароль для приватного ключа** пустым.



Система не поддерживает использование приватных ключей с паролем для TLS-шифрования соединения.

7. Нажмите на кнопку **Создать**. Система создаст секрет и отобразит соответствующее уведомление. Новый секрет появится в списке раздела **Ресурсы** → **Секреты**.

Создание и настройка шлюза

Чтобы настроить шифрование соединения со шлюзом:

1. Создайте шлюз. При создании задайте следующие параметры:
   a. Выберите пространство, к которому необходимо подключиться:
      - Если необходимо настроить шлюз в сателлите, выберите пространство, созданное на шаге Подключение к пространству сателлита.
      - Если необходимо настроить шлюз в основном экземпляре системы, выберите **Центральное пространство**.
   b. Укажите IP- или FQDN-адрес для подключения к шлюзу.



Адрес шлюза с шифрованием должен совпадать с адресом, указанном в файле TLS-сертификата.

   c. Укажите порт для подключения к шлюзу. Допустимые значения: 30 000—32 767.



Порт должен быть уникальным в рамках инсталляции системы.

   d. Задайте максимальный размер буфера в МБ для хранения событий ИБ.



Эффективный размер буфера будет равен 80% от указанного значения.

   e. Установите переключатель **Включить TLS** в активное положение и выберите секрет, созданный на шаге Создание секрета.
2. Запустите шлюз. Шлюз станет доступен для подключения к нему агентов.

Подключение агента к шлюзу с шифрованием

Чтобы подключить агент к шлюзу с шифрованием соединения:

1. Установите агент, если он еще не установлен.

Если агент уже установлен и подключен к другому кластеру, удалите папку `nats` на хосте агента для его корректной работы. Способ удаления зависит от используемой ОС на хосте.

Linux

```
rm -rf "<nats_path>"
```

Windows

Используйте командную оболочку PowerShell:

```
Remove-Item -Path "<nats_path>" -Recurse
```

macOS

```
rm -rf "<nats_path>"
```

Здесь:

- `<nats_path>` — путь к папке `nats` на хосте.

2. Настройте сетевой доступ к шлюзу по необходимому порту. Для этого задайте настройки подключения к шлюзу в параметре `leafnodes` в секции `nats_server` файла конфигурации `rpoint.conf`.

▼ *Особенности задания параметра* `leafnodes` *с шифрованием*

Для каждого шлюза без TLS-шифрования в поле `url` указывается URL-адрес в формате `nats://<addr>:<port>`.

Для каждого шлюза с TLS-шифрованием адрес указывается в формате `tls://<addr>:<port>`.

Здесь:

- `<addr>` — адрес подключения к шлюзу.
- `<port>` — порт подключения к шлюзу.

Например:

```
"leafnodes": {
  "remotes": [
    {
      "url": "tls://worker01.rpoint.local:32000"
    }
  ]
}
```

3. Запустите агент.

Если настройка была выполнена корректно, агент появится в разделе **Агенты** → **Агенты** веб-интерфейса системы.

Работа с агентами на хосте

На хосте доступны следующие операции над агентами:

- Запуск агента
- Перезапуск агента
- Остановка агента
- Запрос статуса агента
- Запрос версии агента

Способ выполнения каждой операции зависит от используемой ОС на хосте агента.



Для выполнения операций над агентами необходимо обладать правами администратора или суперпользователя (root).

Запуск агента

Linux

- Для дистрибутивов Astra Linux и РОСА Хром Сервер:

```
sudo systemctl start evo-agent.service
```

- Для остальных дистрибутивов:
  - Если переменная окружения `EVO_AGENT_HOME` не задана:

```
sudo /opt/rvision/agent/bin/rpoint-launcher start
```

  - Если переменная окружения `EVO_AGENT_HOME` задана:

```
sudo -E ${EVO_AGENT_HOME}/bin/rpoint-launcher start
```

Windows

Используйте командную оболочку PowerShell:

- Запуск через cmdlet-команду:

```
Start-Service -Name RvisionAgent
```

- Запуск через вызов rpoint-launcher:

- Если переменная RVISION_AGENT_INSTALL_PATH не задана:

```
& 'C:\Program Files (x86)\RVision\Agent\bin\rpoint-launcher.exe' start
```

- Если переменная RVISION_AGENT_INSTALL_PATH задана:

```
& "${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe" start
```

macOS

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher start
```

При запуске агента осуществляется подключение к шлюзу, указанному в конфигурационном файле `rpoint.conf`, а также выполняется сбор его инвентарной информации.

Запуск агента с дополнительными параметрами

- При необходимости вы можете указать адрес подключения к шлюзу явно через параметр `--gateway-url`. Например:

Linux

Для всех дистрибутивов, кроме Astra Linux и РОСА Хром Сервер:

- Если переменная окружения EVO_AGENT_HOME не задана:

```
sudo /opt/rvision/agent/bin/rpoint-launcher start --gateway
-url=nats://<addr>:<port>
```

- Если переменная окружения EVO_AGENT_HOME задана:

```
sudo -E ${EVO_AGENT_HOME}/bin/rpoint-launcher start --gateway
-url=nats://<addr>:<port>
```

Windows

- Если переменная RVISION_AGENT_INSTALL_PATH не задана:

```
& 'C:\Program Files (x86)\RVision\Agent\bin\rpoint-launcher.exe' start
--gateway-url=nats://<addr>:<port>
```

- Если переменная RVISION_AGENT_INSTALL_PATH задана:

```
& "${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe" start --gateway
```

```
-url=nats://<addr>:<port>
```

macOS

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher start --gateway
-url=nats://<addr>:<port>
```

Здесь:

- `<addr>` — адрес подключения к шлюзу.
- `<port>` — порт подключения к шлюзу.

- При необходимости вы можете указать явно, может ли агент переключаться между шлюзами. Подробные сведения о данном механизме приведены в описании параметра `gateway_discovery_enabled` файла конфигурации `rpoint.conf`.

  Чтобы включить или отключить данный механизм, используйте один из следующих параметров:

  - `--gateway-discovery-disable` — отключает механизм переключения между шлюзами и меняет значение параметра `gateway_discovery_enabled` на `false`.
  - `--gateway-discovery-enable` — включает механизм переключения между шлюзами и меняет значение параметра `gateway_discovery_enabled` на `true`.

  Ниже приведены примеры запуска агента без переключения между шлюзами:

  Linux

  Для всех дистрибутивов, кроме Astra Linux и РОСА Хром Сервер:

  ```
  sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher start --gateway-discovery-disable
  ```

  Windows

  ```
  & '<install_path>\bin\rpoint-launcher.exe' start --gateway-discovery-disable
  ```

  macOS

  ```
  sudo /Library/R-Vision/agent/bin/rpoint-launcher start --gateway-discovery
  -disable
  ```

  Здесь:

  - `<install_path>` — путь установки агента.

- При необходимости вы можете указать явно, может ли агент использовать значения по умолчанию для параметров `deny_imports` и `deny_exports`. Подробные сведения об этих

параметрах приведены в описании следующих параметров файла конфигурации `rpoint.conf`:

- `default_deny_exports_enabled`;
- `default_deny_imports_enabled`.

Чтобы указать логику использования значений по умолчанию, используйте следующие параметры:

- `--default-deny-exports-enable` — указывает агенту использовать значения `deny_exports` по умолчанию и меняет значение параметра `default_deny_exports_enabled` на `true`.
- `--default-deny-exports-disable` — указывает агенту не использовать значения `deny_exports` по умолчанию и меняет значение параметра `default_deny_exports_enabled` на `false`.
- `--default-deny-imports-enable` — указывает агенту использовать значения `deny_imports` по умолчанию и меняет значение параметра `default_deny_imports_enabled` на `true`.
- `--default-deny-imports-disable` — указывает агенту не использовать значения `deny_imports` по умолчанию и меняет значение параметра `default_deny_imports_enabled` на `false`.

Ниже приведены примеры запуска агента с отключением использования `deny_exports`:

Linux

Для всех дистрибутивов, кроме Astra Linux и РОСА Хром Сервер:

- Если переменная окружения `EVO_AGENT_HOME` не задана:

```
sudo /opt/rvision/agent/bin/rpoint-launcher start --default-deny-exports
-disable
```

- Если переменная окружения `EVO_AGENT_HOME` задана:

```
sudo -E ${EVO_AGENT_HOME}/bin/rpoint-launcher start --default-deny
-exports-disable
```

Windows

- Если переменная `RVISION_AGENT_INSTALL_PATH` не задана:

```
& 'C:\Program Files (x86)\RVision\Agent\bin\rpoint-launcher.exe' start
--default-deny-exports-disable
```

- Если переменная `RVISION_AGENT_INSTALL_PATH` задана:

```
& "${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe" start --default
-deny-exports-disable
```

macOS

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher start --default-deny
-exports-disable
```

Перезапуск агента

Linux

- Для дистрибутивов Astra Linux и РОСА Хром Сервер:

```
sudo systemctl restart evo-agent.service
```

- Для остальных дистрибутивов:
  - Если переменная окружения `EVO_AGENT_HOME` не задана:

```
/opt/rvision/agent/bin/rpoint-launcher restart
```

  - Если переменная окружения `EVO_AGENT_HOME` задана:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher restart
```

Windows

Используйте командную оболочку PowerShell:

- Перезапуск через cmdlet-команду:

```
Restart-Service -Name RvisionAgent
```

- Перезапуск через вызов rpoint-launcher:
  - Если переменная `RVISION_AGENT_INSTALL_PATH` не задана:

```
& 'C:\Program Files (x86)\RVision\Agent\bin\rpoint-launcher.exe' restart
```

  - Если переменная `RVISION_AGENT_INSTALL_PATH` задана:

```
& "${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe" restart
```

macOS

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher restart
```

При перезапуске агента выполняется сбор его инвентарной информации.

Остановка агента

Linux

- Для дистрибутивов Astra Linux и РОСА Хром Сервер:

```
sudo systemctl stop evo-agent.service
```

- Для остальных дистрибутивов:
  - Если переменная окружения `EVO_AGENT_HOME` не задана:

```
/opt/rvision/agent/bin/rpoint-launcher stop
```

  - Если переменная окружения `EVO_AGENT_HOME` задана:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher stop
```

Windows

Используйте командную оболочку PowerShell:

- Остановка через cmdlet-команду:

```
Stop-Service -Name RvisionAgent
```

- Остановка через вызов rpoint-launcher:
  - Если переменная `RVISION_AGENT_INSTALL_PATH` не задана:

```
& 'C:\Program Files (x86)\RVision\Agent\bin\rpoint-launcher.exe' stop
```

  - Если переменная `RVISION_AGENT_INSTALL_PATH` задана:

```
& "${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe" stop
```

macOS

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher stop
```

Остановка процессов агента

Помимо полной остановки агента, можно также останавливать его конкретные процессы.

Linux

```
sudo pkill -9 <process_name>
```

Windows

Используйте командную оболочку PowerShell:

```
Stop-Process -Name "<process_name>" -Force -ErrorAction SilentlyContinue
```

macOS

```
sudo pkill -9 <process_name>
```

Здесь:

- `<process_name>` — имя процесса, который необходимо остановить.



- Если попытаться остановить любой процесс, кроме rpoint-launcher, то этот процесс будет автоматически перезапущен.
- Если попытаться остановить процесс rpoint-launcher, то дальнейшее поведение процессов будет зависеть от сервиса WatchDog:
  - Если сервис WatchDog активен, то процесс rpoint-launcher будет автоматически перезапущен.
  - Если сервис WatchDog неактивен, то процесс rpoint-launcher не будет перезапущен. В этом случае остальные процессы агента могут быть также успешно остановлены.

Запрос статуса агента

Linux

- Если переменная окружения `EVO_AGENT_HOME` не задана:

```
/opt/rvision/agent/bin/rpoint-launcher status
```

- Если переменная окружения `EVO_AGENT_HOME` задана:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher status
```

Windows

Используйте командную оболочку PowerShell:

- Если переменная `RVISION_AGENT_INSTALL_PATH` не задана:

```
& 'C:\Program Files (x86)\RVision\Agent\bin\rpoint-launcher.exe' status
```

- Если переменная `RVISION_AGENT_INSTALL_PATH` задана:

```
& "${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe" status
```

macOS

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher status
```

В результате выполнения команды будет выведена информация о текущем статусе агента.

Пример информации о статусе агента

```
[{"name":"nats-server","status":"running"},{"name":"vm","reason":"Disabled by
config","status":"not running"}]
```



После установки и запуска агента модуль `nats-server` должен иметь статус `running`, например:

```
[{"name":"nats-server","status":"running"},...]
```

Если статус модуля отличается от `running`, необходимо переустановить агент.

Запрос версии агента

Linux

- Если переменная окружения `EVO_AGENT_HOME` не задана:

```
/opt/rvision/agent/bin/rpoint-launcher --version
```

- Если переменная окружения `EVO_AGENT_HOME` задана:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher --version
```

Windows

Используйте командную оболочку PowerShell:

- Если переменная `RVISION_AGENT_INSTALL_PATH` не задана:

```
& 'C:\Program Files (x86)\RVision\Agent\bin\rpoint-launcher.exe' version
```

- Если переменная `RVISION_AGENT_INSTALL_PATH` задана:

```
& "${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe" version
```

macOS

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher --version
```

В результате выполнения команды будет выведена текущая версия агента.

Обновление агента

Обновление агента отличается в зависимости от используемой операционной системы на хосте:

- Обновление агента в ОС Linux
- Обновление агента в ОС Windows
- Обновление агента в macOS



- Для обновления агента необходимо обладать правами администратора или суперпользователя (root).
- Если при обновлении агента произошел сбой, выполните следующие действия:
  1. Удалите агент.
  2. Установите агент заново.

Обновление агента в ОС Linux

Чтобы обновить агент на хосте с ОС Linux, загрузите и запустите установочный пакет.

Команда запуска зависит от семейства используемого дистрибутива Linux.

Семейство Debian

```
sudo -E dpkg -i <installer_path>
```

При возникновении ошибок зависимости выполните следующую команду для их исправления:

```
sudo apt-get install -f
```

Семейство RedHat

```
sudo -E rpm -U <installer_path>
```

При возникновении ошибок зависимости выполните следующую команду для их исправления:

```
sudo yum install -y
```

Здесь:

- `<installer_path>` — полный путь до установщика.

Обновление агента в ОС Windows



Перед обновлением агента в ОС Windows убедитесь, что у вас установлен Microsoft Visual C++ Redistributable необходимой версии. Поддерживаемые версии Microsoft Visual C++ Redistributable приведены в разделе Технические требования к хосту агента.

Установщик последней версии Microsoft Visual C++ Redistributable можно скачать с официального сайта Microsoft.



Обновление агента в ОС Windows может выполняться из установщиков следующего вида:

- `evo-agent-<build_no>.d-<bit_depth>.msi` — используется для установки только агента. В этом случае необходимо установить компонент Microsoft Visual C++ Redistributable вручную.
- `evo-agent-<build_no>.s-<bit_depth>.msi` — используется для установки агента вместе с Microsoft Visual C++ Redistributable.

Здесь:

- `<build_no>` — номер сборки агента, например: `3.1.0.16940`.
- `<bit_depth>` — разрядность архитектуры процессора на хосте: `x86` или `x64`.

Чтобы обновить агент на хосте с ОС Windows, загрузите и запустите установочный пакет с помощью команды:

```
msiexec /i <installer_path> /L*V '<log_path>'
```

Здесь:

- `<installer_path>` — полный путь до установщика.
- `<log_path>` — полный путь хранения логов.

Обновление агента в macOS

Чтобы обновить агент на хосте с macOS, загрузите и запустите установочный пакет с помощью команды:

```
sudo installer -pkg <installer_path> -target /
```

Здесь:

- `<installer_path>` — полный путь до установщика.

Удаление агента

Удаление установленного агента отличается в зависимости от используемой операционной системы на хосте:

- Удаление агента в ОС Linux
- Удаление агента в ОС Windows
- Удаление агента в macOS



Для удаления агента необходимо обладать правами администратора или суперпользователя (root).

Удаление агента в ОС Linux

Команда удаления агента на хосте с ОС Linux зависит от семейства используемого дистрибутива.

Семейство Debian

```
sudo dpkg -r evo-agent
```

Семейство RedHat

```
sudo rpm -e evo-agent
```

Удаление агента в ОС Windows

На хосте с ОС Windows удалить агент можно через параметры системы или через PowerShell.

Удаление через параметры

1. Откройте меню **Пуск**.
2. Откройте окно **Параметры**.
3. Перейдите в раздел **Приложения** → **Приложения и возможности**. Отобразится список приложений в системе.
4. Найдите в списке приложений evo-agent и нажмите на него. Отобразится панель действий над приложением.
5. Нажмите на кнопку **Удалить** и подтвердите удаление.

Удаление через PowerShell

1. Запустите PowerShell от имени администратора.
2. Выполните команду:

```
Uninstall-Package Evo-Agent
```

Удаление агента в macOS

Чтобы удалить агент на хосте с macOS:

1. Остановите процессы агента:

```
sudo /Library/R-Vision/agent/bin/rpoint-launcher stop
```



Для проверки остановки процессов используйте команду:

```
pgrep -x 'rpoint-analyzer' 'nats' 'rpoint-launcher' || echo
'stopped'
```

Если процессы остановлены успешно, команда выведет `stopped`, в противном случае — отобразит сведения о неостановленных процессах.

2. Удалите приложение агента:

```
sudo rm -rf /Applications/evo-agent.app
```



Для проверки удаления приложения используйте команду:

```
[ ! -d /Applications/evo-agent.app ] && echo "deleted" || echo
"exists"
```

Если приложение удалено успешно, команда выведет `deleted`, в противном случае — `exists`.

3. Если агент был установлен через pkgutil, удалите пакет агента:

```
sudo pkgutil --forget $(pkgutil --pkgs | grep "evo-agent")
```



Для проверки удаления пакета используйте команду:

```
pkgutil --pkgs
```

Если пакет удален успешно, в выводе команды будет отсутствовать `evo-agent`.

4. Удалите папку агента:

```
sudo rm -rf /Library/R-Vision/agent
```



Для проверки удаления папки используйте команду:

```
[ ! -d /Library/R-Vision/agent ] && echo "deleted" || echo "exists"
```

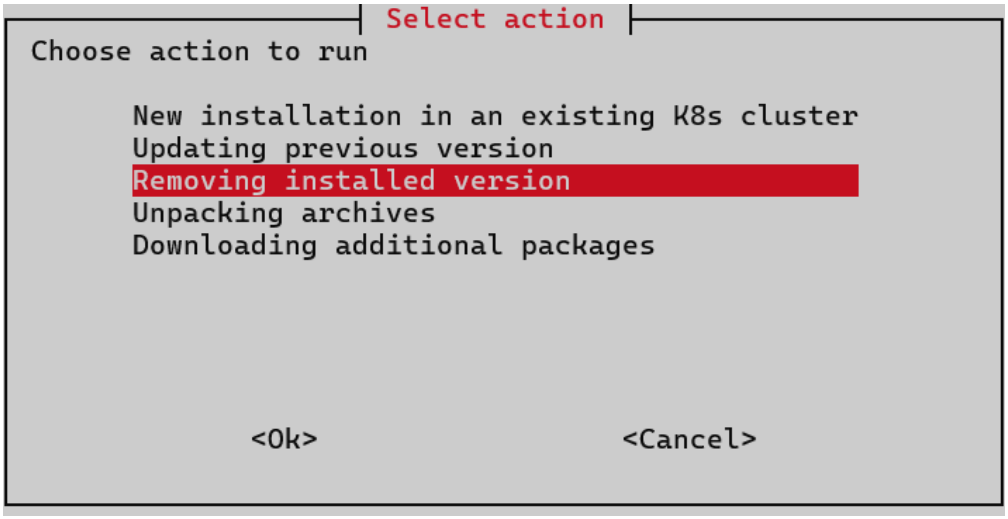
Если папка удалена успешно, команда выведет `deleted`, в противном случае — `exists`.

Удаление системы

В данном разделе описаны действия по удалению основного экземпляра системы или сателлита.

Чтобы удалить основной экземпляр системы или сателлит (далее — "экземпляр системы"):

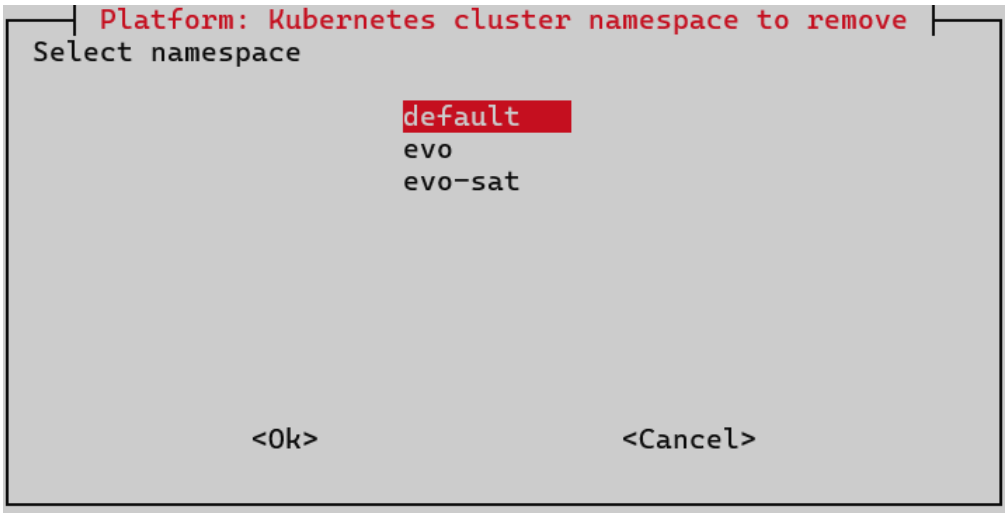
1. Запустите установщик и выберите режим установки **Removing installed version**.



2. Выберите пространство имен (namespace), в котором установлен экземпляр системы.

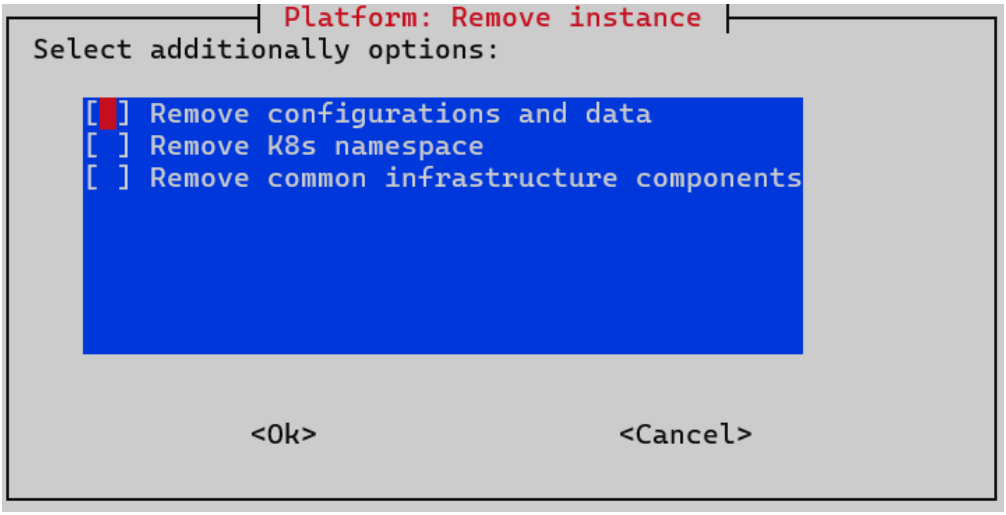


Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.



3. Выберите опции очистки данных и удаления инфраструктурных компонентов:

   - **Remove configurations and data** — удаление данных и конфигураций Kubernetes. Будут удалены следующие данные:
     - конфигурации дискового пространства (Persistent Volume Claims);
     - объекты ConfigMap;
     - секреты.
   - **Remove K8s namespace** — удаление пространства имен, в котором установлен экземпляр системы.
   - **Remove common infrastructure components** — удаление инфраструктурных компонентов, таких как Grafana и Prometheus.





Перемещение по списку опций осуществляется с помощью клавиш со стрелками вверх и вниз. Чтобы выбрать опцию или отменить ее выбор, нажмите на клавишу SPACE.

4. Для подтверждения удаления экземпляра системы введите `yes`.

   Будут удалены все развернутые helm-релизы из целевого пространства имен, а также компоненты, выбранные на предыдущем шаге.

Администрирование системы

В данном разделе представлена информация о настройке и администрировании системы.

Содержание раздела:

- Добавление SSL-сертификата
  - Добавление самоподписанного сертификата в список доверенных
- Работа с кластером
  - Подключение к кластеру
  - Мониторинг кластера
    - Состояние кластера
    - Создание пользовательских дашбордов в Grafana
    - Анализ работоспособности сервисов кластера
    - Анализ работоспособности control plane кластера
    - Анализ состояния сателлита
    - Просмотр журнала событий кластера
    - Просмотр состояния подов кластера
    - Список селекторов системных подов
    - Настройка оповещений
      - Настройка оповещений в Telegram
      - Настройка оповещений по электронной почте
    - Работа с метриками Prometheus
  - Управление кластером
    - Управление узлами кластера
    - Работа с подами audit-manager и collector-audit
    - Работа с подом collector-manager
    - Работа с подом active-list-manager
    - Увеличение размера тома для подов
    - Получение идентификатора сервиса в NATS Service Discovery
    - Работа с объектным хранилищем NATS Object Store
    - Настройка стримов в менеджере пространств
    - Работа с паролями в кластере
    - Изменение доменного имени установленного экземпляра системы
    - Настройка отдельного диска под образы контейнеров
      - Ошибки при запуске пода ingress-nginx-controller
    - Поиск установленных экземпляров системы

- Настройка системы
  - Настройка быстрого доступа к продуктам R-Vision
  - Настройка формата даты
  - Настройка времени хранения оповещений
  - Изменение пула адресов в MetalLB
  - Настройка времени хранения запросов
  - Настройка ограничений размера файла драйвера БД
  - Настройка ограничений размера загружаемых файлов
  - Настройка получения событий по произвольному порту
- Резервное копирование и восстановление компонентов системы
  - Резервное копирование хранилища etcd
  - Резервное копирование без Backup Wrapper
    - Резервное копирование PostgreSQL без Backup Wrapper
    - Резервное копирование ClickHouse без Backup Wrapper
  - Резервное копирование сервисов платформы
  - Резервное копирование внутренних баз данных
  - Работа с Backup Wrapper
    - Настройка PostgreSQL для работы с Backup Wrapper
    - Настройка ClickHouse для работы с Backup Wrapper
    - Настройка многоуровневого хранилища ClickHouse
    - Установка и настройка Backup Wrapper
    - Удаление Backup Wrapper
  - Резервное копирование внешних баз данных
    - Настройка резервного копирования из кластера системы
    - Настройка резервного копирования вне кластера системы
  - Восстановление из резервных копий
    - Восстановление внутренних баз данных
    - Восстановление внешних баз данных
      - Восстановление базы данных PostgreSQL
      - Восстановление базы данных ClickHouse
      - Устранение проблем при восстановлении внешних баз данных
    - Восстановление сервисов платформы
  - Архивирование и восстановление исторических событий
    - Настройка утилиты для архивирования и восстановления исторических событий
    - Работа с утилитой для архивирования исторических событий

- Управление лицензией
- Управление пространствами
  - Работа с пространствами
  - Работа с сервисами
- Управление доступом
  - Тенанты
  - Пользователи
  - Группы пользователей
  - Роли
    - Стандартные роли
  - Политики входа и пароля
    - Настройка времени жизни для токенов доступа
  - LDAP-соединения
    - Настройка сервисной учетной записи ALD Pro
    - Атрибуты Distinguished Name
  - Политики управления доступом к событиям
- Настройка агентов
  - Настройки менеджера агентов
  - Сбор инвентарной информации
  - Сбор событий ИБ на агентах
- Аудит
  - Аудит источников событий
    - Политики аудита источников событий
    - Настройка конвейера для аудита коллектора
  - Аудит системных событий
    - Модель события аудита
    - Создание оповещений о критических событиях
    - Отслеживание действий суперадминистраторов
    - Отслеживание действий над объектами
- Мониторинг RQL-запросов и управление ресурсами

Добавление SSL-сертификата

Для работы системы по протоколу безопасного соединения HTTPS необходимо использовать SSL-сертификат.

SSL-сертификат — это цифровой документ, позволяющий использовать протокол безопасной связи

SSL (Secure Sockets Layer). Применение SSL-сертификата предоставляет следующие гарантии для пользователей:

- Ресурс, на котором расположена система, является подлинным и не принадлежит злоумышленникам.
- Передаваемая информация защищена от злоумышленников благодаря установке зашифрованного соединения между ресурсом и браузером пользователя.

Для работы с системой вы можете добавить доверенный или самоподписанный сертификат.



Добавление сертификатов выполняется после установки системы.

Добавление доверенного SSL-сертификата

Доверенный SSL-сертификат — это сертификат, заверенный подписью удостоверяющего центра (УЦ). Такой сертификат показывает, что ресурс был проверен УЦ на предмет подлинности и безопасности работы.



В представленной инструкции в качестве примера используются файлы со следующими названиями:

- `ssl.conf` — файл конфигурации SSL-сертификата;
- `private.key` — файл приватного ключа SSL-сертификата;
- `request.csr` — файл запроса на выпуск SSL-сертификата;
- `rootCA.crt` — файл корневого сертификата от УЦ;
- `rootCA.key` — файл ключа корневого сертификата от УЦ;
- `domain.crt` — файл доверенного SSL-сертификата.

Чтобы добавить доверенный SSL-сертификат, выполните на master-узле кластера Kubernetes следующие действия:

1. Создайте файл `ssl.conf` с описанием конфигурации SSL-сертификата.
2. Сгенерируйте файлы приватного ключа `private.key` и запроса на выпуск сертификата `request.csr`:

```
openssl req -new -config ssl.conf -keyout private.key -out request.csr
```

3. Если у вас есть корневой сертификат `rootCA.crt` от УЦ и его ключ `rootCA.key`, подпишите с его помощью запрос `request.csr`:

```
openssl x509 -req -CA rootCA.crt -CAkey rootCA.key -in request.csr -out domain.crt
-days 365 -CAcreateserial
```



Если в конфигурации SSL-сертификата указано поле `subjectAltName`, то в конец команды необходимо добавить `-extfile ssl.conf -extensions req_ext`:

```
openssl x509 -req -CA rootCA.crt -CAkey rootCA.key -in request.csr
-out domain.crt -days 365 -CAcreateserial -extfile ssl.conf
-extensions req_ext
```

В результате будет создан доверенный сертификат `domain.crt`.



Если у вас нет корневого сертификата, передайте запрос `request.csr` удостоверяющему центру для создания доверенного сертификата.

4. Создайте секрет в кластере Kubernetes:

```
kubectl create secret tls <secret_name> --namespace <namespace> --key private.key
--cert domain.crt
```

Здесь:

- `<secret_name>` — имя секрета.
- `<namespace>` — пространство имен, в котором установлена система.

5. Внесите изменения в Ingress кластера Kubernetes.

Добавление самоподписанного SSL-сертификата

Самоподписанный SSL-сертификат — это сертификат, подписанный самим лицом, которое ответственно за ресурс. Самоподписанный сертификат не имеет статус доверенного в браузере пользователя до тех пор, пока не будет добавлен в список доверенных на компьютере пользователя вручную. Способ добавления самоподписанного сертификата в список доверенных отличается в зависимости от используемой ОС.



В представленной инструкции в качестве примера используются файлы со следующими названиями:

- `ssl.conf` — файл конфигурации SSL-сертификата;
- `private.key` — файл приватного ключа SSL-сертификата;
- `request.csr` — файл запроса на выпуск SSL-сертификата;
- `domain.crt` — файл самоподписанного SSL-сертификата.

Чтобы добавить самоподписанный SSL-сертификат, выполните на master-узле кластера Kubernetes следующие действия:

1. Создайте файл `ssl.conf` с описанием конфигурации SSL-сертификата.
2. Сгенерируйте файлы приватного ключа `private.key` и запроса на выпуск сертификата `request.csr`:

```
openssl req -new -config ssl.conf -keyout private.key -out request.csr
```

3. Создайте самоподписанный сертификат `domain.crt`:

```
openssl x509 -signkey private.key -in request.csr -req -days 365 -out domain.crt
```



Если в конфигурации SSL-сертификата указано поле `subjectAltName`, то в конец команды необходимо добавить `-extfile ssl.conf -extensions req_ext`:

```
openssl x509 -signkey private.key -in request.csr -req -days 365
-out domain.crt -extfile ssl.conf -extensions req_ext
```

4. Создайте секрет в кластере Kubernetes:

```
kubectl create secret tls <secret_name> --namespace <namespace> --key private.key
--cert domain.crt
```

Здесь:

- `<secret_name>` — имя секрета.
- `<namespace>` — пространство имен, в котором установлена система.

5. Внесите изменения в Ingress кластера Kubernetes.

Конфигурация SSL-сертификата

Конфигурация SSL-сертификата содержит набор обязательных и опциональных полей.

Пример конфигурации SSL-сертификата

```
# Полное доменное имя ресурса системы.
FQDN = <node_fqdn>

# Название организации.
ORGNAME = <org_name>

# (опционально) Альтернативные DNS-имена и IP-адреса, которые должны быть связаны с
сертификатом.
ALTNAMES = DNS:<dns_name_1>,DNS:<dns_name_2>,IP:<ip_address_1>,IP:<ip_address_2>

# Настройки для создания запроса сертификата CSR. Изменения не требуются.
[ req ]
default_bits = 2048
default_md = sha256
prompt = no
encrypt_key = no
distinguished_name = dn
req_extensions = req_ext
```

```
# Настройки уникального имени для сертификата, которые идентифицируют владельца
сертификата.
[ dn ]
C = CH
O = $ORGNAME
CN = $FQDN

# (опционально) Расширения, которые будут добавлены в запрос на сертификат.
[ req_ext ]
subjectAltName = $ALTNAMES
```

Здесь:

- `<node_fqdn>` — полное доменное имя (FQDN) узла.
- `<org_name>` — название организации.
- `<dns_name_N>` — альтернативные DNS-имена.
- `<ip_address_N>` — альтернативные IP-адреса.

Общие поля

Общие поля задаются без привязки к какому-либо разделу полей.

Поле	Описание	Обязательное поле
`FQDN`	Полное доменное имя ресурса, на котором расположена система.	да
`ORGNAME`	Название организации, ответственной за ресурс.	да
`ALTNAMES`	Альтернативные IP-адреса и DNS-имена, которые должны быть связаны с сертификатом. Поле указывается при задании `subjectAltName = $ALTNAMES` в блоке `req_ext`.	нет

Блок полей `req`

В этом блоке задаются настройки для создания запроса сертификата CSR.



Рекомендуется оставлять этот блок без изменений.

Поле	Описание	Обязательное поле
`default_bits`	Указывает длину ключа сертификата в битах. Рекомендуется устанавливать длину 2048 бит.	да
`default_md`	Определяет алгоритм хеширования для подписания сертификата. Рекомендуется использовать алгоритм SHA-256.	да

Поле	Описание	Обязательное поле
`prompt`	Определяет, нужно ли запрашивать ввод данных для полей в блоках `dn` и `req_ext`. Рекомендуется устанавливать значение `no`. В этом случае ввод данных в указанных блоках не запрашивается. Вместо этого используются предопределенные значения в полях `distinguisged_name` и `req_extensions`.	да
`encrypt_key`	Определяет, нужно ли использовать ключ шифрования. Рекомендуется устанавливать значение `no`. В этом случае ключи шифрования не применяются.	да
`distinguished_name`	Указывает на блок полей, где задаются поля X.509. Рекомендуется указывать в поле ссылку на блок `dn`.	да
`req_extensions`	Указывает на блок полей, где задаются расширения. Рекомендуется указывать в поле ссылку на блок `req_ext`.	нет

Блок полей `dn`

Этот блок задает отличительное имя (Distinguished Name), идентифицирующее владельца сертификата.

Поле	Описание	Обязательное поле
`C`	Страна, в которой находится организация.	да
`O`	Название организации, ответственной за ресурс. Рекомендуется использовать значение поля `ORGNAME`.	да
`CN`	Общее имя (Common Name) ресурса, на которое будет выписан сертификат. Рекомендуется использовать значение поля `FQDN`.	да

Блок полей `req_ext`

Этот блок определяет расширения, которые будут добавлены в запрос на сертификат.

Поле	Описание	Обязательное поле
`subjectAltName`	Позволяет сертификату поддерживать несколько доменных имен и IP-адресов. Должен ссылаться на поле `ALTNAMES`.	нет

Изменение Ingress



Для изменения Ingress потребуется список зарезервированных доменов для системы. Рекомендуется подготовить его заранее.

Изменить Ingress можно одним из следующих способов:

- **Изменение Ingress вручную**:

  a. Откройте Ingress в режиме редактирования:

  ```
  kubectl edit ingress evo.core.host-app-host-app
  ```

  b. Добавьте в поле `spec` перед `rules` следующие строки:

  ```
  tls:
    - hosts:
        - <domain_name_1>
        - <domain_name_2>
      secretName: <secret_name>
  ```

  Здесь:

  - `<domain_name_N>` — доменные имена, зарезервированные для системы.
  - `<secret_name>` — имя секрета, созданного на шаге 4.

- **Изменение Ingress через helm-чарты**:

  a. Создайте файл `values.yaml` со следующим содержанием:

  ```
  modules:
    evo.core:
      host-app:
        ingress:
          ingresses:
            host-app:
              tls:
                - hosts:
                    - <domain_name_1>
                    - <domain_name_2>
                  secretName: <secret_name>
  ```

  Здесь:

  - `<domain_name_N>` — доменные имена, зарезервированные для системы.
  - `<secret_name>` — имя секрета, созданного на шаге 4.

  b. Обновите сервис `evo.core.host-app`:

  ```
  helm upgrade -n <namespace> evo.core.host-app /opt/r-vision/common/helm/r-huc
  --reuse-values --values values.yaml
  ```

  Здесь:

- `<namespace>` — пространство имен, в котором установлена система.

Добавление самоподписанного сертификата в список доверенных

Самоподписанный SSL-сертификат — это сертификат, подписанный самим лицом, которое ответственно за ресурс.

По умолчанию самоподписанный сертификат не имеет статус доверенного в браузере пользователя. Чтобы самоподписанный сертификат считался доверенным, его необходимо добавить в список доверенных на компьютере пользователя.

Способ добавления самоподписанного сертификата в список доверенных отличается в зависимости от используемой ОС:

- Добавление сертификата в список доверенных в Windows
- Добавление сертификата в список доверенных в Linux
- Добавление сертификата в список доверенных в macOS

Добавление сертификата в список доверенных в Windows

Чтобы добавить самоподписанный сертификат в доверенные в Windows:

1. Откройте диспетчер сертификатов. Для этого выполните следующие действия:
   a. Нажмите сочетание клавиш WIN+R. Откроется окно **Выполнить**.
   b. Введите в окне `certlm.msc` и нажмите **ОК**.
2. Перейдите в раздел **Доверенные корневые центры сертификации** → **Сертификаты** с помощью дерева разделов в диспетчере сертификатов.
3. Выполните команду **Все задачи** → **Импорт** в контекстном меню раздела **Доверенные корневые центры сертификации** → **Сертификаты** в дереве разделов. Откроется мастер импорта сертификатов.
4. Нажмите **Далее** на стартовой странице мастера.
5. Выберите файл самоподписанного сертификата на странице выбора сертификата и нажмите **Далее**.
6. Удостоверьтесь, что на странице выбора хранилища выбран вариант **Поместить все сертификаты в следующее хранилище** и выбрано хранилище **Доверенные корневые центры сертификации**. Нажмите **Далее**.
7. Проверьте корректность указанных параметров на странице завершения импорта и нажмите **Готово**. Сертификат будет добавлен в список доверенных.

Добавление сертификата в список доверенных в Linux

Чтобы добавить самоподписанный сертификат в доверенные в Linux:

- Для Debian-подобных дистрибутивов:
  1. Перейдите в каталог, в котором содержится файл самоподписанного сертификата.
  2. Скопируйте файл самоподписанного сертификата в каталог `/usr/local/share/ca-`

certificates/:

```
sudo cp domain.crt /usr/local/share/ca-certificates/domain.crt
```

Здесь domain.crt — файл самоподписанного сертификата.

3. Выполните команду:

```
sudo update-ca-certificates
```

Сертификат будет добавлен в список доверенных.

- Для RedHat-подобных дистрибутивов: действия по настройке доверенных сертификатов могут отличаться в зависимости от дистрибутива.



Точный алгоритм настройки и требуемые команды необходимо смотреть в официальной документации для конкретного дистрибутива.

Основной алгоритм действий:

1. Добавьте файл самоподписанного сертификата в каталог /etc/pki/ca-trust/source/anchors/.
2. Выполните команду:

```
update-ca-trust extract
```

Сертификат будет добавлен в список доверенных.

Добавление сертификата в список доверенных в macOS

Чтобы добавить самоподписанный сертификат в доверенные в macOS:

1. Откройте приложение **Связка ключей**.
2. Откройте приложение **Finder** и найдите в нем файл самоподписанного сертификата. Перетащите найденный файл в окно приложения **Связка ключей**.
3. Перейдите на вкладку **Сертификаты** в приложении **Связка ключей**.
4. Дважды нажмите на добавленный самоподписанный сертификат. Откроется окно сведений о сертификате.
5. Раскройте группу параметров **Доверие** и для каждого параметра выберите вариант **Всегда доверять**. Сертификат будет добавлен в список доверенных.

Работа с кластером

Данный раздел предназначен для администраторов и разработчиков, ответственных за управление и мониторинг кластера Kubernetes, в котором развернута система. В нем описаны инструменты и методы контроля состояния кластера и управления его компонентами.

Содержание раздела:

- Подключение к кластеру
- Мониторинг кластера
  - Состояние кластера
  - Создание пользовательских дашбордов в Grafana
  - Анализ работоспособности сервисов кластера
  - Анализ работоспособности control plane кластера
  - Анализ состояния сателлита
  - Просмотр журнала событий кластера
  - Просмотр состояния подов кластера
  - Список селекторов системных подов
  - Настройка оповещений
    - Настройка оповещений в Telegram
    - Настройка оповещений по электронной почте
  - Работа с метриками Prometheus
- Управление кластером
  - Управление узлами кластера
  - Работа с подами audit-manager и collector-audit
  - Работа с подом collector-manager
  - Работа с подом active-list-manager
  - Увеличение размера тома для подов
  - Получение идентификатора сервиса в NATS Service Discovery
  - Работа с объектным хранилищем NATS Object Store
  - Настройка стримов в менеджере пространств
  - Работа с паролями в кластере
  - Изменение доменного имени установленного экземпляра системы
  - Настройка отдельного диска под образы контейнеров
    - Ошибки при запуске пода ingress-nginx-controller
  - Поиск установленных экземпляров системы

Подключение к кластеру

В данном разделе описано, как подключиться к кластеру Kubernetes.



Универсальный способ работы с кластером Kubernetes — это работа через инструмент командной строки kubectl. Но также вы можете подключиться к кластеру Kubernetes с помощью рабочего стола Lens, чтобы работать с кластером в графическом интерфейсе. Подробности приведены в разделе Подключение к

кластеру Kubernetes с помощью Lens.

Подключение к кластеру Kubernetes с помощью командной строки

Чтобы использовать утилиту kubectl для работы с кластером из командной строки, выполните следующие действия:

1. Получите файл `kubeconfig` кластера, в котором установлена система.
2. Установите и настройте утилиту kubectl в зависимости от вашей операционной системы согласно инструкциям в документации Kubernetes:
   - Linux;
   - Windows.
3. Укажите путь к файлу `kubeconfig` в системных переменных окружения:

Linux

В ОС Linux путь к файлу `kubeconfig` по умолчанию — `$HOME/.kube/config`. Вы можете изменить его с помощью переменной окружения `KUBECONFIG`:

```
export KUBECONFIG=<path_to_confg>
```

Здесь:

- `<path_to_config>` — путь к файлу `kubeconfig`.

Windows

В ОС Windows путь к файлу `kubeconfig` по умолчанию — `$%USERPROFILE%\.kube\config`. Вы можете изменить его с помощью переменной окружения `KUBECONFIG`:

```
$env:KUBECONFIG = <path_to_confg>
```

Здесь:

- `<path_to_config>` — путь к файлу `kubeconfig`.



Проверить доступ к кластеру можно, выполнив следующую команду:

```
kubectl get nodes
```

Если доступ к кластеру есть, отобразится список узлов кластера, например:

```
NAME                     STATUS   ROLES           AGE    VERSION
INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
master01.company.local   Ready    control-plane   140d   v1.26.3
```

```
10.10.10.1    <none>        RED OS MUROM (7.3.2)
worker01.company.local   Ready    <none>          140d   v1.26.3
10.10.10.2    <none>        RED OS MUROM (7.3.2)
worker02.company.local   Ready    <none>          140d   v1.26.3
10.10.10.3    <none>        RED OS MUROM (7.3.2)
worker03.company.local   Ready    <none>          140d   v1.26.3
10.10.10.4    <none>        RED OS MUROM (7.3.2)
```

Подключение к кластеру Kubernetes с помощью Lens

В данном разделе описано, как подключиться к кластеру Kubernetes с помощью Lens — рабочего стола для управления кластерами.

О Lens

Lens — это рабочий стол для управления кластерами Kubernetes, предоставляющий интерфейс для мониторинга и администрирования.

Установка Lens

1. Скачайте установщик Lens для вашей операционной системы с официального сайта и выполните установку на ваше локальное устройство, с которого вы планируете подключаться к кластеру.
2. Запустите установленное приложение и войдите в свою учетную запись **Lens ID**. Если у вас уже есть **Lens ID**, то нажмите **Login with Lens ID**, введите свои реквизиты и выполните вход.

После успешной авторизации откроется рабочее меню приложения Lens.

Подключение к кластеру с помощью Lens

Вы можете подключиться к кластеру Kubernetes с помощью Lens следующими способами:

- Указание файлов `kubeconfig`
- Ручное добавление кластера

Указание файлов `kubeconfig`

Lens автоматически обнаруживает кластеры из указанных файлов `kubeconfig`. Доступные кластеры можно найти в разделе **Catalog**.

Чтобы подключиться к кластеру с помощью Lens:

1. Перейдите в раздел **Catalog** → **Clusters**.
2. Нажмите на кнопку **Add Cluster** в правом нижнем углу.
3. Добавьте файл (или файлы) конфигурации кластера или папку с файлами. Выберите один из следующих вариантов:
   - **Sync kubeconfig file(s)** — путь к конкретным файлам `kubeconfig`.
   - **Sync kubeconfig folder(s)** — путь к папкам, содержащим файлы `kubeconfig`.



Файл конфигурации кластера (обычно `kubeconfig`) содержит информацию о

кластере и учетные данные для доступа к нему. Вы можете получить этот файл у администратора кластера или сгенерировать его самостоятельно с помощью командной строки Kubernetes (kubectl).

4. При успешном добавлении кластер отобразится в списке.
5. Выберите строку с нужным кластером в списке для подключения. Откроется окно **Cluster Overview** с подробной информацией о выбранном кластере.

Ручное добавление кластера

Lens поддерживает ручное добавление кластеров через меню **Add from Kubeconfig**. В этом меню можно указать отдельный файл `kubeconfig`, отличный от указанного в меню настроек **Preferences**.



Файл `kubeconfig`, добавленный через меню **Add from Kubeconfig**, не объединяется с файлом `kubeconfig`, указанным в меню **Preferences**.

Для ручного добавления кластера:

1. Перейдите в раздел **Catalog** → **Clusters**.
2. Наведите курсор на кнопку **Add Cluster**.
3. Выберите **Add from kubeconfig** и вставьте пользовательский файл `kubeconfig` в соответствующее поле ввода.

Просмотр текущей конфигурации кластера

Для просмотра текущей конфигурации кластера:

1. В окне кластера нажмите кнопку **New tab** и выберите **Terminal session**.
2. На вкладке **Terminal** выполните следующую команду:

```
kubectl config view --minify --raw
```

При подключении к кластеру убедитесь, что у вас есть действительный и рабочий файл `kubeconfig` для кластера.

Удаление кластера из Lens

Вы можете удалить кластер из списка кластеров как в меню **Catalog**, так и в окне кластера. Удаление кластера из Lens не влияет на сам кластер или его конфигурацию.

Мониторинг кластера

В данном разделе представлена информация о методах и инструментах мониторинга состояния кластера Kubernetes, используемых для обеспечения стабильности и высокой доступности системных ресурсов.

Содержание раздела:

- Состояние кластера

- Создание пользовательских дашбордов в Grafana
- Анализ работоспособности сервисов кластера
- Анализ работоспособности control plane кластера
- Анализ состояния сателлита
- Просмотр журнала событий кластера
- Просмотр состояния подов кластера
- Список селекторов системных подов
- Настройка оповещений
  - Настройка оповещений в Telegram
  - Настройка оповещений по электронной почте
- Работа с метриками Prometheus

Состояние кластера

Данный раздел описывает мониторинг состояния кластера Kubernetes. Наблюдение за кластером осуществляется в разделе **Инструменты** → **Мониторинг** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О мониторинге состояния кластера

Мониторинг состояния кластера Kubernetes в системе осуществляется с помощью инструмента Grafana, предоставляющего визуализацию собранных метрик. Раздел **Инструменты** → **Мониторинг** в веб-интерфейсе системы отображает полную информацию о состоянии кластера, контейнеров, узлов и других компонентов.

Основные возможности раздела:

- Отслеживание состояния контейнеров, в том числе количества запущенных, ожидающих и завершенных контейнеров, а также статистики перезапусков контейнеров за последние 30 минут.
- Мониторинг состояния узлов, отображающий количество доступных и недоступных узлов, а также узлов, не готовых к работе.
- Анализ запросов и доступных ресурсов в кластере, включая поды, CPU и оперативную память.
- Отображение состояния горизонтального автомасштабирования подов (HPA), заданий (Jobs) и ресурсов PVC (Persistent Volume Claim).
- Наблюдение за состоянием контроллеров StatefulSet и перезапусками подов.

Секции мониторинга

В разделе **Инструменты** → **Мониторинг** представлены различные секции, каждая из которых отображает определенные аспекты работы кластера Kubernetes:

- **Cluster** — общие метрики по кластеру, в том числе запросы и доступные ресурсы, такие как поды, CPU и оперативная память.
- **Node** — состояние узлов кластера, в том числе их количество, доступность и готовность к работе.
- **Deployments** — состояние контроллеров Deployment в кластере, в том числе количество реплик и их состояние.
- **Statefulset** — состояние контроллеров StatefulSet, в том числе количество реплик и их состояние.
- **Pods** — состояние подов, в том числе их количество, статусы выполнения и перезапуска.
- **Containers** — состояние контейнеров в кластере.
- **HPA** — состояние горизонтального автомасштабирования подов (HPA), в том числе текущее состояние и пределы масштабирования.
- **Jobs** — статус заданий в кластере, в том числе количество успешно выполненных, активных и проваленных заданий.
- **PVC** — сведения о ресурсах PVC, в том числе их связывание и доступность.

Cluster

Секция **Cluster** содержит метрики по использованию ресурсов кластера:

- **Cluster Pod Requested** — доля запрашиваемых подов.
- **Cluster CPU Requested** — доля запрашиваемых ядер процессора.
- **Cluster Memory Requested** — доля запрашиваемой оперативной памяти.
- **Cluster Pod Capacity** — доступная и используемая емкость подов в кластере.
- **Cluster CPU Capacity** — доступная и используемая емкость CPU в кластере.
- **Cluster Mem Capacity** — доступная и используемая емкость оперативной памяти в кластере.

Node

Секция **Node** предоставляет информацию о состоянии узлов:

- **Number of Nodes** — общее количество узлов в кластере.
- **Nodes Unavailable** — количество недоступных узлов.
- **Node NotReady** — узлы, не готовые к работе.
- **Node Pressure** — узлы, близкие к исчерпанию дискового пространства.



При необходимости расширить объем дискового пространства для узлов обратитесь к разделу Увеличение размера тома для подов.

- **node number** — график количества доступных узлов кластера.



Для получения более полных сведений о состоянии узлов обратитесь к разделу Просмотр журнала событий кластера.

Deployments

Секция **Deployments** показывает информацию о контроллерах Deployment:

- **Deployment Replicas - Up To Date** — общие сведения об актуальных репликах контроллера.
- **Deployment Replicas** — количество реплик контроллера.
- **Deployment Replicas - Updated** — количество недавно обновленных реплик.
- **Deployment Replicas - Unavailable** — количество недоступных реплик.

Statefulset

Секция **Statefulset** показывает состояние контроллеров StatefulSet:

- **Statefulset Replicas - Up To Date** — общие сведения об актуальных контроллерах StatefulSet.
- **Statefulset replicas** — состояние контроллеров StatefulSet.

Pods

Секция **Pods** отображает состояние подов:

- **Pods restart in 30m** — сведения о перезапусках подов за последние 30 минут.
- **Evicted** — сведения о подах, вытесненных для освобождения ресурсов.
- **Pods Running** — количество работающих подов.
- **Pods Succeeded** — количество успешно завершенных подов.
- **Pods Failed** — количество подов, контейнеры которых завершили работу с ошибкой.
- **Pods Pending** — количество подов, ожидающих запуска.
- **Pods Unknown** — количество подов в неизвестном состоянии.



Для получения более полных сведений о состоянии подов обратитесь к разделам Просмотр журнала событий кластера и Просмотр состояния подов кластера.

Containers

Секция **Containers** отображает состояние контейнеров в кластере:

- **Containers Running** — количество запущенных контейнеров.
- **Containers Waiting** — количество контейнеров, ожидающих запуска.
- **Containers Terminated** — количество завершенных контейнеров.
- **Containers Restarts (Last 30 Minutes)** — количество перезапусков контейнеров за последние 30 минут.
- **CPU Cores Requested by Containers** — количество ядер процессора, запрашиваемых контейнерами.
- **Memory Requested by Containers** — объем оперативной памяти, запрашиваемой контейнерами.

HPA

Секция **HPA** отображает состояние горизонтального автомасштабирования:

- **hpa** — общие сведения об HPA.
- **total-hpa-current** — текущее состояние HPA.
- **current==max** — сведения о верхнем пределе HPA.
- **current==min** — сведения о нижнем пределе HPA.

Jobs

Секция **Jobs** показывает состояние заданий:

- **Jobs Succeeded** — количество успешно завершенных заданий.
- **Jobs Active** — количество активных заданий.
- **Jobs Failed** — количество проваленных заданий.
- **Job last success time from now** — время последнего успешного выполнения задания.
- **Job failed** — график проваленных заданий.

PVC

Секция **PVC** отображает информацию о ресурсах PVC:

- **pvc is bound** — количество ресурсов PVC, связанных с постоянными томами.
- **pvc is not bound** — количество ресурсов PVC, не связанных с постоянными томами.
- **pvc** — общие сведения о PVC.

Настройка мониторинга

В разделе **Инструменты** → **Мониторинг** доступны следующие настройки и операции:

- Настройка временного периода
- Обновление мониторинга

Настройка временного периода

Сведения о состоянии кластера отображаются за определенный период времени. По умолчанию информация о кластере отображается за последние 24 часа.

Чтобы изменить временной период мониторинга:

1. Нажмите на кнопку с текущим временным периодом в правом верхнем углу раздела.
2. В открывшейся панели задайте новый временной период. Сведения о состоянии кластера в разделе будут обновлены с учетом указанных настроек.

Обновление мониторинга

Чтобы обновить сведения о состоянии кластера, нажмите на кнопку **Refresh** в правом верхнем углу раздела.

При необходимости вы можете настроить автоматическое обновление сведений мониторинга. Для этого:

1. Нажмите на кнопку в виде стрелки вниз справа от кнопки **Refresh**.
2. Выберите период автоматического обновления.

Создание пользовательских дашбордов в Grafana

Вы можете создавать дашборды в Grafana для мониторинга состояния кластера. Чтобы создать новый дашборд:

1. Войдите в веб-интерфейс Grafana.
2. В разделе **Dashboards** нажмите на кнопку **New** и выберите **New dashboard**.
3. Чтобы перейти к созданию первой панели дашборда, на следующем экране нажмите **Add visualization**.
4. Чтобы добавить новую панель на существующий дашборд, нажмите **Add** на верхней панели инструментов, затем выберите **Visualization** в выпадающем меню.

Примеры создания панелей с дополнительными метриками состояния кластера приведены ниже. Подробная информация доступна в официальной документации Grafana.

Примеры панелей пользовательских метрик в Grafana

Создание графика с метриками из Prometheus

В Grafana можно выводить графики метрик, собранных системой Prometheus. Ниже приведен пример построения графика запросов подов кластера на ресурсы процессора и памяти.

Чтобы добавить на дашборд график метрик из Prometheus, выполните следующие действия:

1. Создайте новую панель на дашборде.
2. Выберите источник данных **Prometheus**. Это можно сделать:
   a. при создании панели в окне **Select data source**;
   b. в нижней части экрана, в выпадающем списке **Data source**.
3. Выберите тип визуализации **Time series** из выпадающего списка в правом верхнем углу экрана.
4. На панели настроек запроса (по умолчанию **A**) выберите метрику из выпадающего списка **Metric**:
   a. Для CPU: `cluster:namespace:pod_cpu:active:kube_pod_container_resource_requests`.
   b. Для оперативной памяти: `cluster:namespace:pod_memory:active:kube_pod_container_resource_requests`.



По умолчанию метрики оперативной памяти отображаются в байтах. Можно преобразовать их в более информативные единицы измерения, например, в мегабайты. Для этого используйте деление на число-делитель:

- На панели настроек запроса нажмите на кнопку **Operations**.
- Выберите **Binary operations** → **Divide by scalar**.

- Укажите делитель в поле **Value**. Например, `1048576` для перевода в мегабайты.

Также в списке **Binary operations** доступны другие арифметические операции над значениями графика.

5. Для фильтрации подов по различным критериям, таким как принадлежность к пространству имен или узлу, используйте меню **Label filters** справа от выпадающего списка **Metric**.
6. Чтобы настроить подписи в легенде графика:
   - Откройте выпадающее меню **Options** в нижней части панели настроек запроса.
   - В выпадающем списке **Legend** выберите опцию **Custom**.
   - Укажите текст подписи, например, `{{pod}} CPU request`, где ключевое слово `{{pod}}` будет заменено на имя пода.



В правой панели редактора доступны дополнительные настройки графика, такие как:

- Изменение названия графика в поле **Title**.
- Настройка сортировки значений в легенде графика. Чтобы ее применить:
  - В разделе **Legend** установите режим отображения (**Mode**) на **Table**.
  - В поле **Values** выберите показатели для отображения в таблице легенды. Например, среднее значение и максимум за выбранный временной период.
  - Выбранные показатели отобразятся в таблице легенды. Для сортировки графиков по ним нажмите на заголовки соответствующих столбцов.

Вывод логов Loki

Grafana позволяет отображать логи, собранные системой Loki. Ниже приведен пример вывода логов шардов ClickHouse.

Чтобы добавить на дашборд панель логов, собранных Loki, выполните следующие действия:

1. Создайте новую панель на дашборде.
2. Выберите источник данных **DS_LOKI**. Это можно сделать:
   a. при создании панели в окне **Select data source**;
   b. в нижней части экрана, в выпадающем списке **Data source**.
3. Выберите тип визуализации **Logs** из меню в правом верхнем углу экрана.
4. На панели настроек запроса (по умолчанию **A**) настройте фильтрацию логов, используя меню **Label filters**. Выберите метки объектов кластера, с которых должны быть получены логи: например, пространство имен, контейнер или под.

   Например, для отображения логов шардов ClickHouse в меню **Label filters** задайте:

   - метку `pod`;
   - оператор `=~`;

- имена подов шардов ClickHouse, например, `clickhouse-shard0-0`, `clickhouse-shard1-0`.



Строки логов можно фильтровать по заданным условиям. Для этого:

- На панели настроек запроса нажмите на кнопку **Operations**.
- Выберите **Line filters** в выпадающем списке.
- Укажите тип фильтра. Доступны фильтры с учетом регистра и регистронезависимые.

Например, чтобы отобразить только сообщения, содержащие текст **error** без учета регистра:

- Выберите тип фильтра **Line contains case insensitive**.
- В поле **Text to find** введите `error`.

5. В меню **Options** в нижней части панели настроек запроса можно задать лимит числа строк логов, выводимых на панель, с помощью поля **Line limit**.



В правой панели редактора доступны дополнительные настройки графика, такие как:

- Изменение названия графика в поле **Title**.
- Настройка отображения логов в разделе **Logs**, например:
  - Добавление метки времени в часовом поясе пользователя с помощью флажка **Time**.
  - Включение переноса строк логов с помощью флажка **Wrap lines**.

Сохранение дашборда

После создания панелей сохраните дашборд, нажав на кнопку **Save dashboard** () в верхней панели инструментов. Сохраненный дашборд будет доступен в разделе **Dashboards** веб-интерфейса Grafana.

Анализ работоспособности сервисов кластера

Для обеспечения работы кластера используются следующие сервисы:

- Dkron;
- ClickHouse;
- NATS Key/Value & Service Discovery

В данном разделе описаны способы проверки работоспособности этих сервисов.



Состояние NATS можно проверить только с помощью командной строки, а всех остальных сервисов — с помощью как командной строки, так и функции проброса портов рабочего стола Lens.

Проверка состояния подов сервисов с помощью командной строки

Просмотреть состояние, логи подов и подробную информацию о них можно с помощью команд, описанных в разделе **Просмотр состояния подов кластера**.

Проверка состояния подов сервисов с помощью проброса портов

Часть сервисов, используемых кластером, имеет панели управления и методы API, с помощью которых можно получить подробную информацию об их состоянии. Подключиться к ним можно с помощью функции проброса портов рабочего стола Lens.

Проверка состояния Dkron

Чтобы получить доступ к панели управления распределенного планировщика задач Dkron:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Pods**.
3. Выберите пространство имен, в котором установлена система, в выпадающем списке **Select Namespace** в правой части экрана. Отобразится таблица подов кластера.
4. В поле поиска подов введите **dkron**. Отобразится под, на котором размещен экземпляр Dkron.
5. Нажмите на под левой кнопкой мыши. В правой части экрана откроется панель с подробной информацией о поде.
6. В разделе **Ports** панели информации найдите ссылку на сервис **http**: например, `http: 8080/TCP`.
7. Нажмите на ссылку. В окне браузера откроется страница панели управления Dkron.

На панели управления Dkron отображается статистика запланированных, выполненных и проваленных заданий, а также текущее состояние узлов Dkron.

Проверка состояния ClickHouse

Чтобы получить доступ к HTTP API СУБД ClickHouse:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Pods**.
3. Выберите пространство имен, в котором установлена система, в выпадающем списке **Select Namespace** в правой части экрана. Отобразится таблица подов кластера.
4. В поле поиска подов введите **clickhouse**. Отобразятся поды, на которых размещены шарды ClickHouse.
5. Нажмите на любой под левой кнопкой мыши. В правой части экрана откроется панель с подробной информацией о поде.
6. В разделе **Ports** панели информации найдите ссылку на сервис **http**: например, `http: 8123/TCP`.
7. Нажмите на ссылку. В окне браузера откроется страница, на которой отобразится сообщение о состоянии ClickHouse.

При нормальной работе ClickHouse, по умолчанию на странице отображается сообщение `Ok`. Сообщение может быть изменено в конфигурации ClickHouse.

Чтобы открыть графический клиент для выполнения запросов к СУБД, добавьте `/play` к URL. Например, `http://localhost:61399/play`.



Для использования клиента введите логин и пароль администратора ClickHouse в поля в правом верхнем углу экрана.

Логин и пароль можно найти в разделе **Environment** информации о поде в Lens. Они указаны в переменных `CLICKHOUSE_ADMIN_USER` и `CLICKHOUSE_ADMIN_PASSWORD`

Анализ работоспособности control plane кластера

В случае возникновения проблем в работе кластера, в особенности после перезапуска, следует проверить состояние его control plane (плоскости управления). Control plane включает следующие компоненты:

- `kube-apiserver` — компонент, который представляет API Kubernetes;
- `etcd` — хранилище конфигураций и состояний ресурсов кластера Kubernetes;
- `kube-scheduler` — компонент, который отслеживает созданные поды без привязанного узла и выбирает узел, на котором они должны работать;
- `kube-controller-manager` — диспетчер контроллеров.

Чтобы проверить состояние control plane, выполните следующие действия в командной строке:

1. Проверьте состояние компонентов control plane с помощью команды:

   ```
   kubectl get --raw='/readyz?verbose'
   ```

   Все компоненты должны иметь состояние `ok`:

   ▼ *Вывод команды* `kubectl get --raw='/readyz?verbose'`, *если кластер функционирует нормально*

   ```
   [+]ping ok
   [+]log ok
   [+]etcd ok
   [+]etcd-readiness ok
   [+]informer-sync ok
   [+]poststarthook/start-kube-apiserver-admission-initializer ok
   [+]poststarthook/generic-apiserver-start-informers ok
   [+]poststarthook/priority-and-fairness-config-consumer ok
   [+]poststarthook/priority-and-fairness-filter ok
   [+]poststarthook/storage-object-count-tracker-hook ok
   [+]poststarthook/start-apiextensions-informers ok
   [+]poststarthook/start-apiextensions-controllers ok
   [+]poststarthook/crd-informer-synced ok
   [+]poststarthook/bootstrap-controller ok
   [+]poststarthook/rbac/bootstrap-roles ok
   [+]poststarthook/scheduling/bootstrap-system-priority-classes ok
   [+]poststarthook/priority-and-fairness-config-producer ok
   [+]poststarthook/start-cluster-authentication-info-controller ok
   ```

```
[+]poststarthook/start-kube-apiserver-identity-lease-controller ok
[+]poststarthook/start-kube-apiserver-identity-lease-garbage-collector ok
[+]poststarthook/start-legacy-token-tracking-controller ok
[+]poststarthook/aggregator-reload-proxy-client-cert ok
[+]poststarthook/start-kube-aggregator-informers ok
[+]poststarthook/apiservice-registration-controller ok
[+]poststarthook/apiservice-status-available-controller ok
[+]poststarthook/kube-apiserver-autoregistration ok
[+]autoregister-completion ok
[+]poststarthook/apiservice-openapi-controller ok
[+]poststarthook/apiservice-openapiv3-controller ok
[+]shutdown ok
readyz check passed
```

Если состояние компонента не соответствует значению `ok`, изучите сообщение об ошибке для этого компонента. Например, оно может быть следующим: `[-]etcd failed: reason withheld`.

Если по сообщению об ошибке не удалось установить ее причину, проверьте состояние подов компонентов, выполнив шаг 2.

2. Для проверки состояния подов компонентов выполните следующую команду:

```
kubectl get pods -n kube-system -o wide | grep <component>
```

Здесь:

- `<component>` — имя компонента из списка выше.

Если компонент работает корректно, в выводе команды под должен иметь статус `Running` (третий столбец). Если статус пода отличается от `Running`, проанализируйте логи пода (шаг 3).



Если команда `kubectl get pods -n kube-system -o wide | grep etcd` не возвращает результатов, это означает, что компонент `etcd` установлен как сервис. Для проверки его состояния используйте следующую команду:

```
systemctl status etcd.service
```

▼ *Пример вывода состояния компонента* `etcd`

```
● etcd.service - etcd
     Loaded: loaded (/etc/systemd/system/etcd.service;
enabled; vendor preset: disabled)
     Active: active (running) since Mon 2024-09-16 14:27:48
MSK; 2 months 13 days ago
   Main PID: 695 (etcd)
      Tasks: 12 (limit: 4652)
     Memory: 403.8M
        CPU: 1d 5h 58min 7.285s
```

```
     CGroup: /system.slice/etcd.service
             └─ 695 /usr/local/bin/etcd

Nov 29 14:07:35 master01-inst.test.coe.rvision.local
etcd[695]: {"level":"info","ts":"2024-11-
29T14:07:35.980+0300","caller":"mvcc/hash.go:137","msg":"stori
ng new hash","hash":669749590,"revision":42723972,"compact-
revision":42723201}
Nov 29 14:12:35 master01-inst.test.coe.rvision.local
etcd[695]: {"level":"info","ts":"2024-11-
29T14:12:35.935+0300","caller":"mvcc/index.go:214","msg":"comp
act tree index","revision":42724748}
...
```

Если сервис `etcd` работает корректно, его статус (`Active`) отображается как `active (running)`. В противном случае обратитесь в службу поддержки, приложив вывод команды `systemctl status etcd.service`.

3. Если статус пода сервиса отличается от `Running`, выполните команду для получения его логов:

```
kubectl logs -n kube-system <pod_name> --all-containers=true
```

Здесь:

- `<pod_name>` — имя пода, полученное на шаге 2.



При большом количестве событий в логе можно сузить вывод, перенаправив результаты команды `kubectl logs` в утилиты для обработки текста. Например, чтобы отобразить только последние несколько событий:

Пример вывода 20 последних событий в журнале с помощью команды `tail`

```
kubectl logs -n kube-system kube-scheduler-master.rvision.local
--all-containers=true | tail -n 20
```

Если логи пода недоступны, можно просмотреть таблицу последних событий (`Events`), связанных с подом. Для этого выполните команду:

```
kubectl -n kube-system describe pod <pod_name>
```

Здесь:

- `<pod_name>` — имя пода, полученное на шаге 2.

4. Если анализ логов или таблицы `Events` не выявил причину неисправности кластера, обратитесь в службу поддержки и приложите вывод команд, выполненных на шагах 1—3.

Анализ состояния сателлита

Если в системе установлены сателлиты, но установленные в них коллекторы или сервисы не работают, можно проанализировать состояние подов кластера сателлита. Для этого выполните следующие действия:

1. Подключитесь к терминалу master-узла кластера сателлита.
2. Выполните следующую команду:

   ```
   kubectl -n <namespace> get pods
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлен сателлит.

Если сателлит работает корректно, все его поды имеют статус `Ready`. Если есть поды, статус которых отличается от `Ready`, или все поды имеют статус `Ready`, но сателлит работает некорректно, обратитесь в службу поддержки по адресу support@rvision.ru.

Просмотр журнала событий кластера

С помощью командной строки

Чтобы просмотреть журнал событий кластера, выполните следующую команду:

```
kubectl -n <namespace> events
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

▼ *Пример журнала событий кластера*

```
LAST SEEN                TYPE      REASON                   OBJECT
MESSAGE
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-14277d5d-b2ac-4c7f-a14e-261aa5345751-correlator1
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-464a69cb-93cf-4111-9907-4445237a24e5-correlator0
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-464a69cb-93cf-4111-9907-4445237a24e5-correlator1
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-464a69cb-93cf-4111-9907-4445237a24e5-correlator2
```

```
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-53027b15-cb78-4019-aacc-2d09232c30ff-correlator0
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-53027b15-cb78-4019-aacc-2d09232c30ff-correlator2
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-7083cdce-3ecf-4968-93ae-e4b7058edc68
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-14277d5d-b2ac-4c7f-a14e-261aa5345751-correlator0
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-a4a12d20-c14e-48aa-9c4e-4977f9a2e6cb-correlator0
waiting for first consumer to be created before binding
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/collector-7766b6c7-8bfa-4aa4-b7c0-351687f737fa
Resource with issues on node(s): worker02.company.local
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/data-clickhouse-shard0-0
Resource with issues on node(s): worker02.company.local
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/dkron
Resource with issues on node(s): worker02.company.local
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/collector-edd90ee5-2bc2-4c2a-9513-ace74b7a5d56
Resource with issues on node(s): worker02.company.local
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/collector-8f80fa2a-63a2-4986-9447-80a7a74a93c6-correlator0
Resource with issues on node(s): worker02.company.local
...
```



При большом количестве событий в списке можно сузить вывод, передав результаты команды `kubectl -n <namespace> events` в утилиты обработки текста. Например, чтобы отобразить только последние несколько событий:

Вывод 20 последних событий в журнале с помощью команды `tail`

```
kubectl -n <namespace> events | tail -n 20
```

С помощью Lens

Чтобы просмотреть журнал событий кластера:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Events**.
3. Выберите пространство имен, в котором установлена система, в выпадающем списке **Select Namespace** в правой части экрана. Отобразится таблица событий кластера.

Просмотр состояния подов кластера

Просмотр статуса и логов подов может понадобиться при решении проблем с системой. Инструкция по поиску подов, на которых расположены сервисы системы, описана в разделе Список селекторов системных подов.

Просмотр состояния подов в списке

Чтобы просмотреть состояние подов кластера, выполните следующую команду:

```
kubectl -n <namespace> get pods
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

▼ *Пример списка подов кластера*

```
NAME                        READY   STATUS      RESTARTS          AGE
clickhouse-shard0-0         1/1     Running     1 (7h53m ago)     2d3h
clickhouse-shard1-0         1/1     Running     0                 2d3h
....
nats-0                      3/3     Running     0                 22d
nats-box-575                1/1     Running     0                 22d
pgbouncer-846c              2/2     Running     1 (27d ago)       35d
postgresql-0                1/1     Running     0                 64d
```

Состояние подов отображается в столбце `STATUS`. Возможные статусы:

- `Pending`: под принят кластером, но один или несколько контейнеров еще не установлены и не подготовлены к запуску. Этот статус включает время, в течение которого под ожидает планирования, а также загружаются по сети образы контейнеров.
- `Running`: под привязан к узлу, а также все его контейнеры созданы. По крайней мере один контейнер работает или находится в процессе запуска или перезапуска.
- `Succeeded`: все контейнеры пода успешно завершили работу и не будут перезапущены.
- `Failed`: все контейнеры пода завершили работу, и по крайней мере один контейнер завершил работу с ошибкой (с кодом, не равным 0, или был остановлен системой) и не будет автоматически перезапущен.
- `ImagePullBackOff`: Kubernetes не может загрузить образ контейнера для пода, что препятствует его созданию.
- `Unknown`: состояние пода не может быть получено. Такое состояние обычно возникает из-за ошибки связи с узлом, в котором под должен работать.



Также состояние подов кластера можно просмотреть на дашборде Grafana в

разделе **Инструменты** → **Мониторинг** веб-интерфейса системы.

Просмотр подробной информации о поде

Чтобы просмотреть подробную информацию о поде, например, состояние, узел, на котором он находится, список его контейнеров, выполните следующую команду:

```
kubectl -n <namespace> describe pods/<pod_name>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<pod_name>` — имя пода, информацию о котором требуется просмотреть.

▼ *Пример вывода команды* `kubectl describe`

```
Name:             test-pod
Namespace:        system
Priority:         0
Service Account:  default
Node:             dev.system.local/10.0.0.13
Start Time:       Wed, 30 Oct 2024 14:49:49 +0300
Labels:           app.kubernetes.io/component=nats
                  app.kubernetes.io/instance=nats
                  app.kubernetes.io/managed-by=Helm
                  app.kubernetes.io/name=nats
                  app.kubernetes.io/version=2.10.20
                  controller-revision-hash=nats-56bcf785dd
                  helm.sh/chart=nats-1.2.4
                  statefulset.kubernetes.io/pod-name=nats-0
Annotations:      checksum/config:
8b3d84c2a7387f68211b5f98f2d6d910a2c0491fcaef10d1187d3d1fac81dea9
                  cni.projectcalico.org/containerID:
d1f8a2f334f8f1c9a47c835ef92f6e95b8cf43c1e085982de2c396c972203e9f
                  cni.projectcalico.org/podIP: 10.0.0.191/32
                  cni.projectcalico.org/podIPs: 10.0.0.191/32
Status:           Running
IP:               10.0.0.191
IPs:
  IP:           10.0.0.191
Controlled By:  StatefulSet/nats
Containers:
  nats:
    Container ID:
containerd://b3638b859a891ae7b6dc897ec50aa234ff09698decf4eccb5adca0a0794f7e24
    Image:         sec/nats-server:2.10.20-scratch
    Image ID:      sec/nats-
server@sha256:d5508b97e054c298bba593e55ec35b7c5a164def88a4ccd9c456d4b598d1a674
    Ports:         4222/TCP, 8222/TCP
    Host Ports:    0/TCP, 0/TCP
```

```
    Args:
      --config
      /etc/nats-config/nats.conf
    State:          Running
      Started:      Wed, 30 Oct 2024 14:49:51 +0300
    Ready:          True
    Restart Count:  0
    Liveness:       http-get http://:monitor/healthz%3Fjs-enabled-only=true
delay=10s timeout=5s period=30s #success=1 #failure=3
    Readiness:      http-get http://:monitor/healthz%3Fjs-server-only=true delay=10s
timeout=5s period=10s #success=1 #failure=3
    Startup:        http-get http://:monitor/healthz delay=10s timeout=5s period=10s
#success=1 #failure=90
    Environment:
      POD_NAME:     nats-0 (v1:metadata.name)
      SERVER_NAME:  $(POD_NAME)
    Mounts:
      /etc/nats-config from config (rw)
      /var/run/nats from pid (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-4fwsp (ro)
Events:           <none>
```



Раздел `Events` в выводе команды `kubectl describe` отображает последние события, связанные с подом. Эти данные полезны для диагностики и устранения проблем в работе пода.

Подробная информация о команде `kubectl describe` доступна в документации Kubernetes.

Просмотр логов пода



Логи пода очищаются при его остановке.

Чтобы просмотреть логи контейнеров, расположенных на поде, выполните команду `kubectl logs`.

Просмотр логов конкретного контейнера:

```
kubectl -n <namespace> logs <pod_name> -c <container_name>
```

Просмотр логов всех контейнеров на поде:

```
kubectl -n <namespace> logs <pod_name> --all-containers=true
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<pod_name>` — имя пода, логи которого требуется просмотреть.

- `<container_name>` — имя контейнера, логи которого требуется просмотреть.



Чтобы логи передавались в потоковом режиме, то есть обновлялись в реальном времени, укажите опцию `-f` команды `kubectl logs`:

```
kubectl -n <namespace> logs -f <pod_name> --all-containers=true
```

С подробной справкой о команде `kubectl logs` можно ознакомиться в документации Kubernetes.

Просмотр потребления подом ресурсов процессора и памяти

Чтобы просмотреть потребление подом ресурсов процессора и памяти, выполните следующую команду:

```
kubectl -n <namespace> top pod <pod_name>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<pod_name>` — имя пода, информацию о котором требуется просмотреть.

▼ *Пример вывода команды* `kubectl top pod`

```
NAME       CPU(cores)   MEMORY(bytes)
test-pod   29m          109Mi
```

Чтобы наблюдать за изменением потребления ресурсов со временем, используйте команду выше как аргумент команды `watch`:

```
watch -n<seconds> kubectl -n <namespace> top pod <pod_name>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлена система.
- `<pod_name>` — имя пода, информацию о котором требуется просмотреть.
- `<seconds>` — число секунд между запусками команды, являющейся аргументом `watch`.

С подробной справкой о команде `kubectl top pod` можно ознакомиться в документации Kubernetes.

Список селекторов системных подов

В данном разделе перечислены сервисы системы, а также селекторы Kubernetes для получения полных имен подов, на которых они размещены. Эти селекторы можно использовать при необходимости просмотреть статус и/или логи подов сервисов.

Поды сервисов в пространстве имен системы

Большая часть подов сервисов находится в пространстве имен, в котором установлена система, и доступна по селектору `app.kubernetes.io/name`.

Каждому поду присваивается случайный идентификатор, добавляемый к имени сервиса, для которого создается под.

Чтобы получить полное имя пода сервиса в пространстве имен системы, выполните следующую команду:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/name=<service_name>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

- `<service_name>` — имя сервиса.

▼ *Пример получения имени пода для сервиса* `evo.streams.collector-manager`

```
kubectl -n system get pods -l app.kubernetes.io/name=evo.streams.collector-manager
```

Пример вывода команды:

```
NAME                                              READY   STATUS    RESTARTS   AGE
evo.streams.collector-manager-59dbb7767-hx8nq     1/1     Running   0          43m
```

В этом примере полное имя пода — `evo.streams.collector-manager-59dbb7767-hx8nq`.

В таблице ниже приведены имена и описания сервисов, к подам которых можно обратиться, используя вышеуказанный шаблон команды.



Команды для получения полных имен подов сервисов, которые не находятся в пространстве имен системы или недоступны по селектору `app.kubernetes.io/name`, содержатся в разделе Поды других сервисов.

Имя сервиса	Функции
`clickhouse`	СУБД ClickHouse.
`dkron`	Запуск задач по расписанию в распределенной среде.
`evo.core.access`	Авторизация в системе, управление пользователями.
`evo.core.bff`	Организация взаимодействия с основными модулями системы через интерфейс.

Имя сервиса	Функции
evo.core.cron-service	Запуск задач по расписанию.
evo.core.host-app	Отображение интерфейса системы.
evo.core.ldap-client	Получение данных из каталогов доменов.
evo.core.license-service	Управление лицензией.
evo.core.mf-feed	Конфигурация микрофронтендов.
evo.core.smtp-client	Отправка почтовых уведомлений.
evo.ep.bff	Организация взаимодействия с модулем управления агентами через интерфейс.
evo.ep.manager	Управление агентами, группами агентов и шлюзами.
evo.space.bff	Организация взаимодействия с модулем управления пространствами и сервисами через интерфейс.
evo.streams.active-list-manager	Управление активными списками.
evo.streams.alert-manager	Управление оповещениями.
evo.streams.audit-manager	Аудит.
evo.streams.bff	Организация взаимодействия с модулем управления событиями ИБ через интерфейс.
evo.streams.collector-manager	Управление коллекторами.
evo.streams.comment-manager	Управление комментариями.
evo.streams.dashboard-manager	Управление дашбордами.
evo.streams.documentation	Отображение пользовательской документации.
evo.streams.driver-manager	Управление драйверами внешних баз данных.
evo.streams.entity-lookup-manager	Глобальный поиск сущностей.
evo.streams.entity-relation-manager	Управление связями между сущностями.
evo.streams.event-model-manager	Управление моделями событий.
evo.streams.event-storage-manager	Управление хранилищами событий.
evo.streams.expertise-client	Выполнение операций над элементами экспертизы разных типов.
evo.streams.expertise-manager	Управление элементами экспертизы.
evo.streams.file-manager	Управление файлами.
evo.streams.integration-manager	Управление интеграциями.
evo.streams.log-manager	Управление логами.
evo.streams.metric-manager	Управление метриками.
evo.streams.pipeline-verification	Валидация конвейеров.
evo.streams.query-log-manager	Управление логами выполнения запросов.
evo.streams.search	Поиск событий.

Имя сервиса	Функции
`evo.streams.secret-manager`	Управление секретами.
`evo.streams.settings-manager`	Управление настройками и политиками доступа к событиям.
`evo.streams.template-manager`	Управление шаблонами: • точек входа на конвейерах; • конечных точек на конвейерах; • фильтров на конвейерах; • VRL-трансформаций на конвейерах; • шин событий в коллекторах; • настроек таблиц; • почтовых сообщений; • шаблонов инцидентов для интеграции с R-Vision SOAR; • VRL-сниппетов.
`log-manager`	Управление логами менеджера пространств и сервисов.
`metric-manager`	Предоставление метрик пространств и сервисов из Prometheus.
`nats`	Обнаружение сервисов, хранилище "ключ — значение".
`postgresql`	СУБД PostgreSQL.
`service-discovery`	Регистрация и контроль состояния сторонних сервисов в пространстве.
`space-controller`	Управление сервисами в пространствах.
`space-manager`	Управление пространствами.
`vector-aggregator`	Центральный агрегатор телеметрии, читает данные из NATS и записывает их в сервисы телеметрии, такие как Prometheus и Loki.
`vector-collector`	Коллектор телеметрии в пространстве, принимает данные от `vector-daemon` и sidecar-контейнеров Vector и записывает их в NATS.
`vector-daemon`	Демон Vector, который собирает логи и метрики и отправляет их в `vector-collector`.
`wl-<collector_id>`	Сервис коллектора. Для каждого включенного коллектора создается отдельный под. Идентификатор коллектора `<collector_id>` можно найти в поле **ID** карточки коллектора в разделе системы **Ресурсы** → **Коллекторы**.

Поды других сервисов

Поды некоторых сервисов находятся вне пространства имен, в котором установлена система, или доступны по селектору, отличному от `app.kubernetes.io/name`. Команды для получения полных имен этих подов представлены в таблице ниже.

Имя сервиса	Функции	Команда для получения полного имени пода
`alertmanager`	Получение оповещений от Prometheus и передача уведомлений.	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=alertmanager`
`grafana`	Визуализация данных мониторинга для отображения дашбордов, графиков и диаграмм.	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=grafana`
`ingress-nginx`	Реверсивный прокси для маршрутизации внешних запросов к сервисам приложений.	`kubectl -n ingress-nginx get pods -l app.kubernetes.io/name=ingress-nginx`
`kube-state-metrics`	Мониторинг состояния кластера через Prometheus и Grafana.	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=kube-state-metrics`
`loki`	Централизованный сбор и хранение логов сервисов.	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=loki`
`metrics-server`	Реализация Kubernetes Metrics API, предоставляющего элементам кластера информацию об использовании вычислительных ресурсов.	`kubectl -n kube-system get pods -l app.kubernetes.io/name=metrics-server`
`minio`	Хранение файлов конфигураций, загруженных в систему файлов и логов сервисов и инфраструктуры.	`kubectl -n <namespace> get pods -l app=minio` Здесь: • `<namespace>` — название пространства имен, в котором установлена система.
`prometheus`	Сбор, хранение и отображение исторических данных мониторинга.	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=prometheus`

Имя сервиса	Функции	Команда для получения полного имени пода
`prometheus-nats-exporter`	Сбор метрик NATS и их передача в Prometheus.	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=prometheus-nats-exporter`
`prometheus-operator`	Автоматизация развертывания и обновления стека мониторинга в Kubernetes.	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=kube-prometheus-stack-prometheus-operator`
`promtail`	Централизованный сбор и хранение логов приложений.	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=promtail`

Настройка оповещений

О настройке оповещений

В состав системы входит инструмент визуализации Grafana, который используется для мониторинга метрик, собранных с узлов кластера.

Процесс настройки оповещений в Grafana включает следующие этапы:

1. Авторизация в веб-интерфейсе Grafana.
2. Создание правила оповещений с условиями срабатывания.
3. Настройка точки контакта для отправки оповещений (например, Email, Slack, Telegram).
4. Настройка политики оповещений для маршрутизации оповещений на основе их содержания.

Более подробную информацию о настройке можно найти в официальной документации Grafana.

Вход в веб-интерфейс Grafana

Форма авторизации в сервисе Grafana по умолчанию скрыта из веб-интерфейса системы. Вы можете получить к ней доступ следующими способами:

- Локальное подключение к Grafana
- Предоставление доступа к Grafana для всех пользователей



Предоставление доступа к Grafana для всех пользователей повышает риск взаимодействия потенциального нарушителя с системой. В целях безопасности рекомендуется подключаться к сервису Grafana локально.

Локальное подключение к Grafana

Чтобы подключиться к сервису Grafana локально:

1. Подключитесь к кластеру.

2. Получите конфигурацию сервиса `kube-prometheus-stack-grafana`, выполнив команду:

```
kubectl describe svc kube-prometheus-stack-grafana -n kube-prometheus-stack
```

3. Узнайте номер порта, по которому установлено соединение с Grafana, в поле `port` в блоке `spec.ports`.

▼ *Пример блока `ports` конфигурации сервиса `kube-prometheus-stack-grafana`*

```
ports:
  - name: http-web
    protocol: TCP
    port: 80
    targetPort: 3000
```

В данном примере установлено соединение с Grafana по порту 80.

4. Настройте переадресацию порта сервиса `kube-prometheus-stack-grafana`, выполнив команду:

```
kubectl port-forward svc/kube-prometheus-stack-grafana 8080:<port> -n kube-
prometheus-stack
```

Здесь:

- `<port>` — номер порта, определенный на предыдущем шаге.

5. Перейдите по адресу `localhost:8080/grafana` в браузере.
6. В открывшемся окне авторизации введите логин и пароль для доступа к Grafana.

Предоставление доступа к Grafana для всех пользователей



Предоставление доступа к Grafana для всех пользователей повышает риск взаимодействия потенциального нарушителя с системой. В целях безопасности рекомендуется подключаться к сервису Grafana локально.

Чтобы предоставить доступ к Grafana для всех пользователей:

1. Подключитесь к кластеру.
2. Предоставьте доступ к форме авторизации Grafana, выполнив команду:

```
kubectl -n kube-prometheus-stack get ingress kube-prometheus-stack-grafana -o json
|
jq '
  .spec.rules |= map(
    .http.paths += [
      {
        "backend": {
```

```
          "service": {
            "name": "kube-prometheus-stack-grafana",
            "port": {
              "number": 80
            }
          }
        },
        "path": "/grafana",
        "pathType": "Prefix"
      }
    ]
  )
' |
kubectl apply -f -
```

3. Перейдите по адресу `https://<domain_name>/grafana` в браузере.

   Здесь:

   - `<domain_name>` — доменное имя системы.

4. В открывшемся окне авторизации введите логин и пароль для доступа к Grafana.

Учетные данные Grafana

Чтобы получить учетные данные для авторизации в сервисе Grafana:

1. Подключитесь к кластеру.
2. Получите логин пользователя, выполнив команду:

```
kubectl get secret -n kube-prometheus-stack kube-prometheus-stack-grafana -o
jsonpath="{.data.admin-user}"
```

3. Получите пароль пользователя, выполнив команду:

```
kubectl get secret -n kube-prometheus-stack kube-prometheus-stack-grafana -o
jsonpath="{.data.admin-password}"
```



Полученные логин и пароль зашифрованы в кодировке base64.

4. Расшифруйте логин и пароль, например, используя команду:

Windows

Используйте командную оболочку PowerShell:

```
[Text.Encoding]::UTF8.GetString([Convert]::FromBase64String("<login_or_passw
```

```
ord>"))
```

Linux

```
echo "<login_or_password>" | base64 --decode
```

Здесь:

- `<login_or_password>` — логин или пароль, которые вы получили на предыдущем шаге.

Добавление правила оповещения

Правило оповещения в Grafana предназначено для автоматического мониторинга данных и создания оповещений при достижении определенных условий. Оно позволяет задавать параметры для регулярной проверки метрик или логов и инициировать оповещение в случае отклонения от заданных норм.

Выполните следующие шаги для создания правила оповещения:

1. Перейдите в меню **Alerting** → **Alert rules** и выберите опцию для создания нового правила оповещения.
2. В разделе **Set a query and alert condition** выполните следующие действия:
   a. Выберите источник данных (например, **Prometheus**).
   b. Установите временной диапазон для анализа данных (например, `now-10m to now`).
   c. Нажмите **Make this the alert condition** для создания условия срабатывания оповещения.
   d. В разделе **Operations** укажите функции, которые будут применяться к метрикам (например, `Last` и `Strict`).
   e. Установите пороговое значение для срабатывания оповещения (например, `IS ABOVE 0`).
3. Перейдите к разделу **Alert evaluation behavior**:
   a. Установите частоту проверки условия в поле **Evaluate every** (например, `1m`).
   b. Укажите продолжительность, в течение которой условие должно оставаться истинным, чтобы оповещение сработало, в поле **For** (например, `5m`).
4. В разделе **Add details for your alert**:
   a. Укажите название правила оповещения в поле **Rule name**.
   b. Выберите папку и группу для хранения правила в полях **Folder** и **Group**.
   c. Добавьте описание и аннотации к оповещению в полях **Summary**, **Description** и **Runbook URL**. Используйте соответствующие поля для добавления идентификаторов оповещения, панели, или дашборда.
5. В разделе **Notifications**:
   a. Настройте обработку оповещений. Grafana связывает оповещения с точками контактов, используя метки.
   b. Чтобы добавить пользовательские метки, выберите **Custom Labels** и укажите ключ и

значение метки. Метки позволяют маршрутизировать оповещения по определенным правилам.

c. Для добавления новой метки нажмите **Add label** и заполните поля **Choose key** и **Choose value**.

d. Grafana использует корневой маршрут (**Root route**) по умолчанию для всех оповещений. Если не указаны пользовательские метки, оповещения будут направлены по этому маршруту.

6. Нажмите **Save and exit** для сохранения правила оповещения.

Настройка точек контакта

Используйте точки контакта Grafana для выбора предпочитаемого канала связи для получения оповещений при срабатывании правил оповещений. Вы можете добавлять, редактировать, удалять и тестировать точки контакта.

Точки контакта

На вкладке **Contact Points** вы можете:

- Искать точки контакта по имени и типу.
- Просматривать все существующие точки контакта.
- Добавлять новые точки контакта и редактировать существующие.
- Удалять точки контакта. Обратите внимание, что вы не можете удалить точки контакта, которые используются политикой оповещений. Для этого необходимо либо удалить политику оповещений, либо обновить ее, указав другую точку контакта.

Добавление точки контакта

Выполните следующие шаги для добавления точки контакта:

1. В меню слева перейдите в раздел **Alerting** → **Contact points**
2. В выпадающем меню **Choose Alertmanager** выберите менеджер оповещений. По умолчанию выбран **Grafana**.
3. На вкладке **Contact Points** нажмите **New contact point**.
4. Введите имя точки контакта.
5. Выберите тип точки в поле **Contact point type** и заполните обязательные поля. Например, если вы выбрали **Email**, введите адреса электронной почты через точку с запятой.
6. Некоторые интеграции точек контакта, такие как **Email** или **Webhook**, имеют дополнительные настройки. В разделе **Optional settings** укажите дополнительные параметры для выбранной интеграции точки контакта. Например, можно задать формат заголовка и текст оповещения.
7. В разделе **Notification settings** по желанию выберите **Disable resolved message**, если вы не хотите получать оповещение, когда оповещение будет разрешено.
8. Чтобы добавить еще одну интеграцию точки контакта, нажмите **New contact point type** и повторите шаги 6-8.
9. Сохраните изменения.

Тестирование точки контакта

Выполните следующие шаги для тестирования точки контакта:

1. В меню слева перейдите в раздел **Alerting** → **Contact points**, чтобы просмотреть список существующих точек контакта.
2. На вкладке **Contact Points** найдите точку контакта, которую хотите протестировать, затем нажмите **Edit**. Вы также можете создать новую точку контакта, если это необходимо.
3. Нажмите **Test**, чтобы открыть диалоговое окно тестирования точки контакта.
4. Выберите, отправить ли предварительно определенное тестовое оповещение или выберите **Custom**, чтобы добавить свои собственные аннотации и метки для включения в оповещение.
5. Нажмите **Send test notification**, чтобы инициировать оповещение.

Настройка политики оповещений

Политики оповещений в Grafana управляют маршрутизацией оповещений к точкам контакта. Политики имеют древовидную структуру, где каждая политика может содержать одну или несколько дочерних политик, а также может включать правила для сопоставления меток оповещений. Оповещения оцениваются по умолчанию с помощью корневой политики, после чего последовательно оцениваются каждой дочерней политикой.

Редактирование корневой политики оповещений

Для настройки корневой политики оповещений в Grafana:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Убедитесь, что в выпадающем меню **Choose Alertmanager** выбран **Grafana**.
3. В секции **Root policy - default for all alerts** нажмите **Edit**.
4. В поле **Default contact point** выберите точку контакта, куда будут направляться все оповещения, если не установлены другие правила маршрутизации.
5. В секции **Group by** выберите метки, по которым будут группироваться оповещения. Например, можно выбрать метки `grafana_folder` и `alertname`, чтобы группировать оповещения по этим критериям.
6. Разверните секцию **Timing options** и настройте параметры времени:
    a. **Group wait** — время ожидания перед отправкой первого оповещения для новой группы. По умолчанию 30 секунд.
    b. **Group interval** — минимальный интервал времени между отправкой оповещений в группе. По умолчанию 5 минут.
    c. **Repeat interval** — минимальный интервал времени для повторной отправки оповещения, если в группе не было добавлено новых оповещений. По умолчанию 4 часа.
7. Нажмите **Save**, чтобы сохранить изменения.

Добавление дочерней политики

Для создания дочерней политики, которая будет использоваться для определенных меток:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Убедитесь, что в выпадающем меню **Choose Alertmanager** выбран **Grafana**.
3. В секции **Specific routing** нажмите **New specific policy**.
4. В секции **Matching labels** добавьте одно или несколько правил для соответствия меткам оповещений.
5. В поле **Contact point** выберите точку контакта, куда будут отправляться оповещения, если они соответствуют текущей политике.
6. При необходимости включите опцию **Continue matching subsequent sibling nodes**, чтобы продолжить обработку других дочерних политик после совпадения с текущей.
7. При необходимости включите опцию **Override grouping**, если требуется изменить группировку оповещений.
8. При необходимости включите опцию **Override general timings**, чтобы переопределить временные параметры, настроенные в корневой политике.
9. Нажмите **Save policy**, чтобы сохранить настройки.

Добавление соседней политики

Для создания соседней политики рядом с существующей дочерней политикой:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Найдите дочернюю политику, для которой требуется создать соседнюю политику.
3. Нажмите **Add new policy** и выберите **New sibling above** или **New sibling below**.
4. Настройте новую политику аналогично дочерней, задав метки, точку контакта и дополнительные параметры.
5. Нажмите **Save policy**, чтобы сохранить настройки.

Поиск политик

Для поиска политик в дереве настроек используйте следующие параметры:

- **Label matchers** — для поиска по меткам.
- **Contact Points** — для поиска по точкам контакта.

Для поиска по точке контакта выберите ее из выпадающего меню **Search by contact point**. Политики, использующие эту точку контакта, будут выделены в интерфейсе.

Для поиска по меткам введите соответствующие значения в поле **Search by matchers**. Можно комбинировать несколько меток, разделяя их запятой.

Настройка времени отключения оповещений

Mute timings используются для указания временных интервалов, в течение которых оповещения должны быть отключены. Эти настройки не наследуются от родительской политики и должны быть настроены на каждом уровне.

Для добавления временных интервалов отключения:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Нажмите **Add mute timing**.
3. Укажите временные интервалы, в которые оповещения должны быть отключены.
4. Нажмите **Save timing**, чтобы сохранить настройки.

Теперь политики оповещений настроены и будут корректно обрабатывать оповещения в зависимости от заданных параметров.

Настройка оповещений в Telegram

Для настройки оповещений в Telegram в Grafana необходимо выполнить следующие шаги:

1. Создать и настроить Telegram-бота.
2. Подключить Telegram-бота к Grafana.
3. Настроить точку контакта для отправки оповещений через Telegram.
4. Добавить созданную точку контакта в политику уведомлений для маршрутизации оповещений.

Настройка Telegram-бота

Для создания и настройки Telegram-бота:

1. Откройте приложение Telegram на вашем устройстве.
2. Найдите бота с именем **@BotFather**.
3. Отправьте команду **/newbot**.
4. Введите название бота, например, **System Notifier**.
5. Введите имя пользователя для бота. Имя должно оканчиваться на "bot" или "_bot", например, **system_notify_bot**.
6. Сохраните значение токена, предоставленное в ответе.
7. Создайте групповую беседу или откройте уже существующую в приложении Telegram.
8. Добавьте созданного бота в эту группу, найдя его по имени пользователя.
9. Скопируйте идентификатор чата из URL-адреса в адресной строке браузера. Адрес должен выглядеть примерно так: `https://web.telegram.org/a/#-4266605675`. Идентификатор чата — это последовательность чисел, следующих за символом `#`. Например, `-4266605675`.

Настройка точки контакта в Grafana

После настройки Telegram-бота, выполните следующие шаги для его интеграции с Grafana:

1. Выполните вход в веб-интерфейс Grafana.
2. Перейдите в **Alerting** → **Contact points**.
3. Нажмите **New contact point**, чтобы создать новую точку контакта.
4. Введите имя точки контакта в поле **Name**.
5. В списке **Contact point type** выберите **Telegram**.
6. Укажите настройки Telegram:

- **BOT API Token** — токен Telegram-бота.
- **Chat ID** — ID чата в Telegram.

7. В разделе **Optional Telegram settings** (опционально):
   a. Включите опцию **Disable resolved message**, чтобы не получать сообщения о разрешении оповещения.
   b. Включите опцию **Include image**, чтобы добавить изображение в оповещение.
8. Нажмите **Test** для отправки тестового сообщения и проверки работы интеграции.
9. Нажмите **Save contact point**, чтобы сохранить настройки.

Добавление точки контакта в политику уведомлений

После создания точки контакта для оповещений через Telegram необходимо добавить ее в политику уведомлений.

Для добавления точки контакта в политику уведомлений выполните следующие шаги:

1. Перейдите в **Alerting** → **Notification policies**.
2. Убедитесь, что в выпадающем меню **Choose Alertmanager** выбран **Grafana**.
3. В секции **Root policy - default for all alerts** нажмите **Edit**.
4. В поле **Default contact point** выберите созданную ранее точку контакта для оповещений через Telegram.
5. При необходимости в разделе **Group by** выберите метки, по которым будут группироваться оповещения.
6. При необходимости настройте параметры времени в разделе **Timing options**:
   a. **Group wait** — время ожидания перед отправкой первого уведомления для новой группы.
   b. **Group interval** — минимальный интервал времени между отправкой уведомлений в группе.
   c. **Repeat interval** — минимальный интервал времени для повторной отправки уведомления, если в группе не было добавлено новых оповещений.
7. Нажмите **Save**, чтобы сохранить изменения.

После выполнения этих шагов все оповещения, соответствующие установленным критериям, будут отправляться на указанную точку контакта через Telegram.

Настройка оповещений по электронной почте

Для настройки оповещений по электронной почте в Grafana необходимо выполнить следующие шаги:

1. Настроить SMTP-сервер в конфигурационном файле Grafana.
2. Создать и настроить точку контакта для отправки оповещений по электронной почте через веб-интерфейс Grafana.
3. Добавить созданную точку контакта в политику уведомлений для маршрутизации оповещений.

Конфигурация SMTP-сервера

Прежде чем настраивать канал оповещений по электронной почте, необходимо настроить SMTP-сервер в конфигурационном файле Grafana.

Чтобы настроить SMTP-сервер в Grafana:

Через командную строку

1. Подключитесь к кластеру.
2. Откройте для редактирования ConfigMap `kube-prometheus-stack-grafana`, выполнив команду:

```
kubectl edit configmap kube-prometheus-stack-grafana -n kube-prometheus-
stack
```

3. Добавьте следующие строки в блок `data.grafana.ini`:

```
[smtp]
enabled = true
host = <host>
user = <username>
password = <password>
from_address = <sender_address>
from_name = <sender_name>
skip_verify = <skip_verify>
```

Здесь:

- `<host>` — адрес и порт вашего SMTP-сервера.
- `<username>` — логин пользователя, если требуется.
- `<password>` — пароль пользователя, если требуется.
- `<sender_address>` — адрес отправителя.
- `<sender_name>` — имя отправителя.
- `<skip_verify>` — логическое значение, которое позволяет выключить проверку SSL- или TLS-сертификата при подключении к почтовому серверу:
  - `true` — проверка сертификата выключена.
  - `false` — проверка сертификата включена.



Отключение проверки сертификата (`skip_verify = true`) в рабочем окружении может быть небезопасным. Рекомендуется устанавливать для `skip_verify` значение `true` только при тестировании.

▼ *Пример настроек SMTP*

```
[smtp]
enabled = true
host = smtp.example.com:587
user = yourusername
password = yourpassword
from_address = alert@example.com
```

```
from_name = Grafana Alerts
skip_verify = false
```

4. Сохраните изменения в ConfigMap `kube-prometheus-stack-grafana`.
5. Перезапустите контроллер Deployment `kube-prometheus-stack-grafana`, выполнив команду:

   ```
   kubectl -n kube-prometheus-stack rollout restart deployment kube-prometheus-stack-grafana
   ```

   Дождитесь перезапуска контроллера Deployment `kube-prometheus-stack-grafana`.

Через Lens

1. Подключитесь к кластеру.
2. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
3. Выберите ConfigMap `kube-prometheus-stack-grafana` из списка.
4. В открывшейся панели справа добавьте в блок `grafana.ini` следующие строки:

   ```
   [smtp]
   enabled = true
   host = <host>
   user = <username>
   password = <password>
   from_address = <sender_address>
   from_name = <sender_name>
   skip_verify = <skip_verify>
   ```

   Здесь:

   - `<host>` — адрес и порт вашего SMTP-сервера.
   - `<username>` — логин пользователя, если требуется.
   - `<password>` — пароль пользователя, если требуется.
   - `<sender_address>` — адрес отправителя.
   - `<sender_name>` — имя отправителя.
   - `<skip_verify>` — логическое значение, которое позволяет выключить проверку SSL- или TLS-сертификата при подключении к почтовому серверу:
     - `true` — проверка сертификата выключена.
     - `false` — проверка сертификата включена.

     Отключение проверки сертификата (`skip_verify = true`) в рабочем окружении может быть небезопасным. Рекомендуется устанавливать для `skip_verify` значение `true` только при тестировании.

▼ *Пример настроек SMTP*

```
[smtp]
enabled = true
host = smtp.example.com:587
user = yourusername
password = yourpassword
from_address = alert@example.com
from_name = Grafana Alerts
skip_verify = false
```

5. Нажмите на кнопку **Save**.
6. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.
7. Выберите контроллер Deployment `kube-prometheus-stack-grafana` из списка.
8. В открывшейся панели справа нажмите на кнопку **Restart**.

   Дождитесь перезапуска контроллера Deployment `kube-prometheus-stack-grafana`.

Настройка точки контакта

После настройки SMTP-сервера в конфигурационном файле можно настроить канал оповещений по электронной почте в веб-интерфейсе Grafana.

1. Выполните вход в веб-интерфейс Grafana.
2. Перейдите в раздел **Alerting** → **Contact points**.
3. Нажмите **New contact point**, чтобы создать новую точку контакта.
4. Введите имя точки контакта в поле **Name**.
5. В списке **Contact point type** выберите **Email**.
6. В поле **Addresses** укажите адреса электронной почты получателей (можно указать несколько адресов через разделитель `;`).
7. В разделе **Optional Email settings** (опционально):
   a. Включите опцию **Single email**, чтобы отправить одно письмо всем получателям.
   b. В поле **Message** введите сообщение, которое будет включено в письмо. Можно использовать шаблонные переменные.
   c. В поле **Subject** укажите тему письма. Можно использовать шаблонные переменные.
8. В разделе **Notification settings** (опционально):
   a. Включите опцию **Disable resolved message**, чтобы не получать сообщения о разрешении оповещения.
9. Нажмите **Test** для отправки тестового сообщения и проверки работы интеграции.
10. Нажмите **Save contact point**, чтобы сохранить настройки.

Точка контакта теперь готова к использованию в правилах оповещения.

Добавление точки контакта в политику уведомлений

После настройки точки контакта для оповещений по электронной почте необходимо добавить ее в политику уведомлений, чтобы Grafana могла направлять оповещения на эту точку контакта.

Для добавления точки контакта в политику уведомлений выполните следующие шаги:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Убедитесь, что в выпадающем меню **Choose Alertmanager** выбран **Grafana**.
3. В секции **Root policy - default for all alerts** нажмите **Edit**.
4. В поле **Default contact point** выберите ранее созданную точку контакта для оповещений по электронной почте.
5. При необходимости в секции **Group by** выберите метки, по которым будут группироваться оповещения. Например, можно выбрать метки `grafana_folder` и `alertname`, чтобы группировать оповещения по этим критериям.
6. При необходимости разверните секцию **Timing options** и настройте параметры времени:
   a. **Group wait** — время ожидания перед отправкой первого уведомления для новой группы. По умолчанию 30 секунд.
   b. **Group interval** — минимальный интервал времени между отправкой уведомлений в группе. По умолчанию 5 минут.
   c. **Repeat interval** — минимальный интервал времени для повторной отправки уведомления, если в группе не было добавлено новых оповещений. По умолчанию 4 часа.
7. Нажмите **Save**, чтобы сохранить изменения.

После выполнения этих шагов все оповещения, которые соответствуют установленным критериям, будут отправляться на указанную точку контакта по электронной почте.

Работа с метриками Prometheus

В дистрибутив системы входит система мониторинга Prometheus, собирающая различные метрики работы кластера. В данном разделе описано, как получить доступ к панели управления Prometheus и просмотреть метрики, собранные системой.

Подключение к Prometheus

Чтобы получить доступ к панели управления Prometheus:

1. Осуществите проброс портов, выполнив на хосте с сетевым доступом к master-узлу кластера следующую команду:

   ```
   kubectl port-forward -n kube-prometheus-stack service/kube-prometheus-stack-prometheus <local_port>:9090
   ```

   Здесь:

   - `<local_port>` — порт на хосте, по которому будет осуществляться доступ к Prometheus.

2. В браузере откройте URL `localhost:<local_port>`, где `<local_port>` — порт с шага 1. Отобразится панель управления Prometheus.

Просмотр списка метрик, получаемых из источников

Источники метрик — это сервисы, из которых Prometheus получает метрики.

Чтобы просмотреть список источников, на панели управления Prometheus перейдите в раздел **Status** → **Targets**. Для каждого источника доступны метки **Labels**, по которым можно фильтровать значения метрик. Например, в качестве метки может выступать пространство имен или под, на котором располагается источник.

Просмотреть список метрик, получаемых Prometheus из всех источников, можно на главной странице панели управления:

- Чтобы просмотреть список всех доступных метрик:
  1. Введите в строку запроса Prometheus следующий запрос:

     ```
     group by(__name__) ({__name__!=""})
     ```

  2. Нажмите на кнопку **Execute** или на клавишу ENTER.
- Чтобы найти метрики по ключевому слову в названии, введите ключевое слово в строку запроса Prometheus. Отобразится список автодополнения, содержащий метрики с указанным ключевым словом в названии.

Просматривать список метрик, получаемых из конкретного источника, необходимо непосредственно на источнике. Для этого выполните следующие действия:

1. Перейдите в раздел Prometheus **Status** → **Targets**.
2. Выберите нужный источник из списка.
3. Получите из столбца **Endpoint** таблицы порт и путь к конечной точке, на которой находится источник метрик.

   Например, для адреса `http://10.233.116.160:8001/metrics` порт — `8001`, а путь — `/metrics`.

4. Получите из столбца **Labels** таблицы пространство имен и под кластера, где находится источник метрик. Пространство имен указано в значении метки `namespace`, а под — в значении метки `pod`.
5. Осуществите проброс портов до источника метрик, выполнив на хосте с сетевым доступом к master-узлу кластера следующую команду:

   ```
   kubectl port-forward -n <namespace> pod/<source_pod>
   <source_local_port>:<source_remote_port>
   ```

   Здесь:

   - `<namespace>` — пространство имен кластера, в котором находится источник метрик.

- `<source_pod>` — под, на котором находится источник метрик.
- `<source_local_port>` — порт на хосте, по которому будет осуществляться доступ к источнику метрик. Должен отличаться от порта, выбранного для доступа к Prometheus.
- `<source_remote_port>` — порт, на котором находится источник метрик.

6. В браузере откройте URL `localhost:<source_local_port>/<source_path>`.

   Здесь:

   - `<source_local_port>` — порт с предыдущего шага.
   - `<source_path>` — путь к конечной точке источника метрик, полученный на шаге 3.

     Откроется URL, соответствующий источнику метрик.

▼ *Пример получения списка метрик шарда ClickHouse*

1. Перейдите в раздел Prometheus **Status** → **Targets**.
2. Выберите источник `serviceMonitor/kube-prometheus-stack/kube-prometheus-stack-clickhouse/0`.

   Далее рассматривается получение списка метрик для шарда `clickhouse-shard0-0`. Для второго шарда процесс аналогичен.

3. Получите из столбца **Endpoint** таблицы информации об источнике порт и путь конечной точки, отправляющей метрики шарда `clickhouse-shard0-0`. Пусть конечная точка имеет адрес `http://10.0.0.173:8001/metrics`. Тогда:
   - порт `8001`;
   - путь `/metrics`.
4. Получите из столбца **Labels** таблицы пространство имен и под кластера, где находится источник метрик. Пусть пространство имен в значении метки `namespace` — `system`, а под в значении метки `pod` — `clickhouse-shard0-0`.
5. Осуществите проброс портов до источника метрик, выполнив на хосте с сетевым доступом к master-узлу кластера следующую команду:

   ```
   kubectl port-forward -n system pod/clickhouse-shard0-0 5555:8001
   ```

6. В браузере откройте URL `localhost:5555/metrics`. Отобразится список метрик ClickHouse.

Вывод значений метрик в интерфейсе Prometheus

На главной странице панели управления Prometheus можно выводить значения получаемых системой метрик. Для этого:

1. Введите запрос на языке PromQL в строку запроса Prometheus.
2. Нажмите на кнопку **Execute** или на клавишу ENTER.
3. При необходимости измените представление данных, пользуясь вкладками:
   - **Table** — таблица значений метрик в момент времени, заданный в поле ввода **Evaluation time**;

- **Graph** — графическое представление временного ряда значений метрики. Для него можно выбрать интервал времени для отображения, конечный момент времени **End time** и интервал в секундах между точками графика **Res. (s)**.



Метрики, собранные Prometheus, также можно отобразить на дашборде Grafana.

Примеры метрик состояния кластера

Использование системных ресурсов

- Использование процессора:
  - `container_cpu_usage_seconds_total` — время процессора в секундах, потребленное контейнерами кластера.
  - `machine_cpu_cores` — количество логических ядер процессора.

    ▼ *PromQL-запрос потребления ресурсов процессора всеми контейнерами кластера*

    Метрика `container_cpu_usage_seconds_total` имеет тип `counter`, то есть является счетчиком с накоплением. Чтобы вычислить изменение метрики за временной период, воспользуйтесь функцией `rate` языка PromQL:

    ```
    sum(rate(container_cpu_usage_seconds_total{container!=""}[1m])) /
    sum(machine_cpu_cores) * 100
    ```

  - `container_threads` — количество потоков, запущенных внутри контейнера.

    ▼ *PromQL-запрос количества потоков внутри указанного контейнера*

    ```
    container_threads{container="<container_name>"}
    ```

    Здесь:

    - `<container_name>` — имя контейнера.

- Использование оперативной памяти:
  - `container_memory_usage_bytes` — потребление оперативной памяти контейнерами в байтах.
  - `machine_memory_bytes` — размер оперативной памяти в байтах, установленной на машине.

    ▼ *PromQL-запрос потребления оперативной памяти контейнерами на узле*

    ```
    sum(container_memory_usage_bytes{node="<node_name>"}) /
    sum(machine_memory_bytes{node="<node_name>"}) * 100
    ```

    Здесь:

    - `<node_name>` — имя узла кластера.

- Использование дискового пространства:

- `host_filesystem_used_bytes` — размер используемого дискового пространства в байтах.
- `host_filesystem_total_bytes` — размер установленного дискового пространства в байтах.

▼ *PromQL-запрос процента свободного места в файловой системе*

```
sum(host_filesystem_used_bytes{filesystem="<filesystem>"}) /
sum(host_filesystem_total_bytes{filesystem="<filesystem>"}) * 100
```

Здесь:

- `<filesystem>` — имя файловой системы.

- Использование сети:
  - `container_network_transmit_bytes_total` — объем переданных по сети данных в байтах.
  - `container_network_receive_bytes_total` — объем полученных по сети данных в байтах.

▼ *PromQL-запрос количества мегабайт, переданных по сети за одну минуту*

Метрики `container_network_transmit_bytes_total` и `container_network_receive_bytes_total` имеют тип `counter`. Чтобы вычислить изменение метрики за временной период, воспользуйтесь функцией `rate` языка PromQL:

```
sum(rate(container_network_transmit_bytes_total{node="<node_name>"}[1m])) /
(1024 ^ 2)
```

Здесь:

- `<node_name>` — имя узла кластера.
- Байты переведены в мегабайты путем деления на 1024^2.

Метрики ClickHouse

- Общие метрики запросов:
  - `ClickHouseMetrics_Query` — количество выполняющихся запросов.
  - `ClickHouseProfileEvents_FailedQuery` — количество запросов, завершившихся с ошибкой.
  - `ClickHouseProfileEvents_FailedSelectQuery` — количество запросов типа `SELECT`, завершившихся с ошибкой.
  - `ClickHouseProfileEvents_FailedInsertQuery` — количество запросов типа `INSERT`, завершившихся с ошибкой.
  - `ClickHouseMetrics_GlobalThreadActive` — количество потоков, выполняющих задачи, в глобальном пуле потоков ClickHouse.
  - `ClickHouseErrorMetric_ALL` — количество ошибок сервера ClickHouse с последнего перезапуска.

▼ *PromQL-запрос суммарного значения метрики по одному шарду ClickHouse*

```
ClickHouseMetrics_Query{pod="<pod_name>"}
```

Здесь:

- `<pod_name>` — имя пода, на котором расположен шард ClickHouse.

Вместо `ClickHouseMetrics_Query` можно подставить любую метрику из списка общих метрик.

▼ *PromQL-запрос значения метрики по всем шардам ClickHouse*

```
sum(ClickHouseMetrics_Query)
```

Вместо `ClickHouseMetrics_Query` можно подставить любую метрику из списка общих метрик.

- Метрики времени выполнения запросов:
  - `ClickHouseProfileEvents_QueryTimeMicroseconds` — время выполнения запросов в микросекундах.
  - `ClickHouseProfileEvents_SelectQueryTimeMicroseconds` — время выполнения запросов типа `SELECT` в микросекундах.
  - `ClickHouseProfileEvents_InsertQueryTimeMicroseconds` — время выполнения запросов типа `INSERT` в микросекундах.
  - `ClickHouseProfileEvents_OtherQueryTimeMicroseconds` — время выполнения запросов всех типов, кроме `SELECT` и `INSERT`, в микросекундах.

  ▼ *PromQL-запрос времени, затраченного на выполнение запросов в течение одной минуты*

  Метрики, описанные в этом разделе, имеют тип `counter`. Чтобы вычислить изменение метрики за временной период, воспользуйтесь функцией `rate` языка PromQL:

  ```
  sum(rate(ClickHouseProfileEvents_QueryTimeMicroseconds[1m])) / (10^6)
  ```

  Здесь:

  - Микросекунды переведены в секунды путем деления на 10^6.

- Метрики подключений к серверу СУБД:
  - `ClickHouseProfileEvents_DistributedConnectionTries` — количество всех попыток подключения к серверу СУБД.
  - `ClickHouseProfileEvents_DistributedConnectionUsable` — количество успешных попыток подключения к серверу СУБД.
  - `ClickHouseProfileEvents_DistributedConnectionFailTry` — количество неудачных попыток подключения к серверу СУБД, завершившихся повторной попыткой соединения.
  - `ClickHouseProfileEvents_DistributedConnectionFailAtAll` — количество неудачных попыток подключения к серверу СУБД, завершившихся после исчерпания повторных попыток.

▼ *PromQL-запрос количества всех попыток подключения к серверу СУБД в течение одной минуты*

Метрики, описанные в этом разделе, имеют тип `counter`. Чтобы вычислить изменение метрики за временной период, воспользуйтесь функцией `rate` языка PromQL:

```
sum(rate(ClickHouseProfileEvents_DistributedConnectionTries[1m]))
```

Управление кластером

В данном разделе описано управление компонентами кластера средствами командной строки и рабочего стола Lens.



По умолчанию для редактирования конфигурации кластера и его компонентов используется текстовый редактор **vim**. Чтобы использовать другой редактор, переопределите переменную окружения `KUBE_EDITOR` в файле `~/.bashrc`.

▼ *Пример изменения текстового редактора по умолчанию на nano*

```
echo "export KUBE_EDITOR=nano" | tee --append ~/.bashrc
source ~/.bashrc
```

Содержание раздела:

- Управление узлами кластера
- Работа с подами audit-manager и collector-audit
- Работа с подом collector-manager
- Работа с подом active-list-manager
- Увеличение размера тома для подов
- Получение идентификатора сервиса в NATS Service Discovery
- Работа с объектным хранилищем NATS Object Store
- Настройка стримов в менеджере пространств
- Работа с паролями в кластере
- Изменение доменного имени установленного экземпляра системы
- Настройка отдельного диска под образы контейнеров
  - Ошибки при запуске пода ingress-nginx-controller
- Поиск установленных экземпляров системы

Управление узлами кластера

Данный раздел содержит инструкции по работе с узлами кластера. Доступные операции:

- Просмотр состояния узлов кластера
- Добавление узла к кластеру

- Перезапуск узла кластера
- Удаление узла кластера

Просмотр состояния узлов кластера

С помощью командной строки

Чтобы просмотреть состояние узлов кластера, выполните следующую команду:

```
kubectl get nodes -o wide
```

▼ *Пример списка узлов кластера*

```
NAME                     STATUS   ROLES           AGE    VERSION   INTERNAL-IP
EXTERNAL-IP   OS-IMAGE
master01.company.local   Ready    control-plane   140d   v1.26.3   10.10.10.1
<none>        RED OS MUROM (7.3.2)
worker01.company.local   Ready    <none>          140d   v1.26.3   10.10.10.2
<none>        RED OS MUROM (7.3.2)
worker02.company.local   Ready    <none>          140d   v1.26.3   10.10.10.3
<none>        RED OS MUROM (7.3.2)
worker03.company.local   Ready    <none>          140d   v1.26.3   10.10.10.4
<none>        RED OS MUROM (7.3.2)
```

Состояние узлов отображается в столбце `STATUS`. Возможные статусы:

- `Ready`: узел исправен и готов к размещению подов.
- `NotReady`: узел столкнулся с некоторой проблемой, и на нем нельзя разместить под.
- `SchedulingDisabled`: на узле отключено создание новых подов.
- `Unknown`: узел недоступен для работы.

С помощью Lens

Чтобы просмотреть состояние узлов кластера:

1. Подключитесь к кластеру с помощью Lens.
2. Перейдите в раздел **Nodes**. В правой части экрана отобразится таблица узлов кластера. Состояние узлов отобразится в столбце **Conditions**.



Также состояние узлов кластера можно посмотреть в разделе **Инструменты → Мониторинг** веб-интерфейса системы.

Добавление узла к кластеру

Добавление узла к кластеру производится в два этапа:

1. Подготовительные работы на **узле, добавляемом в кластер**:

a. Установите утилиты, требуемые для подключения узла к кластеру:

```
sudo dnf install bash-completion vim telnet chrony jq htop tcpdump zip unzip
rsync ebtables ethtool conntrack -y
```

b. Настройте синхронизацию времени с сервером времени:

```
sudo timedatectl set-ntp true
sudo sed -i "/^server /d" /etc/chrony.conf
sudo bash -c 'echo "server <time_server>" >> /etc/chrony.conf'
sudo systemctl enable --now chronyd
sudo systemctl restart chronyd
sudo systemctl status chronyd
chronyc sources
```

Здесь:

- `<time_server>` — адрес корпоративного сервера времени (например, `ntp.rvision.local`).

2. Работы на **master-узле**:

a. Внесите в файл `inventory/offline-cluster/inventory.ini` информацию о новом узле:

```
sudo nano inventory/offline-cluster/inventory.ini
```

Внесите информацию об узле в группу `all` и в группу, для которой он предназначен. В качестве образца синтаксиса используйте поля инвентарных данных уже имеющихся в этом файле узлов.

b. Запустите Ansible-плейбуки для добавления узла в кластер:

```
sudo chmod -R 755 inventory/offline-cluster/credentials
sudo chmod 777 inventory/offline-cluster/credentials
ansible-playbook -uuser -b -i inventory/offline-cluster/inventory.ini
sysprep.yml --extra-vars archive_directory=../../.. --extra-vars
sysprep_name=sys-prep-redos
ansible-playbook -uuser -b -i inventory/offline-cluster/inventory.ini
cluster.yml
cd ~
```

c. Отключите offline-репозитории:

```
sudo dnf config-manager --set-disabled deps-offline
sudo dnf config-manager --set-enabled base updates
```

d. Проверьте список узлов в кластере:

```
sudo kubectl get node
```

e. Проверьте, какой набор приложений был развернут с помощью `kubespray`:

```
sudo kubectl get po -A
```

Перезапуск узла кластера

При выполнении работ по обслуживанию узла кластера (например, обновлению ОС или добавлению вычислительных ресурсов на хост) может понадобиться перезапуск узла. Чтобы сделать это, выполните на узле следующие команды:

1. Переведите узел в `drain`-режим:

```
kubectl drain <node_name> --ignore-daemonsets
```

Здесь:

- `<node_name>` — имя узла (например, `system-worker05.k8s.dev.rvision.local`).



Операция `drain` позволяет перераспределить все поды с узла. В процессе выполнения `drain` узел помечается как `unschedulable`. Это предотвращает появление на нем новых подов. Затем `drain` начинает перемещать поды с узла и завершает работу контейнеров, которые на данный момент запущены на узле, отправляя сигнал `TERM` контейнерам в поде.

2. Остановите сервисы `kubelet` и `containerd`:

```
sudo systemctl stop kubelet
sudo systemctl stop containerd
```

3. Произведите обслуживание узла.
4. Повторно запустите сервисы `kubelet` и `containerd`:

```
sudo systemctl start containerd
sudo systemctl start kubelet
```

5. Сделайте узел вновь назначаемым, чтобы на нем появлялись поды:

```
kubectl uncordon <node_name>
kubectl get nodes -o wide
```

Здесь:

- `<node_name>` — имя узла (например, `system-worker05.k8s.dev.rvision.local`).

Удаление узла кластера

Для удаления узла выполните следующие шаги:

1. Откройте файл инвентаризации:

   ```
   sudo nano inventory/offline-cluster/inventory.ini
   ```

   Удалите узел из файла инвентаризации и сохраните его.

2. Переведите узел в `drain`-режим, чтобы освободить ресурсы на нем:

   ```
   kubectl drain --ignore-daemonsets worker02.example.local
   ```

3. После того как ресурсы будут перераспределены, запустите плейбук на удаление узлов:

   Так как плейбуку требуется доступ к файлу `/etc/kubernetes/admin.conf` и привилегированный доступ, эти шаги необходимо производить от имени пользователя root.

   ```
   ansible-playbook --ask-pass -uroot -i inventory/offline-cluster/inventory.ini
   remove-node.yml --extra-vars "node=worker02.example.local"
   ```

   Плейбук запросит подтверждение операции.

4. После завершения работы плейбука запросите список узлов и проверьте, что узел удален:

   ```
   kubectl get nodes
   ```

Работа с подами audit-manager и collector-audit

В данном разделе описана работа с подами `audit-manager` и `collector-audit`.

Под `audit-manager` предназначен для управления аудитом системных событий. Он направляет данные аудита от сервисов к поду `collector-audit`.

Под `collector-audit` предназначен для работы коллектора аудита. **Коллектор аудита** — это внутренний коллектор, который собирает системные события аудита и отправляет их в хранилище событий аудита. Также коллектор аудита направляет данные в стрим NATS с наименованием `r-space-broadcast-commands-output` для доступа к данным аудита из сервисов системы. Коллектор аудита скрыт из интерфейса системы.

Получение полных имен подов

Каждому поду присваивается случайный идентификатор, добавляемый к имени сервиса, для

которого создается под.

Чтобы получить полное имя для каждого из подов, выполните следующую команду:

- Для пода `audit-manager`:

  ```
  kubectl -n <namespace> get pods -l app.kubernetes.io/name=evo.streams.audit-manager
  ```

  Здесь:

  - `<namespace>` — название пространства имен, в котором установлена система.

- Для пода `collector-audit`:

  ```
  kubectl -n <namespace> get pods -l app.kubernetes.io/name=wl-a3741e20-bb69-4ad8-9d33-c0792dbc4701
  ```

  Здесь:

  - `<namespace>` — название пространства имен, в котором установлена система.

▼ *Пример вывода имени пода* `audit-manager`

```
NAME                                          READY   STATUS    RESTARTS   AGE
evo.streams.audit-manager-6cc79cl87l-mnpf6    1/1     Running   0          45m
```

Опция `-l` команды `kubectl get pods` позволяет отфильтровать список подов кластера по значению селектора `app.kubernetes.io/name`.

В примере выше полное имя пода `audit-manager` — `evo.streams.audit-manager-6cc79cl87l-mnpf6`.

Просмотр логов подов



Логи пода очищаются при его остановке.

Для просмотра логов подов `audit-manager` и `collector-audit` используйте инструкцию из раздела Просмотр логов пода. В качестве имени пода (`<pod_name>`) укажите полные имена подов, полученные ранее.

Изменение ресурсов для обработки событий аудита

Если в определенный момент приходит большое количество событий аудита, то вы можете увеличить потребляемые ресурсы и пропускную способность подов `audit-manager` и `collector-audit` следующими способами:

- Через увеличение количества реплик и ресурсов для подов `audit-manager` и `collector-audit`.
- Через увеличение максимального количества неподтвержденных сообщений, которое потребитель (consumer) может получить, для подов `audit-manager` и `collector-audit`.

Изменение ресурсов и пропускной способности пода audit-manager

Чтобы изменить ресурсы и пропускную способность пода `audit-manager`:

1. Подключитесь к кластеру.
2. Откройте ConfigMap `evo.streams-audit-manager-env`, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.streams-audit-manager-env
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

3. Измените максимальное количество неподтвержденных сообщений, которое может принять потребитель пода `audit-manager`, в параметре `SIEM_AUDIT_MANAGER_CONSUMER_MAX_ACK_PENDING`.
4. Сохраните файл.
5. Откройте конфигурацию контроллера Deployment `evo.streams.audit-manager`, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit deployment evo.streams.audit-manager
   ```

   Конфигурация контроллера Deployment откроется в текстовом редакторе, который выбран в системе по умолчанию.

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
6. Измените количество реплик пода в поле `replicas` блока конфигурации `spec`.
7. Измените потребляемые ресурсы пода в следующих полях блока конфигурации `spec.template.spec.containers.resources.limits`:
   - `cpu` — количество ядер CPU пода.
   - `memory` — количество ОЗУ пода.
8. Сохраните и закройте временный файл конфигурации контроллера.
9. Перезагрузите контроллер Deployment `evo.streams.audit-manager`, чтобы применить изменения:

   ```
   kubectl -n <namespace> rollout restart deployment evo.streams.audit-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

Изменение ресурсов и пропускной способности пода collector-audit

Чтобы изменить ресурсы и пропускную способность пода `collector-audit`:

1. Подключитесь к кластеру.
2. Откройте ConfigMap `evo.streams-audit-manager-env`, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.streams-audit-manager-env
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
3. Отредактируйте необходимые параметры:
   - `SIEM_AUDIT_MANAGER_AUDIT_COLLECTOR_CONSUMER_MAX_ACK_PENDING` — количество неподтвержденных сообщений, которое потребитель (consumer) может получить от сервера в коллектор аудита.
   - `SIEM_AUDIT_MANAGER_AUDIT_COLLECTOR_REPLICA_COUNT` — количество реплик пода, который передает данные аудита.
   - `SIEM_AUDIT_MANAGER_AUDIT_COLLECTOR_CPU` — количество ядер CPU пода, который передает данные аудита.
   - `SIEM_AUDIT_MANAGER_AUDIT_COLLECTOR_MEMORY_MB` — ОЗУ пода, который передает данные аудита.

   Изменение ресурсов и количества реплик пода `collector-audit` производится в ConfigMap, потому что при перезапуске пода `audit-manager` пересоздается контроллер Deployment пода `collector-audit`. Данные о ресусах и количестве реплик пода `collector-audit` добавляются в новый контроллер Deployment из ConfigMap `evo.streams-audit-manager-env`.
4. Сохраните файл.
5. Перезагрузите контроллер Deployment `evo.streams.audit-manager`, чтобы применить изменения:

   ```
   kubectl -n <namespace> rollout restart deployment evo.streams.audit-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

Работа с подом collector-manager

В данном разделе описана работа с подом `collector-manager`, который управляет всеми коллекторами в системе. Анализ логов и событий этого пода помогает в диагностике и устранении проблем, связанных с работой коллекторов.

Получение полного имени пода collector-manager

Каждому поду присваивается случайный идентификатор, добавляемый к имени сервиса, для

которого создается под.

Чтобы получить полное имя пода `collector-manager`, выполните следующую команду:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/name=evo.streams.collector-
manager
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

▼ *Пример вывода имени пода* `collector-manager`

```
NAME                                              READY   STATUS    RESTARTS   AGE
evo.streams.collector-manager-59dbb7767-hx8nq     1/1     Running   0          43m
```

Опция `-l` команды `kubectl get pods` позволяет отфильтровать список подов кластера по значению селектора `app.kubernetes.io/name`.

В примере выше полное имя пода `collector-manager` — `evo.streams.collector-manager-59dbb7767-hx8nq`.

Просмотр логов пода collector-manager



Логи пода очищаются при его остановке.

Для просмотра логов пода `collector-manager` используйте инструкцию из раздела Просмотр логов пода. В качестве имени пода (`<pod_name>`) укажите полное имя пода, полученное ранее.

Изменение уровня логирования collector-manager

Если текущие логи пода `collector-manager` содержат избыточную или недостаточную информацию, уровень логирования можно просмотреть и изменить с помощью командной строки. Доступны следующие уровни логирования:

- `error` — записываются критические ошибки, которые могут препятствовать работе коллекторов.
- `warn` — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
- `info` — записывается общая информация о работе и состоянии коллекторов.
- `debug` — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем.
- `trace` — самый подробный уровень логирования, предоставляющий информацию о каждом шаге работы коллекторов.

Просмотр текущего уровня логирования

Чтобы просмотреть текущий уровень логирования:

1. Получите полное имя пода `collector-manager`.
2. Выполните следующую команду:

   ```
   kubectl exec -n <namespace> <pod_name> -- node -e
   "console.log(process.env.SIEM_COLLECTOR_MANAGER_LOG_LEVEL)"
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<pod_name>` — полное имя пода `collector-manager`, полученное на шаге 1.

Изменение уровня логирования

Чтобы изменить уровень логирования:

1. Получите полное имя пода `collector-manager`.
2. Отредактируйте ConfigMap `collector-manager`:
   a. Откройте ConfigMap, выполнив следующую команду:

      ```
      kubectl -n <namespace> edit configmap evo.streams-collector-manager-env
      ```

      Здесь:

      - `<namespace>` — название пространства имен, в котором установлена система.

   b. Укажите требуемый уровень логирования (`error`, `warn`, `info`, `debug` или `trace`) как значение ключа `SIEM_COLLECTOR_MANAGER_LOG_LEVEL`.
   c. Сохраните файл.
   d. Перезагрузите контроллер Deployment `evo.streams.collector-manager`, чтобы применить изменения:

      ```
      kubectl -n <namespace> rollout restart deployment evo.streams.collector-manager
      ```

      Здесь:

      - `<namespace>` — название пространства имен, в котором установлена система.

Работа с подом active-list-manager

В данном разделе описана работа с подом `active-list-manager`, который управляет активными списками.

Получение полного имени пода active-list-manager

Каждому поду присваивается случайный идентификатор, добавляемый к имени сервиса, для которого создается под.

Чтобы получить полное имя пода `active-list-manager`, выполните следующую команду:

```
kubectl get pods -n <namespace> -l app.kubernetes.io/name=evo.streams.active-list-manager
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

▼ *Пример вывода имени пода* `active-list-manager`

```
NAME                                                READY   STATUS    RESTARTS   AGE
evo.streams.active-list-manager-867c6959cf-hn7c5    1/1     Running   0          43m
```

Опция `-l` команды `kubectl get pods` позволяет отфильтровать список подов кластера по значению селектора `app.kubernetes.io/name`.

В примере выше полное имя пода `active-list-manager` — `evo.streams.active-list-manager-867c6959cf-hn7c5`.

Просмотр логов пода active-list-manager



Логи пода очищаются при его остановке.

Для просмотра логов пода `active-list-manager` используйте инструкцию из раздела Просмотр логов пода. В качестве имени пода (`<pod_name>`) укажите полное имя пода, полученное ранее.

Изменение потребляемых ресурсов пода

Чтобы изменить потребляемые ресурсы пода:

1. Подключитесь к кластеру.
2. Откройте конфигурацию контроллера Deployment `evo.streams.active-list-manager`, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit deployment evo.streams.active-list-manager
   ```

   Конфигурация контроллера Deployment откроется в текстовом редакторе, который выбран в системе по умолчанию.

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

3. Измените потребляемые ресурсы пода в следующих полях блока конфигурации `spec.template.spec.containers.resources.limits`:
   - `cpu` — количество ядер CPU пода.
   - `memory` — количество ОЗУ пода.
4. Сохраните и закройте временный файл конфигурации контроллера.
5. Перезагрузите контроллер Deployment `evo.streams.active-list-manager`, чтобы применить изменения:

```
kubectl -n <namespace> rollout restart deployment evo.streams.active-list-manager
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Увеличение интервала liveness-проб пода



Рекомендуется редактировать настройки liveness-проб пода, только если у вас возникла ошибка при импорте файла с большим количеством записей активного списка. После импорта рекомендуется вернуть настройки liveness-проб на значения, которые были установлены изначально.

Чтобы изменить настройки liveness-проб пода:

1. Подключитесь к кластеру.
2. Откройте конфигурацию контроллера Deployment `evo.streams.active-list-manager`, выполнив следующую команду:

```
kubectl -n <namespace> edit deployment evo.streams.active-list-manager
```

Конфигурация контроллера Deployment откроется в текстовом редакторе, который выбран в системе по умолчанию.

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

3. Увеличьте интервалы liveness-проб пода в следующих полях блока конфигурации `spec.template.spec.containers.livenessProbe`:
   - `timeoutSeconds` — количество секунд на ожидание ответа от сервиса.
   - `failureThreshold` — количество неудачных попыток ожидания ответа, прежде чем Kubernetes принудительно перезапустит сервис.
4. Сохраните и закройте временный файл конфигурации контроллера.

5. Перезагрузите контроллер Deployment `evo.streams.audit-manager`, чтобы применить изменения:

   ```
   kubectl -n <namespace> rollout restart deployment evo.streams.audit-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

6. Произведите импорт записей активного списка.
7. Восстановите изначальные интервалы liveness-проб в конфигурации контроллера Deployment `evo.streams.active-list-manager`. Ее можно открыть, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit deployment evo.streams.active-list-manager
   ```

Увеличение размера тома для подов

В данном разделе описано, как увеличить размер тома пода через изменение ресурсов PVC (Persistent Volume Claims).

С помощью командной строки

1. Чтобы просмотреть список всех PVC в кластере, выполните следующую команду:

   ```
   kubectl get pvc -n <namespace>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

   ▼ *Пример списка PVC в кластере*

   ```
   NAME                                                STATUS   VOLUME
   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
   data-clickhouse-shard0-0                            Bound    pvc-fb323380-
   6e5f-43c5-9cf8-8db10d4cf32b   30Gi       RWO            local-path     71d
   data-postgresql-0                                   Bound    pvc-4425107b-
   a5d3-4bf0-b336-1a815d467bdc   8Gi        RWO            local-path     71d
   dkron                                               Bound    pvc-b70dfffa-
   220a-486f-ace2-8957eb79c86e   64Mi       RWO            local-path     71d
   gateway-storage-9e527291-56ca-4df5-acbd-528c65b3412f   Bound    pvc-d466499b-
   055b-4aaf-ad1b-5b02b5989aa6   1Mi        RWO            local-path     50d
   gateway-storage-b3d6eea7-43f2-4410-92a1-6a00794909d2   Bound    pvc-cdc41e23-
   c9ee-4ffe-a726-859f47f6c3e0   30Mi       RWO            local-path     71d
   gateway-storage-be9c2592-1170-4961-9bbb-7654ce715d41   Bound    pvc-688ba632-
   1cc8-41a6-8b6a-e75f688f5b62   100Mi      RWO            local-path     71d
   ```

```
license                                                  Bound    pvc-8b346e5f-
a547-46e9-bb2e-d8bc6d130e09   10Mi       RWO            local-path     71d
minio                                                    Bound    pvc-467c6b2a-
dd03-49ec-aea7-d484e1e2b283   8Gi        RWO            local-path     71d
mongodb                                                  Bound    pvc-96d5a340-
31ca-453e-a839-40ede572ed87   8Gi        RWO            local-path     71d
nats-bridge-js-nats-bridge-0                             Bound    pvc-1ed1e8c3-
3d5d-41ca-9b0a-b5de8a8f1e80   10Gi       RWO            local-path     71d
nats-main-js-nats-main-0                                 Bound    pvc-46ddf2af-
330b-47bb-9d19-ad4bd94fe63e   30Gi       RWO            local-path     71d
nats-space-js-nats-space-0                               Bound    pvc-01e1c741-
22db-49f1-a650-0ea7060504ad   30Gi       RWO            local-path     71d
```

Для получения подробной информации о PVC выполните следующую команду:

```
kubectl describe pvc <pvc_name> -n <namespace>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<pvc_name>` — имя PVC.

▼ *Пример подробной информации о PVC*

```
Name:          data-clickhouse-shard0-0
Namespace:     evo
StorageClass:  local-path
Status:        Bound
Volume:        pvc-fb323380-6e5f-43c5-9cf8-8db10d4cf32b
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
cluster.local/local-path-provisioner
               volume.kubernetes.io/selected-node: worker-01
               volume.kubernetes.io/storage-provisioner:
cluster.local/local-path-provisioner
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      30Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Used By:       clickhouse-shard0-0
Events:        <none>
```

2. Чтобы открыть файл конфигурации PVC, выполните следующую команду:

```
kubectl edit pvc <pvc_name>
```

Здесь:

- `<pvc_name>` — имя PVC.

3. В открывшемся файле задайте новый размер PVC в ключе `spec.resources.requests.storage`:

```
# ...
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      # отредактируйте значение этого ключа
      storage: 64Mi
# ...
```

4. Закройте файл. Kubernetes перераспределит дисковое пространство в соответствии с обновленным PVC, и размер тома изменится через некоторое время.

С помощью Lens

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Storage** → **Persistent Volume Claims**.
3. Выберите пространство имен кластера в выпадающем списке **Select Namespace**. Отобразится таблица PVC.
4. Выберите нужный PVC в списке. В правой части экрана отобразится панель свойств PVC.
5. Нажмите на кнопку **Edit** в правом верхнем углу экрана. В нижней части экрана отобразится файл конфигурации PVC.
6. Отредактируйте значение ключа `spec.resources.requests.storage` аналогично п. 3 инструкции выше.
7. Закройте файл. Kubernetes перераспределит дисковое пространство в соответствии с обновленным PVC, и размер тома изменится через некоторое время.

Получение идентификатора сервиса в NATS Service Discovery

Автоматическое обнаружение сервисов необходимо для поддержания корректной работы системы при различных конфигурациях кластера, в которых она может быть установлена. Данное руководство описывает получение идентификатора сервиса в компоненте обнаружения сервисов NATS Service Discovery на примере ClickHouse.

Чтобы получить идентификатор ClickHouse в NATS Service Discovery, выполните следующие действия:

1. Подключитесь к терминалу master-узла кластера.
2. Подключитесь к терминалу пода `nats-main-box`, выполнив следующую команду:

```
kubectl exec -itn <namespace> deployment/nats-main-box -c nats-box -- sh
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

3. Получите идентификатор БД ClickHouse из регистра Service Discovery, выполнив следующую команду:

```
nats kv ls r-space-service-discovery-registry
```

Пример вывода идентификаторов сервисов

```
space-main.nats-space
space-main.evo-clickhouse1
```

Искомый идентификатор содержит подстроку `clickhouse`; в примере — `space-main.evo-clickhouse1`.

Работа с объектным хранилищем NATS Object Store

NATS Object Store — это объектное хранилище, используемое для хранения файлов конфигурации различных объектов системы, например, коллекторов, таблиц обогащения и секретов. Эти конфигурации можно экспортировать из NATS Object Store.

Подключение к NATS Object Store

Чтобы получить доступ к терминалу NATS Object Store, выполните следующие действия:

1. Подключитесь к терминалу master-узла кластера.
2. Подключитесь к терминалу пода `nats-main-box`, выполнив следующую команду:

```
kubectl exec -itn <namespace> deployment/nats-main-box -c nats-box -- sh
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Экспорт конфигураций из NATS Object Store

Чтобы экспортировать конфигурацию объекта из NATS Object Store:

1. Найдите идентификатор (ID) объекта в графическом интерфейсе системы.
2. Откройте терминал NATS Object Store.
3. Получите список объектов в бакете `r-space-global-objects`, выполнив команду:

```
nats object ls r-space-global-objects
```

4. Выгрузите содержимое ключей, содержащих ID с шага 1. Для этого выполните следующую

команду для каждого из ключей:

```
nats object get r-space-global-objects <key> --output <out_file>
```

Здесь:

- `<key>` — ключ из бакета `r-space-global-objects`.
- `<out_file>` — путь к файлу внутри контейнера `nats-box`, в который будет записано содержимое ключа.

  ▼ *Пример выгрузки конфигурации коллектора*

  Пусть имеется коллектор с ID `4dedef79-c3e5-49ee-93f5-e1d31e408e96`. В бакете есть четыре ключа, содержащие этот ID:

  - `streams.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/global-functions/`;
  - `streams.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/event-source/ad67fcb9-f234-476b-b71a-7bc2609eb9b7/event-sources/`;
  - `streams.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/event-source/ad67fcb9-f234-476b-b71a-7bc2609eb9b7/event-sources-distributed-collector/`;
  - `streams.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/event-source/ad67fcb9-f234-476b-b71a-7bc2609eb9b7/event-sources-distributed-correlator/`.

  Для получения полной конфигурации коллектора необходимо выгрузить содержимое всех четырех ключей.

  При необходимости вы также можете выгрузить только определенные элементы конфигурации. Например, чтобы получить конфигурацию глобальных функций, выгрузите значение ключа `streams.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/global-functions/`.

5. Содержимое ключей на предыдущем шаге будет выгружено в файловую систему контейнера `nats-box`. Чтобы выгрузить их в файловую систему master-узла кластера, выполните следующие действия:

   a. В терминале master-узла кластера получите полное имя пода сервиса `nats-main-box`, выполнив следующую команду:

   ```
   kubectl -n <namespace> get pods -l app.kubernetes.io/instance=nats-main | grep nats-main-box
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

Пример вывода имени пода `nats-main-box`

```
nats-main-box-658b9fc759-h4wzm   1/1     Running   0          10d
```

В примере выше полное имя пода `nats-main-box` — `nats-main-box-658b9fc759-h4wzm`.

b. Скопируйте файлы конфигурации из пода `nats-main-box` на master-узел кластера, выполнив следующую команду:

```
kubectl -n <namespace> cp <nats_main_box_full_name>:<out_file> <target_out_file>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<nats_main_box_full_name>` — полное имя пода `nats-main-box`.
- `<out_file>` — имя файла внутри контейнера `nats-box`, в который было записано содержимое ключа конфигурации на шаге 4.
- `<target_out_file>` — имя файла на master-узле кластера, в который нужно скопировать содержимое ключа.

Настройка стримов в менеджере пространств

В данном разделе приведены переменные окружения, определяющие условия хранения данных в стримах менеджера пространств.

Задать значения данных переменных можно через ConfigMap `space-manager-extravars`.

Через командную строку

1. Подключитесь к кластеру.
2. Откройте ConfigMap для редактирования, выполнив команду:

```
kubectl edit configmap space-manager-extravars -n <namespace>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

Через Lens

1. Подключитесь к кластеру.
2. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
3. Выберите ConfigMap `space-manager-extravars` в пространстве имен, в котором

установлена система.

4. В открывшейся панели справа измените значения требуемых переменных.

Переменные окружения для настройки стримов

Общие настройки стримов

- `NATS_CONFIGURATOR_ACTIVE_LISTS_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value активных списков.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_BROADCAST_COMMANDS_INPUT_STREAM_MAX_SIZE` — максимальный объем данных в стриме Broadcast-команд для пространств.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_GLOBAL_OBJECTS_OS_MAX_SIZE` — максимальный объем данных в NATS Object Store для глобальных объектов.

  Значение по умолчанию: `500MB` (500 МБ).

Настройки стримов на стороне центрального кластера

- `NATS_CONFIGURATOR_MAIN_ACTIVE_LISTS_CHANGES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value изменений активных списков на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_BROADCAST_COMMANDS_OUTPUT_STREAM_MAX_SIZE` — максимальный объем данных в стриме Broadcast-команд из пространства на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_COMMANDS_INPUT_STREAM_MAX_SIZE` — максимальный объем данных в стриме команд для пространств на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_COMMANDS_OUTPUT_STREAM_MAX_SIZE` — максимальный объем данных в стриме команд из пространства на стороне центрального кластера.

  Значение по умолчанию: `1000MB` (1 ГБ).

- `NATS_CONFIGURATOR_MAIN_RUNTIME_CONFIGS_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value Runtime-конфигураций на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_RUNTIME_CONFIG_STATUSES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value статусов Runtime-конфигураций на стороне центрального кластера.

Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_SERVICE_DISCOVERY_REGISTRY_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value Service Discovery Registry на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_SERVICE_DISCOVERY_STATUSES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value статусов Service Discovery на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_SPACE_STATUSES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value статусов пространств на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_TELEMETRY_STREAM_MAX_AGE` — максимальное время хранения данных в стриме телеметрии на стороне центрального кластера.

  Значение по умолчанию: `4h` (4 часа).

- `NATS_CONFIGURATOR_MAIN_TELEMETRY_STREAM_MAX_SIZE` — максимальный объем данных в стриме телеметрии на стороне центрального кластера.

  Значение по умолчанию: `5GB` (5 ГБ).

- `NATS_CONFIGURATOR_MAIN_WORKLOADS_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value сервисов на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_MAIN_WORKLOAD_STATUSES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value статусов сервисов на стороне центрального кластера.

  Значение по умолчанию: `100MB` (100 МБ).

Настройки стримов в NATS Shared Key/Value

- `NATS_CONFIGURATOR_SHARED_KV_MAX_SIZE` — максимальный объем данных в NATS Shared Key/Value.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SHARED_KV_REGISTRY_KV_MAX_SIZE` — максимальный объем данных в NATS Shared Key/Value для Registry.

  Значение по умолчанию: `10MB` (10 МБ).

Настройки стримов на стороне пространства

- `NATS_CONFIGURATOR_SPACES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value пространств.

  Значение по умолчанию: `10MB` (10 МБ).

- `NATS_CONFIGURATOR_SPACE_ACTIVE_LISTS_CHANGES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value изменений активных списков на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_BROADCAST_COMMANDS_OUTPUT_STREAM_MAX_SIZE` — максимальный объем данных в стриме Broadcast-команд из пространства на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_COMMANDS_INPUT_STREAM_MAX_SIZE` — максимальный объем данных в стриме команд для пространств на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_COMMANDS_OUTPUT_STREAM_MAX_SIZE` — максимальный объем данных в стриме команд из пространства на стороне пространства.

  Значение по умолчанию: `1000MB` (1 ГБ).

- `NATS_CONFIGURATOR_SPACE_DASHBOARDS_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value дашбордов на стороне пространства.

  Значение по умолчанию: `10MB` (10 МБ).

- `NATS_CONFIGURATOR_SPACE_OBJECTS_OS_MAX_SIZE` — максимальный объем данных в NATS Object Store для объектов пространств.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_RUNTIME_CONFIGS_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value Runtime-конфигураций на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_RUNTIME_CONFIG_STATUSES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value статусов Runtime-конфигураций на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_SERVICE_DISCOVERY_REGISTRY_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value Service Discovery Registry на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_SERVICE_DISCOVERY_STATUSES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value статусов Service Discovery на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_SPACE_STATUSES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value статусов пространств на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_TELEMETRY_STREAM_MAX_AGE` — максимальное время хранения данных в стриме телеметрии на стороне пространства.

  Значение по умолчанию: `4h` (4 часа).

- `NATS_CONFIGURATOR_SPACE_TELEMETRY_STREAM_MAX_SIZE` — максимальный объем данных в стриме телеметрии на стороне пространства.

  Значение по умолчанию: `5GB` (5 ГБ).

- `NATS_CONFIGURATOR_SPACE_WORKLOADS_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value сервисов на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

- `NATS_CONFIGURATOR_SPACE_WORKLOAD_STATUSES_KV_MAX_SIZE` — максимальный объем данных в NATS Key/Value статусов сервисов на стороне пространства.

  Значение по умолчанию: `100MB` (100 МБ).

Работа с паролями в кластере

При изменении паролей от систем хранения и баз данных нужно поменять соответствующие секреты в Kubernetes (Kubernetes Secrets) — объекты, которые хранят конфиденциальную информацию, такую как пароли и токены. Может также потребоваться перезапуск соответствующих приложений, чтобы они смогли повторно прочитать актуальные данные.

В случае с ClickHouse регистрационные данные обновляются через NATS Key/Value — сервис, который хранит конфигурационные данные и позволяет централизованно управлять паролями и настройками.

Смена пароля пользователя ClickHouse

Чтобы обновить пароль пользователя ClickHouse:

1. Измените пароль ClickHouse, выполнив следующий запрос в консоли ClickHouse:

   SQL-запрос

   ```
   ALTER USER <user> IDENTIFIED BY '<new_password>';
   ```

   Здесь:

   - `<user>` — имя пользователя.
   - `<new_password>` — новый пароль.

2. Если требуется изменить пароль пользователя writer, обновите пароль в блоке `<remote_servers>` в конфигурации кластера.
3. Подключитесь к терминалу master-узла кластера.
4. Получите название ключа ClickHouse в бакете `r-space-service-discovery-registry`, выполнив команду:

```
kubectl exec -n <namespace> deployment/nats-space-box -- nats kv ls r-space-service-discovery-registry
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

5. Запишите содержимое ключа ClickHouse в отдельный файл формата JSON на master-узле, выполнив команду:

```
kubectl exec -n <namespace> deployment/nats-space-box -- nats kv get r-space-service-discovery-registry <key_name> --raw > <path_to_file>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<key_name>` — название ключа, полученное ранее.
- `<path_to_file>` — путь до файла формата JSON, например, `/path/to/config.json`.

6. В файле формата JSON с содержимым ключа ClickHouse измените значения паролей в следующих полях:

- `"owner_password"`;
- `"reader_password"`;
- `"writer_password"`.

▼ *Пример содержимого ключа ClickHouse*

```
{
  "name": "clickhouse1",
  "instances": [
    {
      "address": "click.host",
      "port": 1001,
      "meta": {
        "grpc_port": "1000",
        "is_tls_enabled": "true",
        "https_port": "0000",
        "ca_base_64_string": "base64_e3ncoded_cert=",
        "owner_password": "password",
        "reader_password": "password",
        "writer_password": "password"
      },
      "healthChecks": [
```

```
        {
          "id": "evo-clickhouse1-http-api-ping",
          "url": "http://click.host:0000/ping",
          "method": "GET",
          "interval": 10000,
          "timeout": 1000
        }
      ]
    }
  ]
}
```

7. Загрузите обновленное содержимое ключа ClickHouse, выполнив команду:

   ```
   kubectl exec -i -n <namespace> deployment/nats-space-box -- nats kv put r-space-service-discovery-registry <key_name> < <path_to_file>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<key_name>` — название ключа, полученное ранее.
   - `<path_to_file>` — путь до файла формата JSON, например, `/path/to/config.json`.

8. Перезапустите контроллер Deployment `evo.space.service-discovery`, выполнив команду:

   ```
   kubectl -n <namespace> rollout restart deployment evo.space.service-discovery
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   Дождитесь перезапуска контроллера.

9. Перезапустите связанные контроллеры Deployment, выполнив команды:

   ```
   kubectl -n <namespace> rollout restart deployment evo.streams.event-storage-manager
   kubectl -n <namespace> rollout restart deployment evo.streams.active-list-manager
   kubectl -n <namespace> rollout restart deployment evo.streams.query-log-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

Смена пароля PostgreSQL

Чтобы обновить пароль PostgreSQL:

Через командную строку

1. Подключитесь к кластеру.
2. Узнайте имя пода PostgreSQL, выполнив команду:

   ```
   kubectl get pods -n <namespace> -l app.kubernetes.io/instance=postgresql
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
3. Подключитесь к контейнеру пода PostgreSQL, выполнив команду:

   ```
   kubectl exec -it -n <namespace> <pod_name> -- sh
   ```

   При наличии нескольких контейнеров у пода PostgreSQL добавьте оператор `-c <container_name>` в команду выше.

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<pod_name>` — имя пода PostgreSQL, полученное на предыдущем шаге.
   - `<container_name>` — имя контейнера пода PostgreSQL.
4. Подключитесь к базе данных, выполнив команду:

   ```
   psql -U postgres -d auth
   ```

5. Обновите пароль, выполнив SQL-запрос:

   ```
   ALTER USER <postgres_user> WITH PASSWORD '<new_password>';
   ```

   Здесь:

   - `<postgres_user>` — имя пользователя.
   - `<new_password>` — новый пароль.

   После смены пароля сервисы, использующие PostgreSQL, перестанут работать до обновления секретов в Kubernetes.

6. Измените секреты Kubernetes `evo.core.global`, `evo.streams.global` и `evo.infra.global`, содержащие пароль PostgreSQL. Для этого выполните следующие действия для каждого секрета:

   a. Закодируйте пароль в base64, например, с помощью команды:

```
echo -n <new_password> | base64
```

Здесь:

- `<new_password>` — новый пароль.

b. Откройте необходимый секрет для редактирования, выполнив команду:

```
kubectl -n <namespace> edit secret <secret_name>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<secret_name>` — имя секрета: `evo.core.global`, `evo.streams.global` и `evo.infra.global`.

c. Замените значение ключа `POSTGRES_PASSWORD` в блоке `data` на пароль, закодированный ранее.

d. Обновите значение ключа `POSTGRES_URL`. Для этого добавьте новый незакодированный пароль в следующий URL вместо переменной `<new_password>`:

```
postgres://<postgres_user>:<new_password>@<postgres_host>:<postgres_port>
```

Здесь:

- `<postgres_user>` — декодированное значение ключа `POSTGRES_USER`.
- `<new_password>` — пароль пользователя PostgreSQL.
- `<postgres_host>` — декодированное значение ключа `POSTGRES_HOST`.
- `<postgres_port>` — декодированное значение ключа `POSTGRES_PORT`.

▼ *Пример декодированного значения ключа* `POSTGRES_URL`

```
postgres://user:password@postgresql:0000
```

e. Закодируйте полученный URL сначала в кодировку URLEncode, а затем в base64. Например, вы можете сделать это с помощью утилиты jq, выполнив команду:

```
echo -n
'postgres://<postgres_user>:<new_password>@<postgres_host>:<postgres_port
>' | jq -rR '@uri' | base64
```

f. Сохраните изменения и закройте конфигурацию секрета.



Вы можете проверить, что значение ключа было обновлено, с

помощью команды:

```
kubectl -n <namespace> get secret <secret_name>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<secret_name>` — название редактируемого секрета.

7. Перезапустите связанные контроллеры Deployment, выполнив команды:

```
kubectl -n <namespace> rollout restart deployment/evo.core.access
kubectl -n <namespace> rollout restart deployment/evo.core.mf-feed
kubectl -n <namespace> rollout restart deployment/evo.streams.active-list-
manager
kubectl -n <namespace> rollout restart deployment/evo.streams.dashboard-
manager
kubectl -n <namespace> rollout restart deployment/evo.streams.driver-manager
kubectl -n <namespace> rollout restart deployment/evo.streams.entity-
relation-manager
kubectl -n <namespace> rollout restart deployment/evo.streams.event-model-
manager
kubectl -n <namespace> rollout restart deployment/evo.streams.event-storage-
manager
kubectl -n <namespace> rollout restart deployment/evo.streams.expertise-
manager
kubectl -n <namespace> rollout restart deployment/evo.streams.integration-
manager
kubectl -n <namespace> rollout restart deployment/evo.streams.query-log-
manager
kubectl -n <namespace> rollout restart deployment/evo.streams.secret-manager
kubectl -n <namespace> rollout restart deployment/evo.streams.settings-
manager
kubectl -n <namespace> rollout restart deployment/evo.streams.template-
manager
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

8. Проверьте состояние подов, контроллеры которых были перезапущены:

```
kubectl -n <namespace> get pods
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Названия необходимых подов начинаются с названий их контроллеров.

Через Lens

1. Подключитесь к кластеру.
2. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Pods**.
3. Выберите под PostgreSQL. В правой части экрана откроется карточка управления подом.



Название пода PostgreSQL начинается с `postgresql-`, например, `postgresql-0`.

4. Нажмите на кнопку **Shell** на панели управления подом. Откроется вкладка терминала с подключением к поду PosgtreSQL.
5. Подключитесь к базе данных, выполнив команду:

```
psql -U postgres -d auth
```

6. Обновите пароль, выполнив SQL-запрос:

```
ALTER USER <postgres_user> WITH PASSWORD '<new_password>';
```

Здесь:

- `<postgres_user>` — имя пользователя.
- `<new_password>` — новый пароль.



После смены пароля сервисы, использующие PostgreSQL, перестанут работать до обновления секретов в Kubernetes.

7. Измените секреты Kubernetes `evo.core.global`, `evo.streams.global` и `evo.infra.global`, содержащие пароль PostgreSQL. Для этого выполните следующие действия для каждого секрета:
   a. Закодируйте пароль в base64, например, с помощью команды в окне терминала:

```
echo -n <new_password> | base64
```

   Здесь:

   - `<new_password>` — новый пароль.

   b. В выпадающем меню свойств кластера выберите пункт **Config** → **Secrets**.
   c. Найдите секрет `evo.core.global` в списке и нажмите на него. В правой части экрана откроется карточка с информацией этого секрета.
   d. Нажмите на кнопку **Edit** на панели управления секретом. Откроется вкладка терминала с конфигурацией секрета.

e. Замените значение ключа `POSTGRES_PASSWORD` в блоке `data` на пароль, закодированный ранее.

f. Обновите значение ключа `POSTGRES_URL`. Для этого добавьте новый незакодированный пароль в следующий URL вместо переменной `<new_password>`:

```
postgres://<postgres_user>:<new_password>@<postgres_host>:<postgres_port>
```

Здесь:

- `<postgres_user>` — декодированное значение ключа `POSTGRES_USER`.
- `<new_password>` — пароль пользователя PostgreSQL.
- `<postgres_host>` — декодированное значение ключа `POSTGRES_HOST`.
- `<postgres_port>` — декодированное значение ключа `POSTGRES_PORT`.

▼ *Пример декодированного значения ключа* `POSTGRES_URL`

```
postgres://user:password@postgresql:0000
```

g. Закодируйте полученный URL сначала в кодировку URLEncode, а затем в base64. Например, вы можете сделать это с помощью утилиты jq, выполнив команду:

```
echo -n
'postgres://<postgres_user>:<new_password>@<postgres_host>:<postgres_port
>' | jq -rR '@uri' | base64
```

h. Сохраните внесенные изменения в конфигурации секрета.

i. Повторите вышеперечисленные шаги для изменения пароля в секретах `evo.streams.global` и `evo.infra.global`.

8. Перезапустите связанные контроллеры Deployment. Для этого:

a. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.

b. Найдите следующие контроллеры Deployment:

- `evo.core.access`;
- `evo.core.mf-feed`;
- `evo.streams.active-list-manager`;
- `evo.streams.dashboard-manager`;
- `evo.streams.driver-manager`;
- `evo.streams.entity-relation-manager`;
- `evo.streams.event-model-manager`;
- `evo.streams.event-storage-manager`;
- `evo.streams.expertise-manager`;

- `evo.streams.integration-manager`;
- `evo.streams.query-log-manager`;
- `evo.streams.secret-manager`;
- `evo.streams.settings-manager`;
- `evo.streams.template-manager`.

c. Откройте карточку каждого контроллера и нажмите на кнопку **Restart**.

d. Дождитесь перезапуска всех контроллеров.

9. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Pods**.
10. Проверьте, что все связанные поды корректно перезапустились.



Названия подов, связанных с контроллерами Deployment, начинаются с названий этих контроллеров.

Смена пароля Grafana



Для смены пароля в Grafana необходимо:

1. Подключиться к сервису Grafana локально или открыть доступ для всех пользователей.
2. Узнать данные для первичного входа.

Чтобы обновить пароль пользователя Grafana:

1. Перейдите по адресу `https://<domain_name>/grafana` и авторизуйтесь в сервисе с помощью учетных данных.

   Здесь:

   - `<domain_name>` — доменное имя системы.

2. Нажмите на аватар пользователя в правом верхнем углу. Откроется меню действий.
3. Выберите опцию **Change password**. Откроется страница изменения пароля.
4. Измените пароль в веб-интерфейсе Grafana, введя старый и новый пароль в соответствующие поля.
5. Измените пароль пользователя в секрете `kube-prometheus-stack-grafana`. Для этого:

   a. Зашифруйте новый пароль в кодировке base64, например, используя команду:

   Windows

   Используйте командную оболочку PowerShell:

   ```
   [Convert]::ToBase64String([text.Encoding]::UTF8.GetBytes("<new_password>"
   ))
   ```

Linux

```
echo -n <new_password> | base64
```

Здесь:

- `<new_password>` — новый пароль.

b. Подключитесь к кластеру.

c. Откройте конфигурацию секрета `kube-prometheus-stack-grafana`, выполнив команду:

```
kubectl edit secret -n kube-prometheus-stack kube-prometheus-stack-grafana
```

d. Введите новый пароль в поле `admin-password` в блоке `metadata.data` в кодировке base64.

Изменение доменного имени установленного экземпляра системы

Чтобы изменить доменное имя уже установленного экземпляра системы без потери имеющихся в нем данных, выполните следующие действия:

1. Подключитесь к терминалу master-узла кластера Kubernetes, в котором установлен основной экземпляр системы.
2. Обновите доменное имя в секрете `evo.infra.global`, выполнив следующую команду:

```
kubectl patch secret evo.infra.global -n <namespace>
-p='{"stringData":{"FRONTEND_HOST": "<new_domain_name>","GRAFANA_EXTERNAL_URL":
"https://<new_domain_name>/grafana"}}'
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<new_domain_name>` — новое доменное имя, например, `evo-new.company.local`.

В результате успешного обновления секрета должно отобразиться следующее сообщение:

```
secret/evo.infra.global patched
```

3. Запустите обновление конфигурации системы, выполнив следующую команду:

```
/opt/r-vision/common/bin/evoctl instance update --skip-images --skip-expertise --no
-ansible
```

В результате успешного обновления конфигурации должно отобразиться следующее сообщение:

```
The web interface is available by URL:
  https://<new_domain_name>
```

Здесь:

- <new_domain_name> — новое доменное имя, заданное на предыдущем шаге.

Настройка отдельного диска под образы контейнеров

Вы можете выделить отдельный диск под хранение образов контейнеров на каждом worker-узле. Это обезопасит следующие ресурсы системы:

- Неиспользуемые образы от удаления сборщиком мусора (garbage collector), если в файловой системе worker-узла останется меньше 15% свободного места.
- Запущенные поды от вытеснения, если в файловой системе worker-узла останется меньше 1% свободного места.

Представленные пороги системы Kubernetes являются значениями по умолчанию.

Настройка отдельного диска для образов контейнеров состоит из следующих этапов:

1. Редактирование конфигураций контроллеров DaemonSet
2. Настройка отдельного диска на worker-узлах



Редактирование конфигураций контроллеров DaemonSet делается один раз в самом начале. Настройку отдельного диска для образов необходимо провести для каждого worker-узла.

Редактирование конфигураций контроллеров DaemonSet



Редактирование конфигураций контроллеров DaemonSet производится один раз в самом начале.

Чтобы изменить контроллеры DaemonSet:

1. Подключитесь к кластеру.
2. Измените контроллеры DaemonSet, выполнив команды:

```
cat << EOF > daemonset-affinity-patch.yml
spec:
  template:
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: marked-for-drain
```

```
                operator: NotIn
                values:
                - "true"
EOF
kubectl patch daemonsets.apps -n <namespace> vector-daemon --type=merge -p "$(cat
daemonset-affinity-patch.yml)"
kubectl patch daemonsets.apps -n ingress-nginx ingress-nginx-controller
--type=merge -p "$(cat daemonset-affinity-patch.yml)"
kubectl patch daemonsets.apps -n metallb-system metallb-speaker --type=merge -p
"$(cat daemonset-affinity-patch.yml)"
kubectl patch daemonsets.apps -n kube-prometheus-stack promtail --type=merge -p
"$(cat daemonset-affinity-patch.yml)"
kubectl patch daemonsets.apps -n kube-prometheus-stack loki-canary --type=merge -p
"$(cat daemonset-affinity-patch.yml)"
kubectl patch daemonsets.apps -n kube-prometheus-stack kube-prometheus-stack-
prometheus-node-exporter --type=merge -p "$(cat daemonset-affinity-patch.yml)"
kubectl patch daemonsets.apps -n kube-system calico-node --type=merge -p "$(cat
daemonset-affinity-patch.yml)"
kubectl patch daemonsets.apps -n kube-system kube-proxy --type=merge -p "$(cat
daemonset-affinity-patch.yml)"
kubectl patch daemonsets.apps -n kube-system nodelocaldns --type=merge -p "$(cat
daemonset-affinity-patch.yml)"
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.



Изменение контроллеров DaemonSet позволяет приостановить создание и распределение подов этих контроллеров на worker-узле с меткой `marked-for-drain`.

Настройка отдельного диска на worker-узлах

Настройка отдельного диска на worker-узлах состоит из следующих этапов:

1. Подготовка worker-узла к настройке отдельного диска
2. Настройка отдельного диска и перенос образов контейнеров
3. Запуск служб и возвращение worker-узла в рабочее состояние



Представленные действия необходимо выполнить для каждого worker-узла.

Подготовка worker-узла к настройке отдельного диска

На данном этапе происходит отключение необходимых служб и вытеснение подов на выбранном worker-узле. Для этого:

1. Подключитесь к кластеру.
2. Проверьте все поды выбранного worker-узла, выполнив команду:

   ```
   kubectl get pods --field-selector spec.nodeName=<node_name> -A -o wide
   ```

   Здесь:

   - `<node_name>` — название выбранного worker-узла.

   Проверка подов необходима, чтобы обозначить, какие из них будут недоступны при их перераспределении на другие worker-узлы. Из-за этого часть функционала системы может быть недоступна на время переноса образов контейнеров на отдельный диск.

3. Переведите выбранный worker-узел в drain-режим, выполнив команду:

   ```
   kubectl drain <node_name> --delete-emptydir-data
   ```

   Здесь:

   - `<node_name>` — название выбранного worker-узла.

   Операция drain позволяет перераспределить все поды с выбранного worker-узла.

   В процессе выполнения операции worker-узел помечается как Unschedulable (непланируемый) — это предотвращает появление на worker-узле новых подов. Затем начинается процесс вытеснения (evict) подов с worker-узла. Во время этого процесса система завершает работу запущенных контейнеров.

4. Установите метку `marked-for-drain` для выбранного worker-узла, выполнив команду:

   ```
   kubectl label nodes <node_name> marked-for-drain=true
   ```

   Здесь:

   - `<node_name>` — название выбранного worker-узла.

5. Подключитесь к выбранному worker-узлу.
6. После того как Kubernetes вытеснил все поды с worker-узла, удалите под `nginx-proxy`, так как он используется для взаимодействия с kube-apiserver. Для этого:

   a. Перенесите манифест пода в домашнюю директорию на выбранном worker-узле, выполнив команду:

   ```
   sudo mv /etc/kubernetes/manifests/nginx-proxy.yml /root/
   ```



Перенос файла манифеста необходим, чтобы закрыть kubelet доступ к файлу. Kubelet отслеживает статус пода `nginx-proxy`. Если удалить под `nginx-proxy`, то kubelet заново запустит его из файла манифеста.

b. Удалите под `nginx-proxy`, выполнив команду:

```
kubectl delete pod -n kube-system nginx-proxy-<node_name> --force
```

Здесь:

- `<node_name>` — название выбранного worker-узла.

7. Остановите kubelet на worker-узле, выполнив команду:

```
sudo systemctl stop kubelet
```

8. Остановите containerd на worker-узле, выполнив команду:

```
sudo systemctl stop containerd
```

Настройка отдельного диска и перенос образов контейнеров

В данном разделе приведены шаги по переносу содержимого containerd на отдельный том на примере виртуального диска, эмулирующего физическое устройство.

Чтобы настроить отдельный диск с образами контейнеров:

1. Подключитесь к кластеру.
2. Проверьте размер текущего диска, выполнив команду:

```
sudo fdisk -l /dev/sda
```



Если измененный размер не отображается, запустите принудительное сканирование этого диска, выполнив команду:

```
sudo sh -c "echo 1 > /sys/block/sda/device/rescan"
```

После сканирования снова проверьте размер текущего диска, выполнив команду:

```
sudo fdisk -l /dev/sda
```

3. Инициируйте создание нового раздела для образов контейнеров, выполнив команду:

```
sudo fdisk /dev/sda
```

После выполнения команды откроется утилита fdisk для управления разделами диска. В утилите создайте новый диск для образов контейнера. Для этого:

a. Создайте новый раздел диска, введя значение `n` и нажав клавишу ENTER. Отобразятся поля для первичной настройки и создания раздела:
   i. Введите порядковый номер (partition number) нового раздела диска и нажмите клавишу ENTER.
   ii. Введите первый сектор (first sector) доступного пространства раздела и нажмите клавишу ENTER.
   iii. Введите последний сектор (last sector) доступного пространства раздела и нажмите клавишу ENTER.
b. Проверьте наличие нового раздела, введя значение `p` и нажав клавишу ENTER. Отобразится список доступных разделов диска.
c. Измените тип нового раздела на LVM, введя значение `t` и нажав клавишу ENTER. Отобразятся поля для смены типа раздела:
   i. Введите порядковый номер (partition number) созданного раздела и нажмите клавишу ENTER.
   ii. Введите значение `30` и нажмите клавишу ENTER. Тип созданного раздела изменится на LVM.
d. Проверьте, что тип корректно изменился, введя значение `p` и нажав клавишу ENTER. Отобразится список разделов. В столбце `TYPE` должно отображаться соответствующее значение.
e. Сохраните изменения и закройте утилиту fdisk, введя значение `w` и нажав клавишу ENTER.



Чтобы вывести список доступных команд утилиты fdisk, введите значение `m` в командную строку и нажмите клавишу ENTER.

4. Создайте новую группу томов, выполнив команду:

```
sudo vgcreate vg_containerd /dev/<disk_name>
```

Здесь:

- `<disk_name>` — название раздела, созданного на предыдущем шаге.



Чтобы отобразить все созданные физические тома LVM, выполните команду:

```
sudo pvdisplay
```

Чтобы отобразить все группы томов, выполните команду:

```
sudo vgdisplay
```

5. Создайте логический том, выполнив команду:

```
sudo lvcreate -l 100%FREE -n lv_containerd vg_containerd
```



Чтобы отобразить все логические тома, выполните команду:

```
sudo lvdisplay
```

6. Отформатируйте LVM-раздел в необходимую файловую систему:

```
sudo mkfs.<fs_tag> /dev/vg_containerd/lv_containerd
```

Здесь:

- `<fs_tag>` — тег файловой системы.

7. Создайте временную директорию для образов контейнеров, выполнив команду:

```
sudo mkdir -p /mnt/containerd
```

8. Смонтируйте логический том `lv_containerd` во временную директорию, выполнив команду:

```
sudo mount /dev/vg_containerd/lv_containerd /mnt/containerd
```

9. Проверьте размер директории с образами контейнеров, выполнив команду:

```
du -sh /var/lib/containerd
```

10. Перенесите во временную директорию содержимое директории `/var/lib/containerd`, выполнив команду:

```
sudo rsync -a /var/lib/containerd/ /mnt/containerd/
```



Процесс переноса может занять определенное время в зависимости от размера каталога и производительности диска.

11. Размонтируйте временный том `/mnt/containerd` и смонтируйте его в `/var/lib/containerd`, выполнив команду:

```
sudo umount /mnt/containerd
echo "/dev/mapper/vg_containerd-lv_containerd /var/lib/containerd    <fs-tag>
nodev,defaults        0       0" >> /etc/fstab
mount -a
```

Здесь:

- `<fs_tag>` — тег файловой системы.

Запуск служб и возвращение worker-узла в рабочее состояние

Чтобы вернуть worker-узел в рабочее состояние:

1. Подключитесь к кластеру.
2. Подключитесь к выбранному worker-узлу.
3. Запустите службу containerd, выполнив команду:

```
sudo systemctl start containerd
```

4. Запустите службу kubelet, выполнив команду:

```
sudo systemctl start kubelet
```

5. Верните манифест `nginx-proxy` в изначальную директорию, выполнив команду:

```
sudo mv /root/nginx-proxy.yml /etc/kubernetes/manifests/
```



Kubelet автоматически создаст под `nginx-proxy` из файла манифеста. Дождитесь успешного запуска пода `nginx-proxy`.

6. Снимите метку `marked-for-drain` с выбранного worker-узла, выполнив команду:

```
kubectl label nodes <node_name> marked-for-drain-
```

Здесь:

- `<node_name>` — название выбранного worker-узла.

7. Сделайте worker-узел доступным для назначения подов, выполнив команду:

```
kubectl uncordon <node_name>
```

Здесь:

- `<node_name>` — название выбранного worker-узла.

8. Проверьте, что все поды worker-узла корректно запустились, выполнив команду:

```
kubectl get pods --field-selector spec.nodeName=<node_name> -A -o wide
```

Здесь:

- `<node_name>` — название выбранного worker-узла.

Отобразится список всех подов worker-узла. Убедитесь, что поды контроллеров DaemonSet корректно запустились после снятия метки `marked-for-drain` с worker-узла.



После возвращения worker-узла в рабочее состояние под `ingress-nginx-controller` может не запуститься. Подробности по решению данной проблемы приведены в разделе Ошибки при запуске пода ingress-nginx-controller.

Ошибки при запуске пода ingress-nginx-controller

После переноса образов контейнеров на отдельный диск и восстановления работоспособности worker-узла может не запускаться под `ingress-nginx-controller`.

Может возникнуть ошибка, при которой сбросились разрешения (capabilities) бинарных файлов после переноса содержимого `containerd`. Работа по исправлению ошибки делится на следующие этапы:

1. Подтверждение ошибки с разрешениями бинарных файлов
2. Исправление ошибки разрешений бинарных файлов пода ingress-nginx-controller

Подтверждение ошибки с разрешениями бинарных файлов

Чтобы подтвердить ошибку:

1. Подключитесь к кластеру.
2. Проверьте логи пода `ingress-nginx-controller`. Для этого:
   a. Получите полное название пода с информацией по worker-узлам, выполнив команду:

```
kubectl get pods -n ingress-nginx -o wide
```

Отобразится список всех подов `ingress-nginx-controller`.

▼ *Пример вывода списка всех подов*

```
NAME                             READY   STATUS    RESTARTS         AGE   IP
NODE
ingress-nginx-controller-j25z6   1/1     Running   9 (5d15h ago)    9d
00.000.1.000   siem-testing-w03
ingress-nginx-controller-ppp6c   1/1     Pending   15 (5d15h ago)   41d
```

```
00.000.2.000   siem-testing-w02
ingress-nginx-controller-wltdd   1/1     Running   7 (5d15h ago)    41d
00.000.3.000   siem-testing-w01
```

b. Скопируйте название пода на worker-узле, с которым производились работы.

c. Откройте логи пода `ingress-nginx-controller`, выполнив команду:

```
kubectl logs -n ingress-nginx <pod_name>
```

Здесь:

- `<pod_name>` — название пода, полученное на предыдущем шаге.

3. Найдите в логах следующую запись:

```
port 80 is already in use. Please check the flag --http-port
```



Если вы не нашли в логах упомянутую выше запись, то ошибка запуска пода `ingress-nginx-controller` возникла не из-за сброса разрешения бинарных файлов. Для решения проблемы обратитесь в службу поддержки по адресу support@rvision.ru.

4. Проверьте разрешения бинарных файлов, выполнив команду:

```
getcap $(find /var/lib/containerd/ -name nginx-ingress-controller -type f)
```

Если в разрешении `ingress-nginx-controller` отображается значение `cap_net_bind_service+ep`, то он работает корректно. Если отображается любое другое значение, то у пода `ingress-nginx-controller` сбросились разрешения бинарных файлов.

Исправление ошибки разрешений бинарных файлов пода ingress-nginx-controller

Чтобы исправить ошибку:

1. Подключитесь к кластеру.
2. Установите метку `marked-for-drain` для worker-узла, выполнив команду:

```
kubectl label nodes <node_name> marked-for-drain=true
```

Здесь:

- `<node_name>` — название выбранного worker-узла.



Предварительно необходимо скорректировать конфигурации контроллеров DaemonSet. Подробности приведены в разделе Редактирование конфигураций

контроллеров DaemonSet.



Объединенная файловая система (OverlayFS) запущенного пода `ingress-nginx-controller` примонтирована к файловой системе worker-узла. Это блокирует возможность удаления и перезаписывания слоев образа.

Установка метки `marked-for-drain` запрещает планировать на worker-узел поды контроллеров DaemonSet. Тогда Docker-образ `ingress-nginx-controller` не будет задействован ни одним запущенным контейнером на выбранном worker-узле. Блокировка замены и удаления образа будет снята.

3. Удалите под `ingress-nginx-controller`, выполнив команду:

   ```
   kubectl delete pod <pod_name> -n ingress-nginx --field-selector
   spec.nodeName=<node_name>
   ```

   Здесь:

   - `<pod_name>` — название пода, полученное на этапе подтверждения ошибки.

4. Подключитесь к выбранному worker-узлу.
5. Удалите образ пода `ingress-nginx-controller`, выполнив команду:

   ```
   crictl rmi harbor.rvision.pro/third_party/ingress-nginx/controller:1.11.3-
   distroless-20250428
   ```

6. Проверьте, что образ корректно удалился и не отображается в общем списке, с помощью команды:

   ```
   crictl images | grep ingress
   ```

7. Подключитесь к серверу, на котором производилась установка системы.
8. Распространите образ пода `ingress-nginx-controller` по всему кластеру, выполнив команду:

   ```
   /opt/r-vision/common/bin/evoctl instance upload --archive ingress-nginx.img
   ```

9. Проверьте наличие нового образа, выполнив команду:

   ```
   crictl images | grep ingress
   ```

10. Проверьте корректность разрешений бинарных файлов, выполнив команду:

    ```
    getcap $(find /var/lib/containerd/ -name nginx-ingress-controller -type f)
    ```

11. Снимите метку `marked-for-drain` с выбранного worker-узла, выполнив команду:

    ```
    kubectl label nodes <node_name> marked-for-drain-
    ```

    Здесь:

    - `<node_name>` — название выбранного worker-узла.

12. Дождитесь запуска пода `ingress-nginx-controller` и проверьте его статус, выполнив команду:

    ```
    kubectl get pods -n ingress-nginx --field-selector spec.nodeName=<node_name>
    ```

    Здесь:

    - `<node_name>` — название выбранного worker-узла.

Поиск установленных экземпляров системы

Чтобы получить информацию об экземплярах системы, установленных в кластере Kubernetes и/или удаленных из него, воспользуйтесь опцией `instance find` утилиты evoctl:

1. Подключитесь к master-узлу кластера или к машине, с которой производилась установка системы.
2. Выполните в терминале следующую команду:

    ```
    evoctl instance find -o <output_format>
    ```

    Здесь:

    - `<output_format>` — формат вывода: `table`, `yaml` или `json`.

В результате выполнения команды отобразится информация об установленных и удаленных основных экземплярах системы и сателлитах, в том числе их пространства имен, типы, а в форматах YAML и JSON — также даты установки и удаления.

Пример вывода команды:

Таблица

```
Instance types:
  main            EVO Platform main
  satellite       EVO Platform satellite
  siem            EVO Platform SIEM
  siem-legacy     SIEM legacy (<2.3.0)
  removed         EVO Platform removed

 NS       |  TYPE |  VERSION |  VENDOR  |  PRODUCT          |  FRONTEND
```

```
|  SPACE_ID
siem      | siem  | 2.6.0    | R-Vision | EVO Platform SIEM |
siem.company.local      | space-main
siem-test | siem  | 2.7.1    | R-Vision | EVO Platform SIEM |
test.siem.company.local | space-test
```

YAML

```
Instance types:
  main            EVO Platform main
  satellite       EVO Platform satellite
  siem            EVO Platform SIEM
  siem-legacy     SIEM legacy (<2.3.0)
  removed         EVO Platform removed

- namespace: siem
  type: siem
  version: 2.6.0
  vendor: R-Vision
  product: EVO Platform SIEM
  frontendHost: siem.company.local
  spaceId: space-main
  installed: "2025-10-10T12:42:59+00:00"
  updated: "2025-12-10T04:57:07+00:00"
  removed: ""
- namespace: siem-test
  type: siem
  version: 2.7.1
  vendor: R-Vision
  product: EVO Platform SIEM
  frontendHost: test.siem.company.local
  spaceId: space-test
  installed: "2025-12-18T15:11:04+00:00"
  updated: "2026-04-08T08:35:52+00:00"
  removed: ""
```

JSON

```
Instance types:
  main            EVO Platform main
  satellite       EVO Platform satellite
  siem            EVO Platform SIEM
  siem-legacy     SIEM legacy (<2.3.0)
  removed         EVO Platform removed

[
  {
    "namespace": "siem",
    "type": "siem",
    "version": "2.6.0",
```

```
    "vendor": "R-Vision",
    "product": "EVO Platform SIEM",
    "frontendHost": "siem.company.local",
    "spaceId": "space-main",
    "installed": "2025-10-10T12:42:59+00:00",
    "updated": "2025-12-10T04:57:07+00:00",
    "removed": ""
  },
  {
    "namespace": "siem-test",
    "type": "siem",
    "version": "2.7.1",
    "vendor": "R-Vision",
    "product": "EVO Platform SIEM",
    "frontendHost": "test.siem.company.local",
    "spaceId": "space-test",
    "installed": "2025-12-18T15:11:04+00:00",
    "updated": "2026-04-08T08:35:52+00:00",
    "removed": ""
  }
]
```


Настройка системы

Перед началом работы с системой можно выполнить следующие настройки:

- Настройка быстрого доступа к продуктам R-Vision
- Настройка формата даты
- Настройка времени хранения оповещений
- Изменение пула адресов в MetalLB
- Настройка времени хранения запросов
- Настройка ограничений размера файла драйвера БД
- Настройка ограничений размера загружаемых файлов
- Настройка получения событий по произвольному порту

Настройка быстрого доступа к продуктам R-Vision

В данном разделе описана настройка быстрого доступа к продуктам R-Vision SOAR и R-Vision UEBA из интерфейса системы. В системе эти продукты представлены в разделе **Инструменты** как **SOAR** и **UEBA** соответственно.



Настройка интеграций с внешними системами на получение или отправку событий и отправку оповещений по почте описана в разделе Работа с интеграциями.

Для отображения ссылок на продукты R-Vision в веб-интерфейсе системы требуется указать адреса

подключения к этим продуктам в настройках сервера.



Настройки сервера уникальны для каждого тенанта системы.

Чтобы задать адреса для подключения к продуктам:

1. Откройте выпадающее меню учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки сервера** (). Откроется окно для настройки подключения.
3. Введите адреса серверов продуктов в соответствующие поля:
   - **SOAR URL** — адрес сервера R-Vision SOAR;
   - **UEBA URL** — адрес сервера R-Vision UEBA.
4. Нажмите на кнопку **Сохранить**.

   При успешном подключении разделы **SOAR** (R-Vision SOAR) и **UEBA** (R-Vision UEBA) станут доступны в меню **Инструменты**.



Разделы станут доступны после перезагрузки страницы или нового входа в систему.

Настройка формата даты

Администратор системы может настроить форматы дат, включая добавление новых пользовательских форматов, удаление существующих или редактирование текущих через переменную окружения в системе.

Изменение формата даты

Чтобы изменить формат даты:

Через командную строку

1. Подключитесь к кластеру.
2. Откройте ConfigMap `evo.core-host-app-env` для редактирования, выполнив команду:

   ```
   kubectl edit configmap evo.core-host-app-env -n <namespace>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

3. Добавьте новый формат даты в JSON-массив поля `DATE_FORMATS` в блоке `data`.

```
"[..., {\"label\":\"<date_description>\", \"value\":\"<date_format>\"}]"
```

Здесь:

- `<date_description>` — описание формата даты, которое отображается в интерфейсе системы при выборе формата даты в настройках профиля пользователя.
- `<date_format>` — формат даты и времени, соответствующий стандартам форматирования библиотеки Luxon.



- Если вы хотите изменить существующий формат даты, то замените значения переменных `<date_description>` и `<date_format>` в существующем формате даты.
- Если вы хотите удалить формат даты, то удалите объект с полями `label` и `value` существующего формата даты из массива опции `DATE_FORMATS`.

▼ *Пример нескольких форматов даты*

```
[
{\"label\":\"YYYY-MM-DDThh:mm:ss+hh:mm\",
\"value\":\"yyyy-MM-dd'T'HH:mm:ss.SSSZZ\"},
{\"label\":\"DD.MM.YYYY HH:mm:ss\", \"value\":\"dd.MM.yyyy
HH:mm:ss\"},
{\"label\":\"DD.MM.YYYY\", \"value\":\"dd.MM.yyyy\"}
]
```

4. Сохраните и закройте ConfigMap `evo.core-host-app-env`.
5. Перезапустите контроллер Deployment `evo.core.host-app`, выполнив команду:

```
kubectl -n <namespace> rollout restart deployment evo.core.host-app
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

6. После перезапуска контроллера убедитесь, что изменения вступили в силу. Для этого перейдите в настройки профиля системы.

Через Lens

1. Подключитесь к кластеру.
2. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
3. Выберите ConfigMap `evo.core-host-app-env` из списка.
4. В открывшейся панели справа нажмите на кнопку **Edit**.
5. Добавьте новый формат даты в JSON-массив поля `DATE_FORMATS` в блоке `data`.

```
"[..., {\"label\":\"<date_description>\", \"value\":\"<date_format>\"}]"
```

Здесь:

- `<date_description>` — описание формата даты, которое отображается в интерфейсе системы при выборе формата даты в настройках профиля пользователя.
- `<date_format>` — формат даты и времени, соответствующий стандартам форматирования библиотеки Luxon.



- Если вы хотите изменить существующий формат даты, то замените значения переменных `<date_description>` и `<date_format>` в существующем формате даты.
- Если вы хотите удалить формат даты, то удалите объект с полями `label` и `value` существующего формата даты из массива опции `DATE_FORMATS`.

▼ *Пример нескольких форматов даты*

```
[
{\"label\":\"YYYY-MM-DDThh:mm:ss+hh:mm\",
\"value\":\"yyyy-MM-dd'T'HH:mm:ss.SSSZZ\"},
{\"label\":\"DD.MM.YYYY HH:mm:ss\", \"value\":\"dd.MM.yyyy
HH:mm:ss\"},
{\"label\":\"DD.MM.YYYY\", \"value\":\"dd.MM.yyyy\"}
]
```

6. Сохраните и закройте ConfigMap `evo.core-host-app-env`.
7. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.
8. Выберите контроллер Deployment `evo.core.host-app` из списка.
9. В открывшейся панели справа нажмите на кнопку **Restart**.
10. После перезапуска контроллера убедитесь, что изменения вступили в силу. Для этого перейдите в настройки профиля системы.

Настройка параметра `label`

Параметр `label` в конфигурации формата даты служит для отображения описания формата в пользовательском интерфейсе, что позволяет пользователям выбрать нужный формат даты в настройках своего профиля. Это описание должно быть понятным и точно отражать структуру данных, которую формат представляет.

Пример:

```
"[..., {\"label\":\"Полная дата и время\", \"value\":\"yyyy-MM-
dd'T'HH:mm:ss.SSSZZ\"}]"
```

где:

- `Полная дата и время` — описание формата даты, которое отображается в интерфейсе системы в настройках профиля пользователя при выборе формата даты.

Настройка параметра `value`

Параметр `value` в настройках формата даты определяет технический формат отображения дат и времени, который используется в системе. Формат указывается в соответствии с токенами форматирования, поддерживаемыми библиотекой Luxon.

Примеры настройки параметра `value` с расшифровкой токенов:

- Полная дата и время с часовым поясом:

```
"yyyy-MM-dd'T'HH:mm:ss.SSSZZ"
```

  - `yyyy` — четырехзначный год.
  - `MM` — двузначный месяц.
  - `dd` — двузначный день.
  - `T` — разделитель даты и времени в международном стандарте ISO 8601.
  - `HH` — часы в 24-часовом формате.
  - `mm` — минуты.
  - `ss` — секунды.
  - `SSS` — миллисекунды.
  - `ZZ` — часовой пояс в формате `+hh:mm` или `-hh:mm`.

- Дата и время без часового пояса:

```
"dd.MM.yyyy HH:mm:ss"
```

  - `dd` — двузначный день.
  - `MM` — двузначный месяц.
  - `yyyy` — четырехзначный год.
  - `HH` — часы в 24-часовом формате.
  - `mm` — минуты.
  - `ss` — секунды.

- Дата без времени:

```
"dd.MM.yyyy"
```

  - `dd` — двузначный день.

- `MM` — двузначный месяц.
- `yyyy` — четырехзначный год.

Настройка времени хранения оповещений

Срок хранения оповещений можно настроить через переменную окружения в кластере K8s.

Чтобы задать срок хранения оповещений:

Через командную строку

1. Подключитесь к кластеру.
2. Измените ConfigMap `evo.streams-alert-manager-env`, выполнив команду:

   ```
   kubectl patch configmap -n <namespace> evo.streams-alert-manager-env -p
   '{"data": {"EVO_STREAMS_ALERT_MANAGER_ALERT_LIFETIME_INTERVAL":
   "<lifetime_interval>"}}'
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

     Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

   - `<lifetime_interval>` — срок хранения оповещений, например: `1h`, `7d` или `30d`.
3. Перезапустите контроллер Deployment `evo.streams.alert-manager`, выполнив команду:

   ```
   kubectl -n <namespace> rollout restart deployment evo.streams.alert-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
4. Проверьте, что под контроллера Deployment `evo.streams.alert-manager` корректно перезапустился, с помощью команды:

   ```
   kubectl get pods -n <namespace> -l
   app.kubernetes.io/instance=evo.streams.alert-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

Через Lens

1. Подключитесь к кластеру.

2. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
3. Выберите ConfigMap `evo.streams-alert-manager-env` из списка.
4. В открывшейся панели справа задайте срок хранения оповещений в поле переменной `EVO_STREAMS_ALERT_MANAGER_ALERT_LIFETIME_INTERVAL`.

   Формат:

   ```
   EVO_STREAMS_ALERT_MANAGER_ALERT_LIFETIME_INTERVAL: <lifetime_interval>
   ```

   Здесь:

   - `<lifetime_interval>` — срок хранения оповещений, например: `1h`, `7d` или `30d`.

5. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.
6. Выберите контроллер Deployment `evo.streams.alert-manager`. В правой части экрана откроется панель конфигурации контроллера.
7. Нажмите на кнопку **Restart** для перезагрузки контроллера Deployment `evo.streams.alert-manager`.
8. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Pods**.
9. Проверьте, что под контроллера Deployment `evo.streams.alert-manager` корректно запустился и находится в статусе `Running`. Название пода начинается с "evo.streams.alert-manager".

Изменение пула адресов в MetalLB

В состав дистрибутива системы включен балансировщик нагрузки MetalLB. В данном разделе описан процесс изменения пула адресов, который он выделяет сервисам кластера.

С помощью командной строки

Чтобы изменить пул адресов MetalLB с помощью командной строки:

1. Получите список пулов адресов кластера, выполнив следующую команду:

   ```
   kubectl get IPAddressPools -n metallb-system
   ```

2. Определите пул, который нужно изменить. В примере ниже префикс имени пула совпадает с именем пространства имен кластера (`system`).

   ▼ *Пример вывода списка пулов адресов кластера*

   ```
   NAME          AGE
   system-pool   146d
   ```

3. Откройте конфигурацию пула, выполнив следующую команду:

```
kubectl edit IPAddressPools <pool_name> -n metallb-system
```

Здесь:

- `<pool_name>` — имя пула, полученное на шаге 2.

4. Отредактируйте ключ `spec.addresses` в конфигурационном файле, указав новый пул адресов:

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
# ...
spec:
  # отредактируйте значение этого ключа
  addresses:
  - 10.0.0.15/32
  autoAssign: true
  avoidBuggyIPs: false
```

5. Перезагрузите контроллер Deployment `metallb-controller`, чтобы применить изменения:

```
kubectl rollout restart deployment/metallb-controller
```

С помощью Lens

Чтобы изменить пул адресов MetalLB:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Custom Resources** → **metallb.io** → **IP Address Pool**.
3. Выберите пространство имен `metallb-system` в выпадающем списке **Select Namespace** в правой части экрана. Отобразится таблица пулов IP-адресов.
4. Выберите пул адресов, который необходимо отредактировать.
5. Нажмите на кнопку **Edit** в правом верхнем углу экрана. В нижней части экрана отобразится файл конфигурации пула адресов.
6. Отредактируйте значение ключа `spec.addresses` аналогично п. 4 инструкции выше.
7. Перезагрузите контроллер Deployment `metallb-controller`:
   a. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.
   b. Выберите пространство имен `metallb-system` в выпадающем списке **Select Namespace** в правой части экрана. Отобразится список контроллеров Deployment.
   c. В строке таблицы, где находится контроллер Deployment `metallb-controller`, нажмите на кнопку ⋮ в правом столбце и в выпадающем меню выберите пункт **Restart**.

Настройка времени хранения запросов

Срок хранения истории запросов можно настроить через переменную окружения в кластере Kubernetes.

Чтобы задать срок хранения истории запросов:

Через командную строку

1. Подключитесь к кластеру.
2. Измените время хранения запросов, выполнив команду:

   ```
   kubectl patch configmap -n <namespace> evo.streams-query-log-manager-env -p
   '{"data": {"EVO_STREAMS_QUERY_LOG_MANAGER_QUERY_LOG_LIFETIME_INTERVAL":
   "<lifetime_interval>"}}'
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

     Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

   - `<lifetime_interval>` — временной интервал хранения запросов. Например: `30m`, `10h`, `2d` или `5d`.
3. Перезапустите контроллер Deployment `evo.streams.query-log-manager`, выполнив команду:

   ```
   kubectl -n <namespace> rollout restart deployment evo.streams.query-log-
   manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
4. Убедитесь, что под `evo.streams.query-log-manager` корректно перезапустился, выполнив команду:

   ```
   kubectl get pods -n <namespace> -l
   app.kubernetes.io/instance=evo.streams.query-log-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

Через Lens

1. Подключитесь к кластеру.

2. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
3. Выберите ConfigMap `evo.streams-query-log-manager-env` из списка.
4. В открывшейся панели справа задайте срок хранения истории запросов в поле переменной `EVO_STREAMS_QUERY_LOG_MANAGER_QUERY_LOG_LIFETIME_INTERVAL`.

   Формат:

   ```
   EVO_STREAMS_QUERY_LOG_MANAGER_QUERY_LOG_LIFETIME_INTERVAL:
   <lifetime_interval>
   ```

   Здесь:

   - `<lifetime_interval>` — временной интервал хранения запросов. Например: `30m`, `10h`, `2d` или `5d`.

5. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.
6. Выберите контроллер Deployment `evo.streams.query-log-manager` из списка.
7. В открывшейся панели справа нажмите на кнопку **Restart** на панели управления контроллером.
8. Убедитесь, что изменения вступили в силу:
   a. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Pods**.
   b. Убедитесь, что под `evo.streams.query-log-manager` корректно перезапустился и находится в статусе `Running`.

Настройка ограничений размера файла драйвера БД

По умолчанию система позволяет загружать файлы драйверов БД размером не более 100 000 000 байт (приблизительно 95,37 МБ).

Чтобы изменить максимальный допустимый размер файла драйвера БД, выполните следующие действия на машине, на которой запускался инсталлятор системы:

1. Откройте файл `custom_values.yaml`, хранящийся в директории `/opt/r-vision/streams/helm/evo.streams.frontend`.
2. Измените значение переменной `MAX_FILE_SIZE`.
3. Сохраните и закройте файл.
4. Разверните сервис `evo.streams.frontend` заново с помощью команды:

   ```
   /opt/r-vision/common/bin/evoctl instance update -m evo.streams -g "Streams
   applications" -s evo.streams.frontend
   ```

Настройка ограничений размера загружаемых файлов

По умолчанию система позволяет загружать файлы размером не более 1 МБ.

Вы можете увеличить ограничение размера загружаемых файлов следующими способами:

- Для контроллера ingress-nginx — при необходимости увеличить ограничение для всех объектов Ingress, связанных с контроллером ingress-nginx.
- Для конкретного объекта Ingress — при необходимости настроить ограничение для конкретных объектов Ingress.

Увеличение ограничения для контроллера ingress-nginx

Чтобы изменить максимальный размер загружаемых файлов для контроллера ingress-nginx:

1. Подключитесь к терминалу master-узла кластера Kubernetes.
2. Измените ConfigMap контроллера ingress-nginx с помощью команды:

   ```
   kubectl patch configmaps -n ingress-nginx ingress-nginx-controller -p '{"data":
   {"proxy-body-size": "<file_size>"}}'
   ```

   Здесь:

   - `<file_size>` — ограничение размера загружаемых файлов, заданное в формате "число и единица измерения информации". Например: `100m`, `50m`.
3. Перезапустите все поды, связанные с контроллером ingress-nginx, с помощью команды:

   ```
   kubectl rollout restart deployment -n ingress-nginx -l
   app.kubernetes.io/instance=ingress-nginx
   ```

   При переустановке системы ограничение размера файлов вернется к значению 1 МБ.
4. Удостоверьтесь, что файлы больше 1 МБ корректно загружаются.

Увеличение ограничения для конкретного объекта Ingress

Чтобы изменить максимальный размер загружаемых файлов для конкретного объекта Ingress:

1. Отредактируйте блок `annotations` необходимого объекта Ingress с помощью команды:

   ```
   kubectl patch ingress <ingress_name> -n <namespace> -p '{"metadata":
   {"annotations": {"nginx.ingress.kubernetes.io/proxy-body-size": "<file_size>"}}}'
   ```

   Здесь:

   - `<ingress_name>` — название объекта Ingress.

- `<namespace>` — название пространства имен, в котором установлен выбранный объект Ingress.
- `<file_size>` — ограничение размера загружаемых файлов, заданное в формате "число и единица измерения информации". Например: `100m`, `50m`.



При обновлении или переустановке системы ограничение размера файлов вернется к значению 1 МБ.

2. После автоматического перезапуска подов удостоверьтесь, что файлы больше 1 МБ корректно загружаются.

Настройка получения событий по произвольному порту

Некоторые точки входа, такие как Syslog, позволяют получать события только по портам из определенного диапазона, например, 30000—32767. Однако, если нет возможности передавать события из внешней системы на порты из этого диапазона, вы можете настроить проксирование трафика через Ingress-контроллер. В таком случае информация о событиях будет перенаправляться из порта, на который она поступает, на порт, который будет использоваться в конвейере обработки событий.

Чтобы настроить проксирование трафика TCP и UDP с данными о событиях, выполните следующие действия:

1. Подключитесь к терминалу master-узла кластера.
2. Создайте ConfigMap `tcp-services`, выполнив следующую команду:

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: tcp-services
  namespace: <namespace>
data:
  <tcp_port>: "<namespace>/wl-<collector_id>-collector-main:<entry_point_port>"
EOF
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

- `<tcp_port>` — порт, с которого перенаправляется TCP-трафик.
- `<collector_id>` — идентификатор коллектора, в котором находится точка входа.



Идентификатор можно посмотреть в поле **ID** на вкладке **Информация** в карточке коллектора.

- `<entry_point_port>` — порт точки входа, на который необходимо перенаправить TCP-трафик.

3. Создайте ConfigMap `udp-services`, выполнив следующую команду:

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: udp-services
  namespace: <namespace>
data:
  <tcp_port>: "<namespace>/wl-<collector_id>-collector-main:<entry_point_port>"
EOF
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<udp_port>` — порт, с которого перенаправляется UDP-трафик.
- `<collector_id>` — идентификатор коллектора, в котором находится точка входа.



Идентификатор можно посмотреть в поле **ID** на вкладке **Информация** в карточке коллектора.

- `<entry_point_port>` — порт точки входа, на который необходимо перенаправить UDP-трафик.

4. Добавьте созданные объекты ConfigMap в список аргументов запуска контроллера Nginx, выполнив следующие команды:

```
kubectl -n ingress-nginx patch deployment ingress-nginx-controller --type='json'
-p='[{"op": "add", "path": "/Pod Template/Containers/controller/Args", "value": "--
tcp-services-configmap=<namespace>/tcp-services"}]'
kubectl -n ingress-nginx patch deployment ingress-nginx-controller --type='json'
-p='[{"op": "add", "path": "/Pod Template/Containers/controller/Args", "value": "--
udp-services-configmap=<namespace>/udp-services"}]'
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

5. Чтобы открыть на прослушивание порты, на которые поступают события из внешней системы, добавьте их в конфигурацию сервиса `ingress-nginx-controller`:

a. Создайте файл патча `ingress-nginx-svc-controller-patch.yml` со следующим содержимым:

```
spec:
  ports:
  - name: syslog-tcp
    port: <tcp_port>
  - name: syslog-udp
```

```
    port: <udp_port>
```

Здесь:

- `<tcp_port>` — порт, с которого перенаправляется TCP-трафик.
- `<udp_port>` — порт, с которого перенаправляется UDP-трафик.

b. Примените патч к конфигурации сервиса `ingress-nginx-controller`, выполнив следующую команду:

```
kubectl -n ingress-nginx patch service ingress-nginx-controller --patch "$(cat
ingress-nginx-svc-controller-patch.yml)"
```

6. Дождитесь, пока поды контроллера Nginx перезапустятся и перейдут в статус `Running`.



Чтобы проверить статус подов контроллера Nginx, вы можете выполнить следующую команду:

```
kubectl -n ingress-nginx get pods -l
app.kubernetes.io/instance=ingress-nginx
```

7. Добавьте на конвейер точку входа со следующими свойствами:
   - **Тип точки входа**: в соответствии с типом внешней системы.
   - **Протокол**: TCP или UDP.
   - **Порт**: порт, указанный в значении ключа порта для ConfigMap `tcp-services` или `udp-services`, в соответствии с выбранным вами протоколом.
8. Включите конвейер и убедитесь, что события поступают в систему.



Если после выполнения вышеперечисленных шагов события не поступают в систему, обратитесь в службу поддержки по адресу support@rvision.ru.

Пример настройки проксирования событий

Рассмотрим следующий пример настройки проксирования трафика для точки входа Syslog:

- Точка входа находится в коллекторе с идентификатором `20483267-4a08-49b8-8abd-7389c50a805d`.
- TCP-трафик перенаправляется с порта 514 на 30000.
- UDP-трафик перенаправляется с порта 515 на 30001.
- Система установлена в пространстве имен Kubernetes `evo`.

Чтобы настроить проксирование трафика с данными о событиях, выполните следующие действия:

1. Подключитесь к терминалу master-узла кластера.
2. Создайте ConfigMap `tcp-services`, выполнив следующую команду:

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: tcp-services
  namespace: evo
data:
  514: "evo/wl-20483267-4a08-49b8-8abd-7389c50a805d-collector-main:30000"
EOF
```

3. Создайте ConfigMap `udp-services`, выполнив следующую команду:

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: udp-services
  namespace: evo
data:
  515: "evo/wl-20483267-4a08-49b8-8abd-7389c50a805d-collector-main:30001"
EOF
```

4. Добавьте созданные объекты ConfigMap в список аргументов запуска контроллера Nginx, выполнив следующие команды:

```
kubectl -n ingress-nginx patch deployment ingress-nginx-controller --type='json'
-p='[{"op": "add", "path": "/Pod Template/Containers/controller/Args", "value": "--
tcp-services-configmap=evo/tcp-services"}]'
kubectl -n ingress-nginx patch deployment ingress-nginx-controller --type='json'
-p='[{"op": "add", "path": "/Pod Template/Containers/controller/Args", "value": "--
udp-services-configmap=evo/udp-services"}]'
```

5. Чтобы открыть на прослушивание порты, на которые поступают события из внешней системы, добавьте их в конфигурацию сервиса `ingress-nginx-controller`:

   a. Создайте файл патча `ingress-nginx-svc-controller-patch.yml` со следующим содержимым:

   ```
   spec:
     ports:
     - name: syslog-tcp
       port: 514
     - name: syslog-udp
       port: 515
   ```

   b. Примените патч к конфигурации сервиса `ingress-nginx-controller`, выполнив следующую команду:

```
kubectl -n ingress-nginx patch service ingress-nginx-controller --patch "$(cat
ingress-nginx-svc-controller-patch.yml)"
```

6. Дождитесь, пока поды контроллера Nginx перезапустятся и перейдут в статус `Running`.



Чтобы проверить статус подов контроллера Nginx, вы можете выполнить следующую команду:

```
kubectl -n ingress-nginx get pods -l
app.kubernetes.io/instance=ingress-nginx
```

7. Добавьте на конвейер точку входа со следующими свойствами:
   - **Тип точки входа**: Syslog.
   - **Протокол**: TCP или UDP.
   - **Порт**:
     - Если выбран протокол TCP — 30000.
     - Если выбран протокол UDP — 30001.
8. Включите конвейер и убедитесь, что события поступают в систему.

Резервное копирование и восстановление компонентов системы

В данном разделе приведены методы резервного копирования и восстановления различных компонентов системы.

Содержание раздела:

- Резервное копирование хранилища etcd
- Резервное копирование без Backup Wrapper
  - Резервное копирование PostgreSQL без Backup Wrapper
  - Резервное копирование ClickHouse без Backup Wrapper
- Резервное копирование сервисов платформы
- Резервное копирование внутренних баз данных
- Работа с Backup Wrapper
  - Настройка PostgreSQL для работы с Backup Wrapper
  - Настройка ClickHouse для работы с Backup Wrapper
  - Настройка многоуровневого хранилища ClickHouse
  - Установка и настройка Backup Wrapper
  - Удаление Backup Wrapper

- Резервное копирование внешних баз данных
  - Настройка резервного копирования из кластера системы
  - Настройка резервного копирования вне кластера системы
- Восстановление из резервных копий
  - Восстановление внутренних баз данных
  - Восстановление внешних баз данных
    - Восстановление базы данных PostgreSQL
    - Восстановление базы данных ClickHouse
    - Устранение проблем при восстановлении внешних баз данных
  - Восстановление сервисов платформы
- Архивирование и восстановление исторических событий
  - Настройка утилиты для архивирования и восстановления исторических событий
  - Работа с утилитой для архивирования исторических событий

Резервное копирование хранилища etcd

В этом разделе описано, как осуществлять резервное копирование и восстановление из копии хранилища конфигураций etcd с помощью утилит etcdctl и etcdutl.



В разделе рассматривается работа с etcd при следующих условиях:

- кластер развернут с помощью Kubespray;
- установлен etcd версии 3.5.7 и выше;
- хранилище etcd установлено как системный сервис и управляется с помощью утилиты systemctl.

Инструкция по установке представлена в репозитории разработчика etcd.

Резервное копирование etcd

Выполните следующие действия для каждого узла, на котором установлен сервис etcd:

1. Подключитесь к кластеру.
2. Подключитесь к терминалу узла, на котором установлен сервис etcd.
3. Создайте резервную копию хранилища etcd, выполнив следующую команду:

```
source /etc/etcd.env && ETCDCTL_KEY=$ETCDCTL_KEY ETCDCTL_CACERT=$ETCDCTL_CACERT
ETCDCTL_CERT=$ETCDCTL_CERT ETCDCTL_API=3 etcdctl --endpoints 127.0.0.1:2379
snapshot save /var/backups/etcd-$(date '+%Y%m%d_%H%M%S')/snapshot.db
```



Чтобы получить расширенный лог сообщений о ходе создания резервной копии, добавьте к команде выше опцию `--debug=true`.

4. Вы можете удостовериться, что резервная копия создана успешно, с помощью команды:

```
etcdutl snapshot status /var/backups/<etcd-date>/snapshot.db -w table
```

Здесь:

- `<etcd-date>` — название директории с резервной копией etcd, например, `etcd-20260303_140322`.

▼ *Пример вывода команды*

```
+---------+----------+------------+------------+
|  HASH   | REVISION | TOTAL KEYS | TOTAL SIZE |
+---------+----------+------------+------------+
| 7ef846e |   485261 |      11642 |      94 MB |
+---------+----------+------------+------------+
```

Номер ревизии (`REVISION`) увеличивается на 1 при каждой операции записи в etcd.

Восстановление etcd из резервной копии

Способ восстановления etcd из резервной копии зависит от версии:

- Если версия etcd ниже 3.5.16, то восстановление выполняется с помощью утилиты etcdctl.
- Если версия etcd 3.5.16 или выше, то восстановление выполняется с помощью утилиты etcdutl.

Через etcdctl

Чтобы восстановить etcd из резервной копии:

1. Подключитесь к кластеру.
2. Подключитесь к терминалу узла, на котором установлен сервис etcd.
3. Выполните следующую команду:

```
source /etc/etcd.env && ETCDCTL_KEY=$ETCDCTL_KEY
ETCDCTL_CACERT=$ETCDCTL_CACERT ETCDCTL_CERT=$ETCDCTL_CERT ETCDCTL_API=3
etcdctl snapshot restore /var/backups/<etcd-date>/snapshot.db --data
-dir=<new_dir>
```

Здесь:

- `<etcd-date>` — название директории с резервной копией etcd, например, `etcd-20260303_140322`.
- `<new_dir>` — новый каталог данных etcd, например, `/var/lib/etcd_restored/`.

4. Остановите сервис etcd, выполнив следующую команду:

```
systemctl stop etcd.service
```

5. В файле `/etc/etcd.env` измените значение переменной `ETCD_DATA_DIR` на путь, указанный в качестве значения `<new_dir>` на шаге 1.
6. Запустите сервис etcd, выполнив следующую команду:

   ```
   systemctl start etcd.service
   ```

7. Убедитесь, что сервис etcd корректно запустился, с помощью команды:

   ```
   systemctl status etcd
   ```

8. При необходимости удалите каталог, в котором хранились данные etcd до восстановления, с помощью следующей команды:

   ```
   rm -rf <old_dir>
   ```

   Здесь:

   - `<old_dir>` — директория, в которой хранились данные etcd до восстановления, например, `/var/lib/etcd/`.

Через etcdutl

Для восстановления etcd из резервной копии в кластере Kubernetes следует использовать вариант revision bumps. При данном варианте контроллеры Kubernetes используют так называемые информеры (informers), которые работают как кэш. Восстановление etcd из резервной копии без использования revision bumps может привести к некорректному восстановлению кэшей и, как следствие, неопределенному поведению контроллеров.

Чтобы восстановить etcd из резервной копии:

1. Подключитесь к кластеру.
2. Подключитесь к терминалу узла, на котором установлен сервис etcd.
3. Выполните следующую команду:

   ```
   etcdutl snapshot restore /var/backups/<etcd-date>/snapshot.db --bump
   -revision <n_bumps> --mark-compacted --data-dir=<new_dir>
   ```

   Здесь:

   - `<etcd-date>` — название директории с резервной копией etcd, например, `etcd-20260303_140322`.
   - `<n_bumps>` — 64-битное целое число, на которое следует увеличить `REVISION`. Его можно

оценить, умножив среднее число операций записи в etcd в секунду на число секунд с момента создания резервной копии. Можно установить значение, большее оценки, меньшее — не рекомендуется.

Например, если в etcd производится 100 операций записи в секунду, а резервная копия создана 24 часа назад, минимально допустимое значение `n_bumps` — 24 * 60 * 60 * 100 = 8 640 000.

- `<new_dir>` — новый каталог данных etcd, например, `/var/lib/etcd_restored/`.



Опция `--mark-compacted` отмечает предыдущие ревизии etcd как сжатые. Это позволяет удостовериться, что все процессы отслеживания etcd остановлены, и кэши информеров сделаны недействительными.

4. Остановите сервис etcd, выполнив следующую команду:

```
systemctl stop etcd.service
```

5. В файле `/etc/etcd.env` измените значение переменной `ETCD_DATA_DIR` на путь, указанный в качестве значения `<new_dir>` на шаге 1.
6. Запустите сервис etcd, выполнив следующую команду:

```
systemctl start etcd.service
```

7. Убедитесь, что сервис etcd корректно запустился, с помощью команды:

```
systemctl status etcd
```

8. При необходимости удалите каталог, в котором хранились данные etcd до восстановления, с помощью следующей команды:

```
rm -rf <old_dir>
```

Здесь:

- `<old_dir>` — директория, в которой хранились данные etcd до восстановления, например, `/var/lib/etcd/`.

Резервное копирование без Backup Wrapper

В данном разделе описано, как настроить резервное копирование без сервиса Backup Wrapper с помощью вспомогательных утилит:

- **wal-g** для PostgreSQL;

- **clickhouse-backup** для ClickHouse.

Данный способ подходит в следующих случаях:

- Используется низконагруженная инсталляция с одной базой данных.
- Нет необходимости в HTTP API и мониторинге состояния работы по автоматическому резервному копированию.



В случае высоконагруженных инсталляций рекомендуется использовать сервис Backup Wrapper для автоматического создания резервных копий.

Содержание раздела:

- Резервное копирование PostgreSQL без Backup Wrapper
- Резервное копирование ClickHouse без Backup Wrapper

Резервное копирование PostgreSQL без Backup Wrapper

Система использует СУБД PostgreSQL для хранения конфигураций сервисов. Резервная копия базы данных системы может понадобиться для ее восстановления после сбоя или переноса на другое оборудование.

В данном разделе описано, как производить резервное копирование СУБД PostgreSQL с помощью утилиты wal-g.



Дополнительно с возможностями утилиты wal-g вы можете ознакомиться в официальной документации.



Все действия необходимо производить на физическом сервере или виртуальной машине, на которых развернута база данных PostgreSQL.

Требования

- Актуальная версия базы данных PostgreSQL.
- S3-совместимое хранилище для хранения резервных копий, например: MinIO, AWS S3, Yandex Object Storage.

Установка и настройка утилиты wal-g

Установка и настройка утилиты состоит из следующих этапов:

1. Установка утилиты wal-g
2. Настройка подключения к S3-совместимому хранилищу
3. Настройка WAL-архивирования

Установка утилиты wal-g

Чтобы установить утилиту wal-g:

1. Скачайте и разархивируйте дистрибутив программы Backup Wrapper, выполнив команды:

```
wget https://files.rvision.ru/platform/6dfaa56c92a665a855a9/backup-wrapper-
1.0.0.tar.gz
tar -xzvf backup-wrapper-1.0.0.tar.gz
cd backup-wrapper-1.0.0
```



В архиве программы Backup Wrapper содержится бинарный файл актуальной версии утилиты wal-g. Рекомендуется использовать данный вариант установки.

2. Установите утилиту wal-g, выполнив команды:

```
mv wal-g /usr/local/bin/
chmod +x /usr/local/bin/wal-g
```



Также вы можете установить утилиту wal-g из официального источника, выполнив команды:

```
curl -L https://github.com/wal-g/wal-
g/releases/download/<walg_version>/wal-g-pg-<os_name>-amd64.tar.gz |
tar xz
mv wal-g-pg-<os_name>-amd64 /usr/local/bin/wal-g
chmod +x /usr/local/bin/wal-g
```

Здесь:

- `<walg_version>` — версия утилиты wal-g, например, `v3.0.3`.
- `<os_name>` — название и версия используемой операционной системы, например, `ubuntu-20.04`.

Настройка подключения к S3-совместимому хранилищу

Чтобы настроить подключение к S3-совместимому хранилищу:

1. Создайте файл конфигурации `.walg.json` от имени пользователя postgres, выполнив команду:

```
su - postgres -c 'cat <<EOF >/<path_to_postgresql>/.walg.json
{
  "PGDATA": "<pgdata>",
  "PGHOST": "<pghost>",
  "AWS_ACCESS_KEY_ID": "<access_key>",
  "AWS_SECRET_ACCESS_KEY": "<secret_key>",
  "AWS_ENDPOINT": "<aws_endpoint>",
  "AWS_S3_FORCE_PATH_STYLE": "true",
  "AWS_REGION": "<aws_region>",
  "WALG_S3_PREFIX": "<walg_s3_prefix>",
```

```
  "WALG_COMPRESSION_METHOD": "lz4",
  "WALG_DELTA_MAX_STEPS": "7"
}
EOF'
```

Здесь:

- `<path_to_postgresql>` — путь до корневой директории PostgreSQL, например, `/var/lib/pgsql/`.
- `<pgdata>` — путь до директории с данными PostgreSQL (PGDATA), например, `/var/lib/pgsql/16/data/`.
- `<pghost>` — имя хоста сервера базы данных для подключения. Например, `/var/run/postgresql`, если подключение происходит через UNIX-сокет.
- `<access_key>` — идентификатор ключа доступа к S3-совместимому хранилищу.
- `<secret_key>` — секретный ключ доступа к S3-совместимому хранилищу.
- `<aws_endpoint>` — URL-адрес для подклчения к S3-совместимому хранилищу, например, `http://minio:9000`.
- `<aws_region>` — код региона, в котором развернуты и выполняются ваши сервисы и функции AWS, например, `ru-central1-a`.
- `<walg_s3_prefix>` — путь до хранилища, в котором будут храниться резервные копии, например, `s3://backups/postgresql/wal`.

2. Установите права доступа к файлу `.walg.json`, выполнив команды:

```
chown postgres:postgres /<path_to_postgresql>/.walg.json
chmod 600 /<path_to_postgresql>/.walg.json
```

Здесь:

- `<path_to_postgresql>` — путь до корневой директории PostgreSQL, например, `/var/lib/pgsql/`.

Настройка WAL-архивирования

Чтобы настроить WAL-архивирование:

1. Откройте файл `postgresql.conf`.



Вы можете узнать расположение файла `postgresql.conf`, выполнив SQL-запрос:

```
SHOW config_file
```

2. Добавьте в файл следующие строки:

```
wal_level = replica
archive_mode = on
archive_command = '/usr/local/bin/wal-g wal-push %p'
```

```
archive_timeout = 300
```

3. Выполните команду перезапуска PostgreSQL:

```
sudo systemctl restart <postgresql>
```

Здесь:

- `<postgresql>` — название сервиса postgresql, например `postgresql` или `postgresql-16`.



Чтобы точно узнать название сервиса, выполните следующую команду:

```
systemctl list-units | grep postgres
```

Резервное копирование

Чтобы создать резервную копию таблиц PostgreSQL, выполните следующие действия:

1. Создайте файл скрипта `pg-backup.sh` в директории `/usr/local/bin/` со следующим содержимым:

```
#!/bin/bash
set -euo pipefail

LOG="/var/log/pg-backup.log"
exec >> "$LOG" 2>&1

echo "=== $(date '+%Y-%m-%d %H:%M:%S') Starting backup ==="

DAY_OF_WEEK=$(date +%u)

if [ "$DAY_OF_WEEK" -eq 1 ]; then
    echo "Creating FULL backup (Monday)"
    sudo -u postgres wal-g backup-push --permanent
else
    LAST_FULL=$(sudo -u postgres wal-g backup-list --json 2>/dev/null | \
        jq -r '[.[] | select(.backup_name | test("^base_[^_]+$"))] | sort_by(.time)
| last | .backup_name // empty')

    if [ -z "$LAST_FULL" ]; then
        echo "No FULL backup found, creating FULL"
        sudo -u postgres wal-g backup-push --permanent
    else
        echo "Creating DELTA backup from $LAST_FULL"
        sudo -u postgres wal-g backup-push --delta-from-name="$LAST_FULL"
    fi
fi

echo "=== $(date '+%Y-%m-%d %H:%M:%S') Backup completed ==="
```

```
echo "Cleaning old backups..."
sudo -u postgres wal-g delete retain FULL 4 --confirm
```

2. Выдайте права доступа для созданного файла `pg-backup.sh`, выполнив команду:

```
chmod +x /usr/local/bin/pg-backup.sh
```

3. Настройте планировщик Cron для автоматизации резервного копирования, выполнив команду:

```
echo '<cron_expression> root /usr/local/bin/pg-backup.sh' | sudo tee
/etc/cron.d/pg-backup
```

Здесь:

- `<cron_expression>` — cron-выражение, определяющее расписание запусков wal-g. Например, `0 2 * * *` — резервные копии будут создаваться ежедневно в 02:00.



Чтобы составить cron-выражение, можно воспользоваться генератором, например, crontab.guru.



Вы можете проверить список созданных резервных копий, выполнив команду:

```
sudo -u postgres wal-g backup-list
```

Восстановление из резервной копии

Восстановление из резервной копии описано в разделе Восстановление из резервных копий.

Резервное копирование ClickHouse без Backup Wrapper

В данном разделе описано, как производить резервное копирование СУБД ClickHouse с помощью утилиты clickhouse-backup.



Дополнительно с возможностями утилиты clickhouse-backup вы можете ознакомиться в официальной документации.



Все действия необходимо производить на физическом сервере или виртуальной машине, на которых развернута база данных PostgreSQL.

Требования

- Актуальная версия базы данных ClickHouse.
- S3-совместимое хранилище для хранения резервных копий, например: MinIO, AWS S3, Yandex Object Storage.

Установка и настройка утилиты clickhouse-backup

Установка утилиты состоит из следующих этапов:

1. Установка утилиты clickhouse-backup
2. Настройка конфигурации clickhouse-backup

Установка утилиты clickhouse-backup

Чтобы установить утилиту clickhouse-backup:

1. Скачайте и разархивируйте дистрибутив программы Backup Wrapper, выполнив команды:

```
wget https://files.rvision.ru/platform/6dfaa56c92a665a855a9/backup-wrapper-
1.0.0.tar.gz
tar -xzvf backup-wrapper-1.0.0.tar.gz
cd backup-wrapper-1.0.0
```



В архиве программы Backup Wrapper содержится бинарный файл актуальной версии утилиты clickhouse-backup. Рекомендуется использовать данный вариант установки.

2. Установите утилиту clickhouse-backup, выполнив команды:

```
mv clickhouse-backup /usr/local/bin/
chmod +x /usr/local/bin/clickhouse-backup
```



Также вы можете установить утилиту clickhouse-backup из официального источника, выполнив команды:

```
curl -L https://github.com/Altinity/clickhouse-
backup/releases/download/<version>/clickhouse-backup-linux-amd64.tar.gz
| tar xz
mv clickhouse-backup /usr/local/bin/
chmod +x /usr/local/bin/clickhouse-backup
```

Здесь:

- `<version>` — версия утилиты clickhouse-backup, например, `v2.6.5`.

Настройка конфигурации clickhouse-backup

Чтобы настроить конфигурацию clickhouse-backup:

1. Создайте файл `config.yml` в директории `/etc/clickhouse-backup/` со следующим содержимым:

```
general:
```

```
  disable_progress_bar: true
  backups_to_keep_local: 0
  backups_to_keep_remote: 0
  remote_storage: s3

clickhouse:
  host: localhost
  port: 9000
  username: <username>
  password: "<password>"
  skip_tables:
    - system.*
    - INFORMATION_SCHEMA.*

s3:
  access_key: "<access_key>"
  secret_key: "<secret_key>"
  bucket: "backups"
  endpoint: "<aws_endpoint>"
  region: "<aws_region>"
  path: "clickhouse/prod"
  force_path_style: true
  compression_format: tar
  compression_level: 1
```

Здесь:

- `<username>` — имя пользователя для работы с резервными копиями.
- `<password>` — пароль пользователя для работы с резервными копиями.
- `<access_key>` — идентификатор ключа доступа к S3-совместимому хранилищу.
- `<secret_key>` — секретный ключ доступа к S3-совместимому хранилищу.
- `<aws_endpoint>` — URL-адрес для подключения к S3-совместимому хранилищу, например, `http://minio:9000`.
- `<aws_region>` — код региона, в котором развернуты и выполняются ваши сервисы и функции AWS, например, `ru-central1-a`.

2. Измените права доступа к файлу `config.yml`, выполнив команду:

```
chmod 600 /etc/clickhouse-backup/config.yml
```

Резервное копирование ClickHouse

Чтобы настроить резервное копирование:

1. Создайте файл `ch-backup.sh` в директории `usr/local/bin/` со следующим содержимым:

```
#!/bin/bash
```

```
set -euo pipefail

LOG="/var/log/ch-backup.log"
exec >> "$LOG" 2>&1

echo "=== $(date '+%Y-%m-%d %H:%M:%S') Starting backup ==="

BACKUP_NAME="backup-$(date +%Y-%m-%dT%H-%M-%S)"
DAY_OF_WEEK=$(date +%u)  # 1=Mon, 7=Sun

# Определяем тип бэкапа.
if [ "$DAY_OF_WEEK" -eq 1 ]; then
    echo "Creating FULL backup: $BACKUP_NAME"
    clickhouse-backup create_remote "$BACKUP_NAME"
else
    # Находим последнюю полную резервную копию (без --diff-from-remote в metadata).
    LAST_FULL=$(clickhouse-backup list remote --json 2>/dev/null | \
        jq -r '[.[] | select(.required == "")] | sort_by(.creation_date) | last |
.name // empty')

    if [ -z "$LAST_FULL" ]; then
        echo "No FULL backup found, creating FULL: $BACKUP_NAME"
        clickhouse-backup create_remote "$BACKUP_NAME"
    else
        echo "Creating DIFF backup from $LAST_FULL: $BACKUP_NAME"
        clickhouse-backup create_remote "$BACKUP_NAME" --diff-from
-remote="$LAST_FULL"
    fi
fi

echo "=== $(date '+%Y-%m-%d %H:%M:%S') Backup completed ==="
```

2. Измените права доступа к файлу `ch-backup.sh`, выполнив команду:

```
chmod +x /usr/local/bin/ch-backup.sh
```

3. Настройте планировщик Cron для автоматизации резервного копирования, выполнив команду:

```
echo '<cron_expression> root /usr/local/bin/ch-backup.sh' | sudo tee
/etc/cron.d/ch-backup
```

Здесь:

- `<cron_expression>` — cron-выражение, определяющее расписание запусков clickhouse-backup. Например, `0 3 * * *` — резервные копии будут создаваться ежедневно в 03:00.



Чтобы составить cron-выражение, можно воспользоваться генератором, например, crontab.guru.

Вы можете проверить список созданных резервных копий, выполнив команду:

```
clickhouse-backup list remote
```

Восстановление из резервной копии

Восстановление из резервной копии описано в разделе Восстановление из резервных копий.

Резервное копирование сервисов платформы

Вы можете настроить автоматическое резервное копирование всех ключевых компонентов системы: NATS, MinIO, Dkron.

Требования

- Доступ к S3-совместимому хранилищу с резервными копиями, например: MinIO, AWS S3, Yandex Object Storage.
- Доступ к утилитам kubectl и evoctl.

Настройка автоматического резервного копирования

Чтобы настроить резервное копирование сервисов:

1. Подключитесь к кластеру.
2. Подключитесь к master-узлу или машине, с которой производилась установка системы.
3. Перейдите в директорию установщика системы. По умолчанию `/opt/r-vision`.
4. Если у вас не загружен образ сервиса Backup Wrapper для работы с резервными копиями сервисов, то загрузите его с помощью команды:

   ```
   evoctl instance upload --archive backup-tools.img
   ```

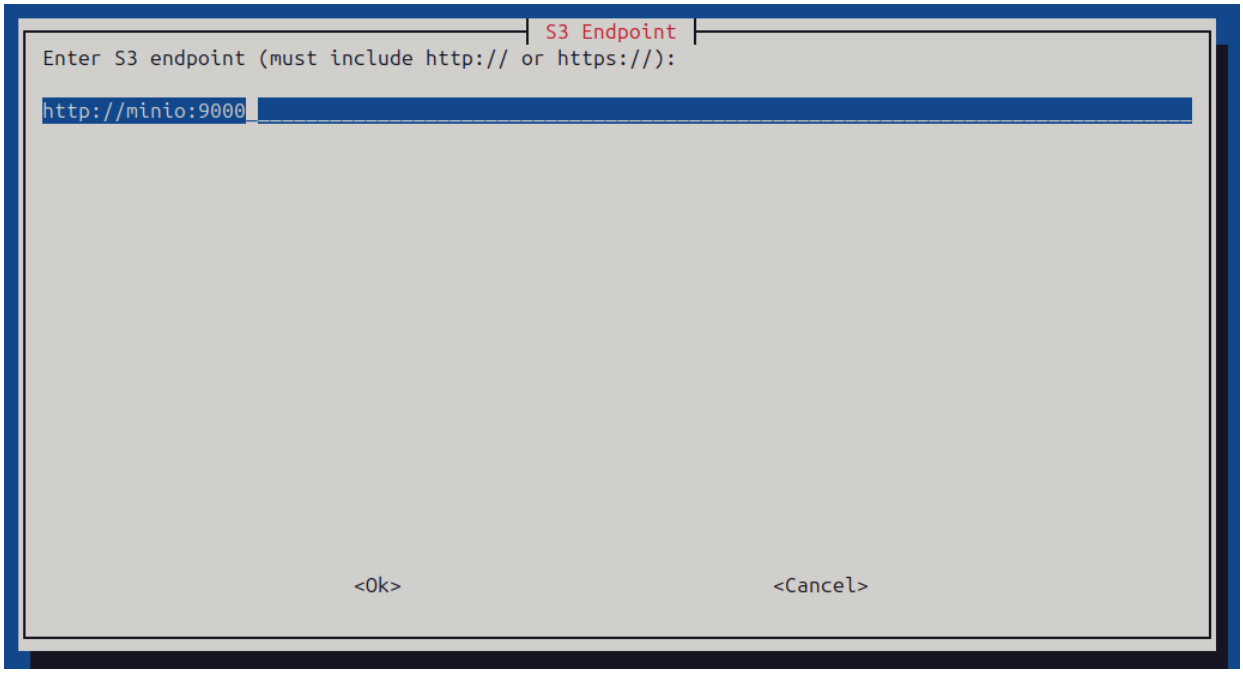
5. Задайте переменную `PATH` с путем к мастеру интерактивной настройки, выполнив команду:

   ```
   export PATH=$PATH:/opt/r-vision/common/bin
   ```

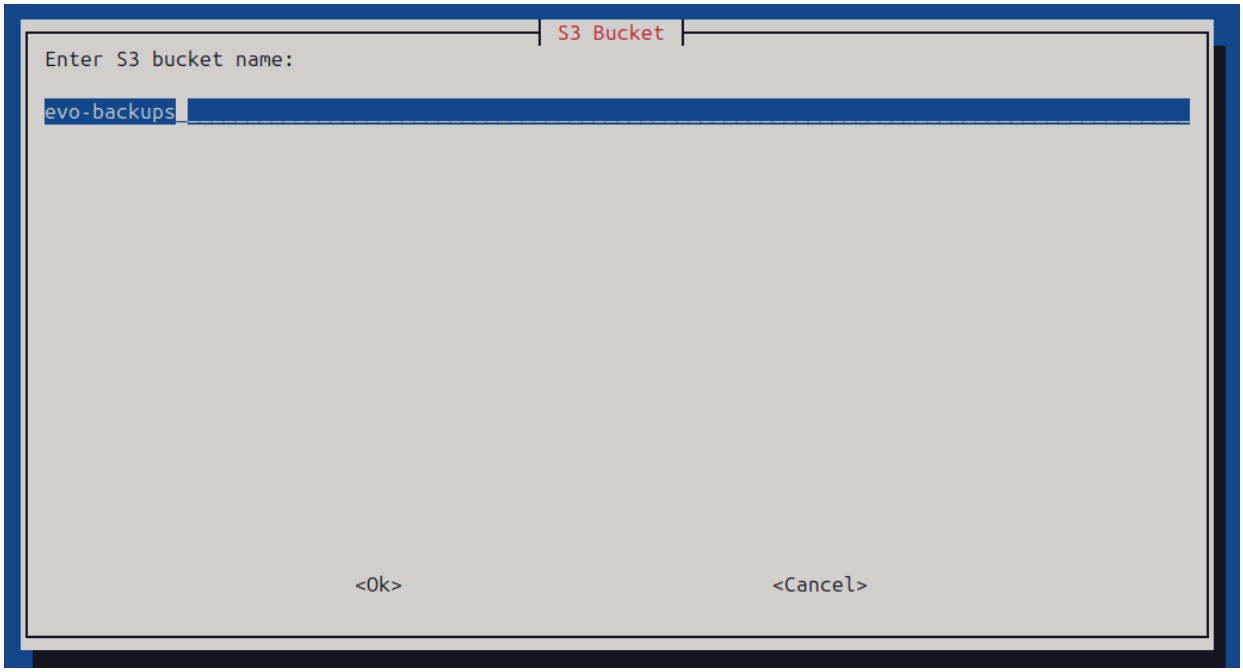
6. Запустите мастер интерактивной настройки, выполнив команду:

   ```
   evoctl instance backup enable
   ```

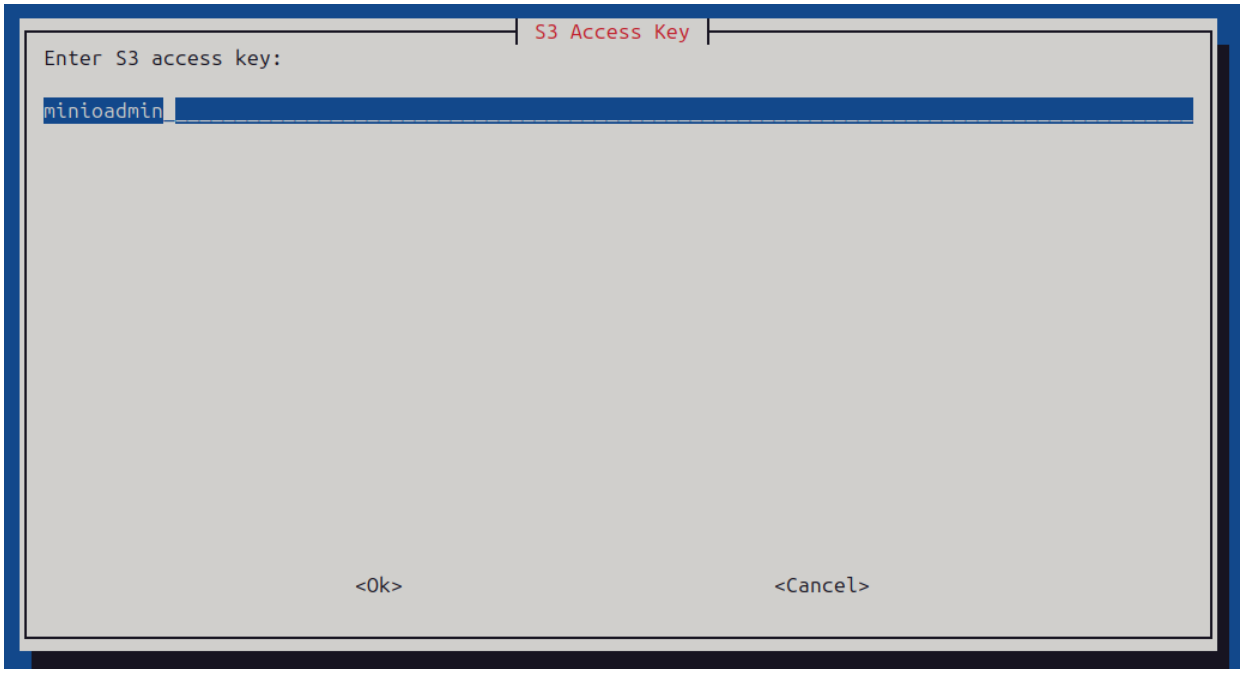
7. В окне мастера интерактивной настройки выберите сервисы, для которых необходимо настроить резервное копирование.
8. Заполните данные для подключения к S3-совместимому хранилищу. Для этого:
   a. Введите URL-адрес для подключения к S3-совместимому хранилищу.



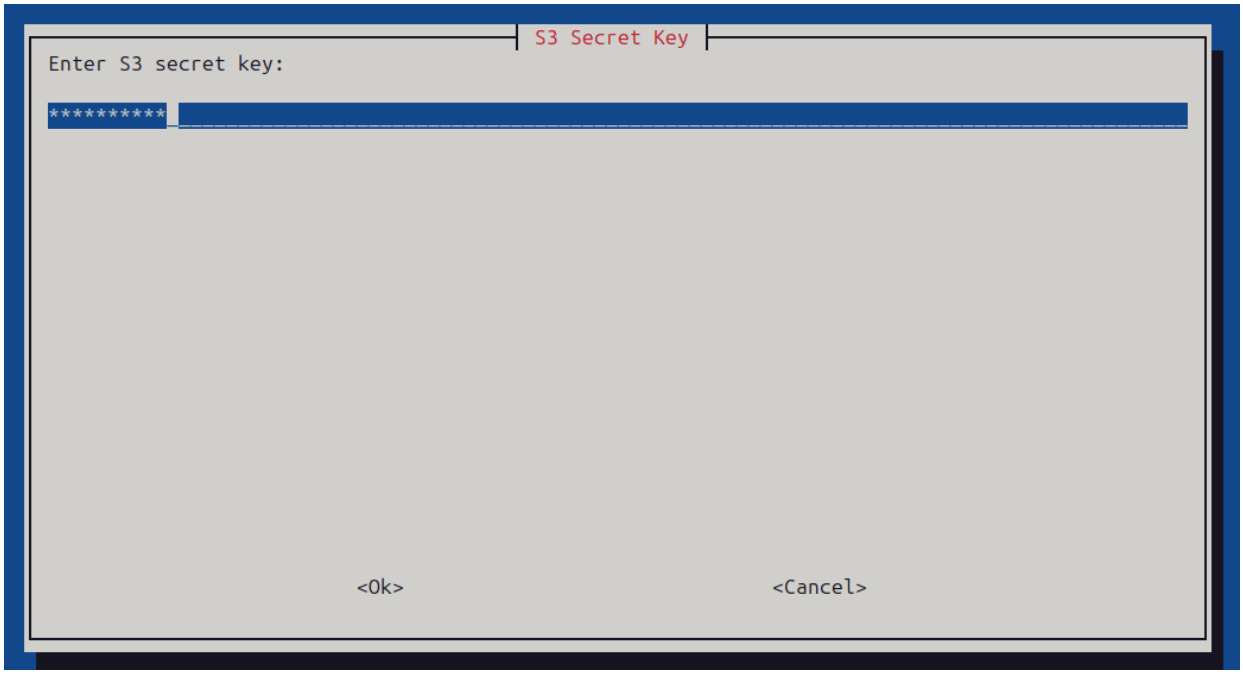
b. Введите имя корзины (bucket).



c. Введите access key для подключения к S3-совместимому хранилищу.



d. Введите secret key для подключения к S3-совместимому хранилищу.



e. Введите secret key повторно для его подтверждения. После ввода данных установщик проверит соединение с S3-совместимым хранилищем.

9. Если вы хотите запустить резервное копирование сразу, то создайте задачи (job), выполнив команды:

```
kubectl create job --from=cronjob/backup-minio manual-minio-$(date +%s) -n
<namespace>
kubectl create job --from=cronjob/backup-dkron manual-dkron-$(date +%s) -n
<namespace>
kubectl create job --from=cronjob/backup-nats manual-nats-$(date +%s) -n
<namespace>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

10. Проверьте логи, что задачи (job) создались корректно, с помощью команд:

```
kubectl logs job/manual-minio-<timestamp> -n <namespace>
kubectl logs job/manual-nats-<timestamp> -n <namespace>
kubectl logs job/manual-dkron-<timestamp> -n <namespace>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<timestamp>` — дата и время, когда была создана задача (job).

Работа с резервными копиями сервисов

Доступные операции:

- Просмотр статуса резервного копирования
- Отключение резервного копирования



Все команды утилиты evoctl необходимо выполнять на master-узле или машине, с которой производилась установка системы.

Просмотр статуса резервного копирования

Чтобы посмотреть статус резервного копирования, выполните команду:

```
evoctl instance backup disable -n <namespace>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Отключение резервного копирования

Чтобы отключить резервное копирование сервисов, выполните одну из команд:

- Для отключения MinIO:

```
evoctl instance backup disable -n <namespace> --minio-backup
```

- Для отключения Dkron:

```
evoctl instance backup disable -n <namespace> --dkron-backup
```

- Для отключения NATS:

```
evoctl instance backup disable -n <namespace> --nats-backup
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



При отключении резервного копирования сервисов секреты с данными подключения к S3-совместимому хранилищу не удаляются.

Чтобы снова включить резервное копирование, выполните команду:

```
evoctl instance backup enable -n <namespace>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Резервное копирование внутренних баз данных

В данном разделе описана настройка резервного копирования баз данных PostgreSQL и ClickHouse, развернутых в кластере Kubernetes. Данный вариант резервного копирования подходит, если база данных:

- развернута через стандартную установку системы;
- управляется helm-чартами в кластере Kubernetes, в котором установлена система;
- является частью инфраструктуры системы.

Требования

- S3-совместимое хранилище для хранения резервных копий, например: MinIO, AWS S3, Yandex Object Storage.
- Доступ к утилитам kubectl, helm и evoctl.

Настройка резервного копирования

Вы можете настроить резервное копирование следующими способами:

- Интерактивная настройка
- Настройка с помощью командной строки

Интерактивная настройка

1. Подключитесь к кластеру, в котором развернута база данных.
2. Подключитесь к master-узлу или виртуальной машине, с которой производилась установка системы.
3. Перейдите в директорию установщика системы. По умолчанию `/opt/r-vision`.
4. Запустите процес настройки резервного копирования, выполнив команду:

   ```
   evoctl instance backup enable -n <namespace>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

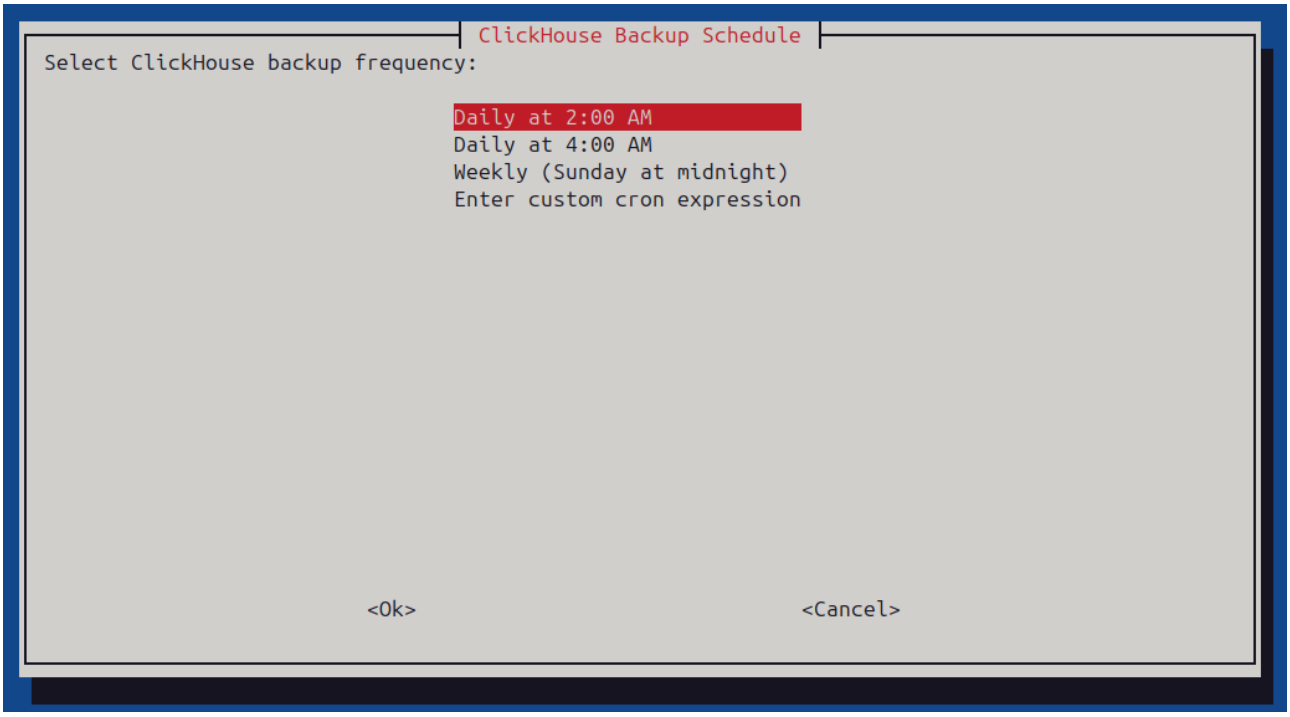
   > Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

   Откроется мастер интерактивной настройки резервного копирования.

5. Выберите тип базы данных.

   > Перемещение по списку опций осуществляется с помощью клавиш со стрелками вверх и вниз. Чтобы выбрать опцию или отменить ее выбор, нажмите на клавишу SPACE.

6. Выберите расписание резервного копирования.



   - **Daily at 2:00 AM** — ежедневно в 02:00. Рекомендуемый вариант.
   - **Daily at 4:00 AM** — ежедневно в 04:00.

- **Weekly (Sunday at midnight)** — еженедельно, в воскресенье в 00:00.
- **Enter custom cron expression** — свое расписание в cron-формате.

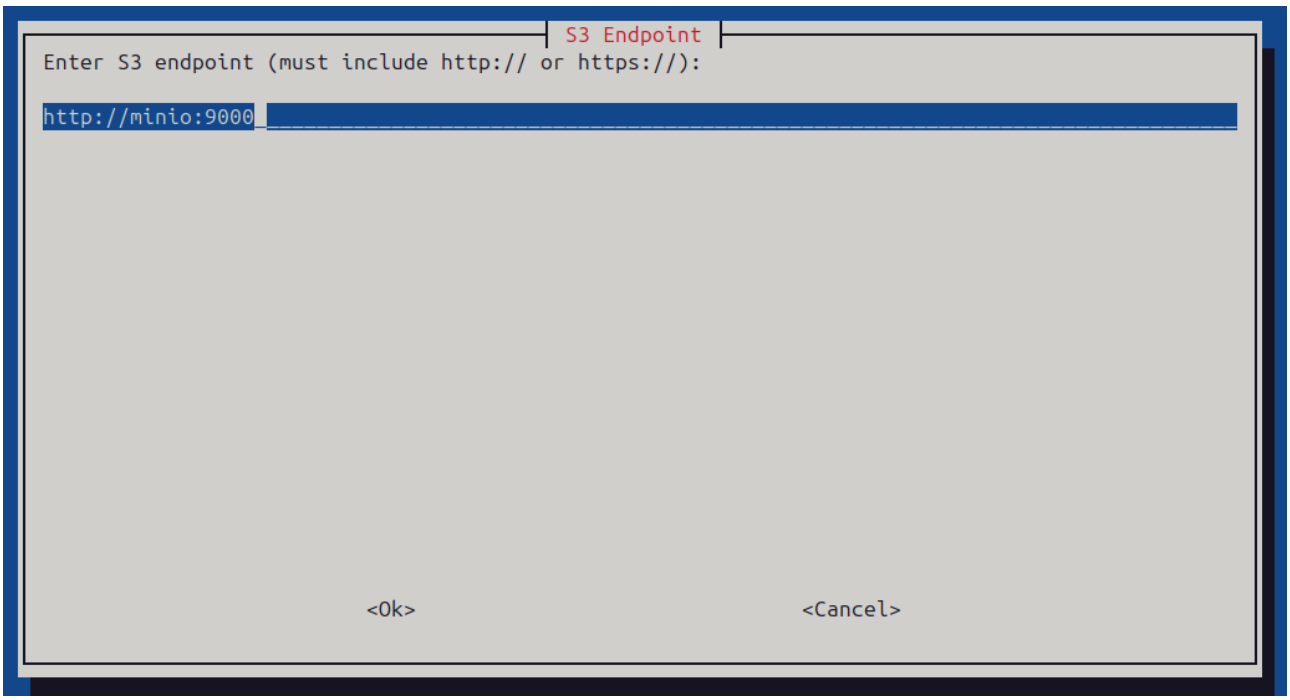


Не рекомендуется настраивать резервное копирование чаще одного раза в сутки.

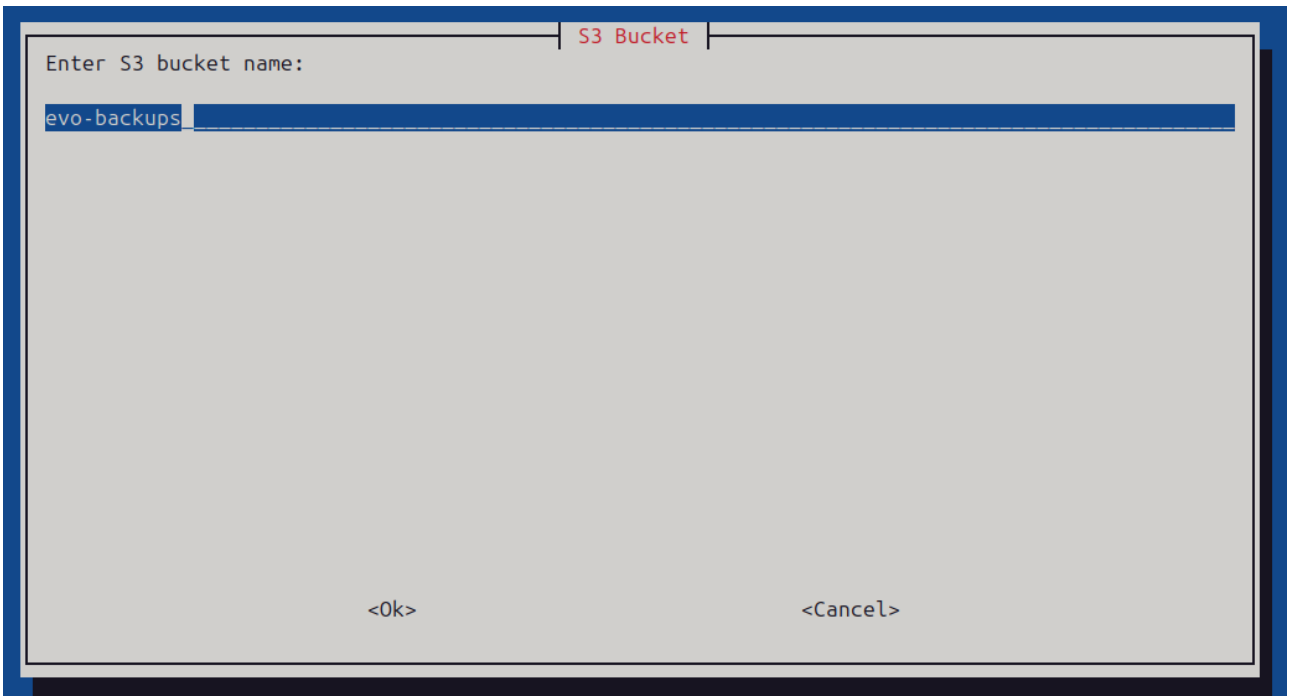


Время в расписании указывается относительно часового пояса в кластере Kubernetes, в котором установлена платформа.

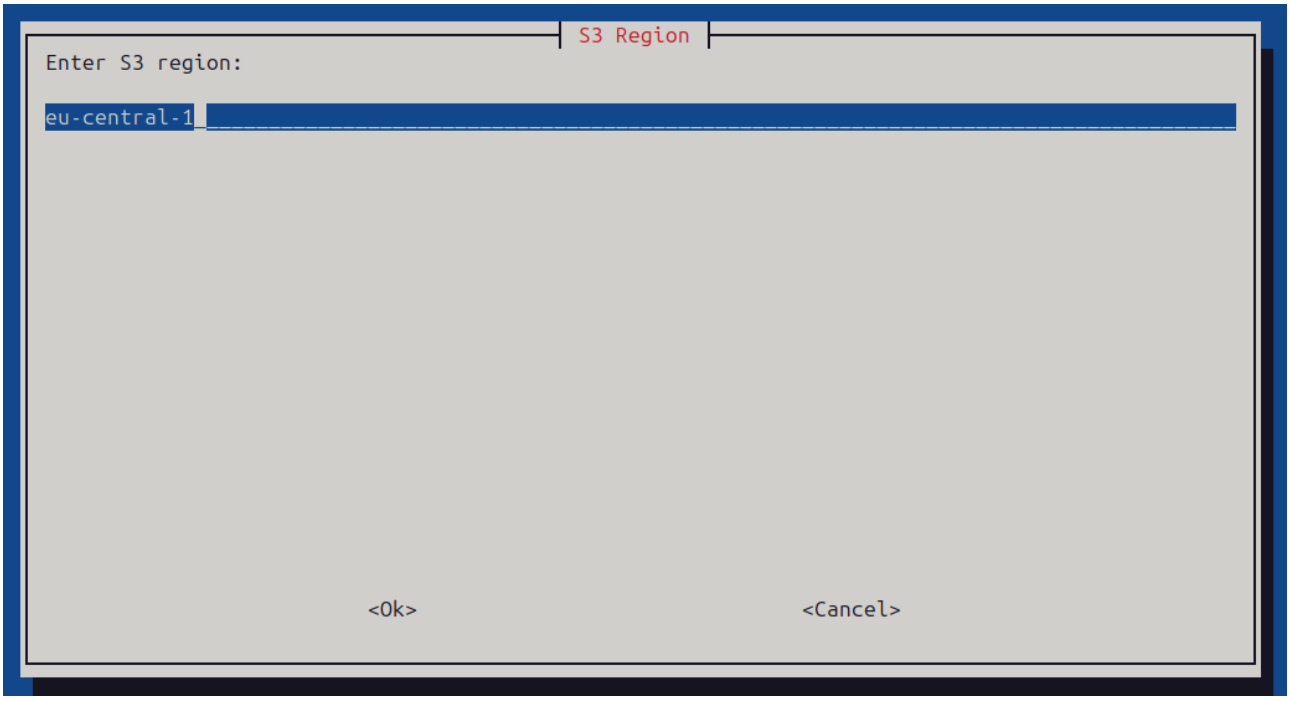
7. Введите URL-адрес для подключения к S3-совместимому хранилищу.



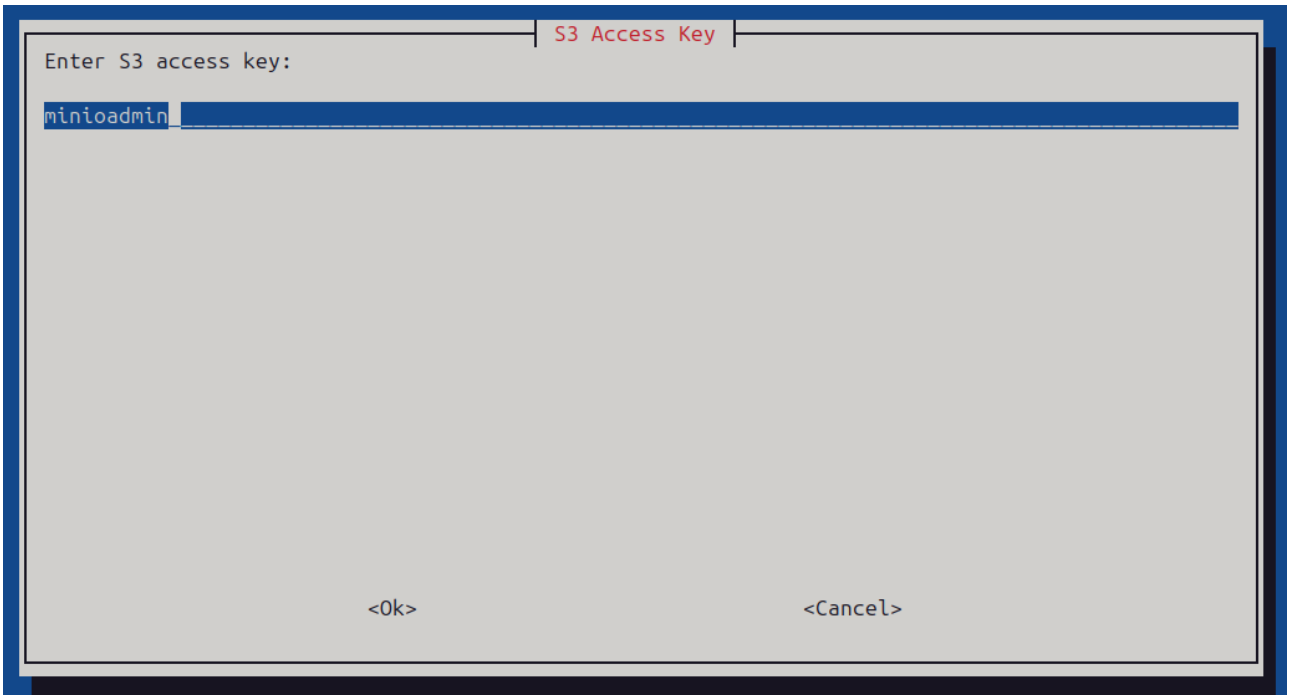
8. Введите имя корзины (bucket).



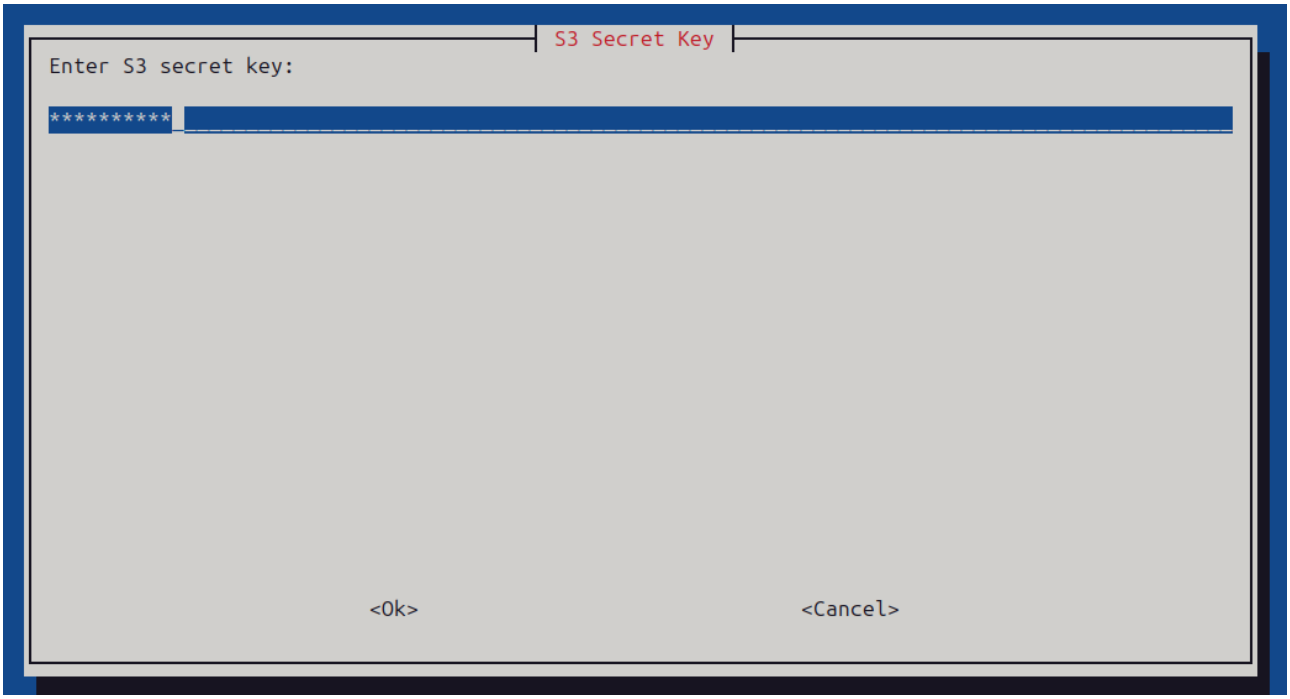
9. Введите регион S3-совместимого хранилища.



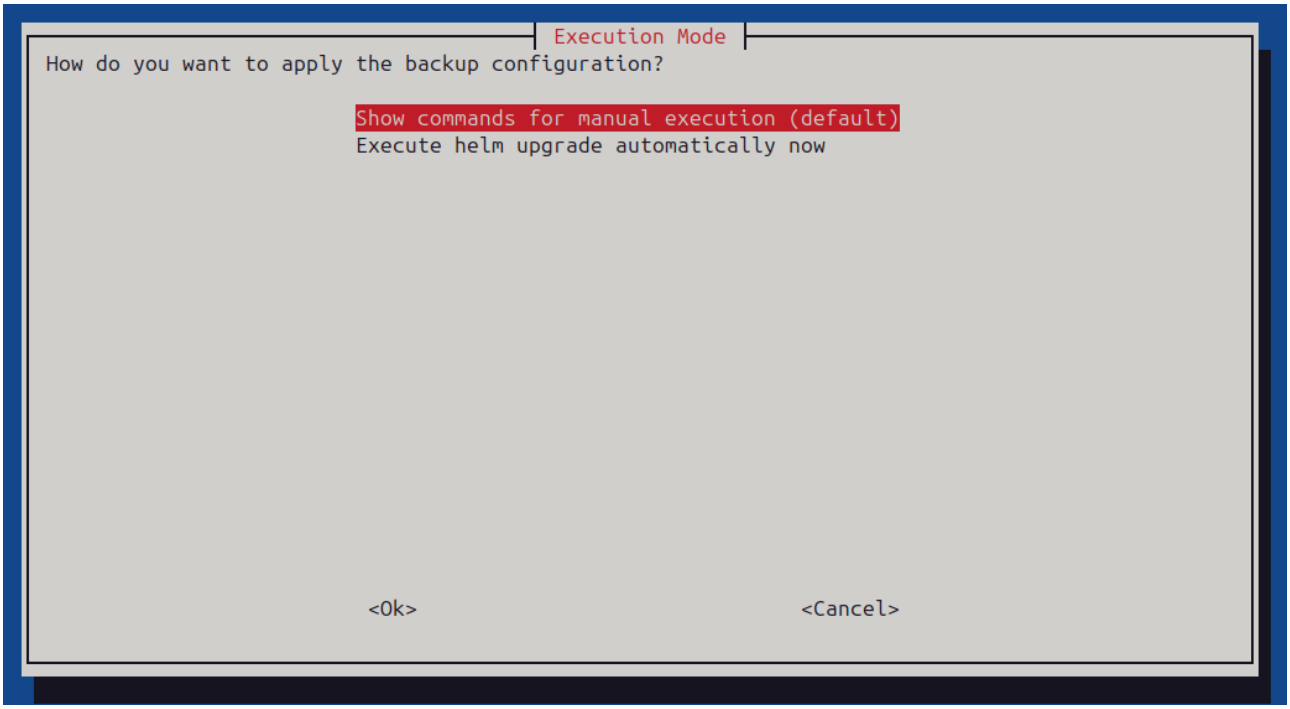
10. Введите access key для подключения к S3-совместимому хранилищу.



11. Введите secret key для подключения к S3-совместимому хранилищу.



12. Введите secret key повторно для его подтверждения. После ввода данных установщик проверит соединение с S3-совместимым хранилищем.
13. Выберите способ применения конфигурации резервного копирования.



- **Show commands for manual execution (default)** — показать команды для применения конфигурации вручную. Рекомендуемый вариант.
- **Execute helm upgrade automatically now** — применить конфигурацию автоматически.

14. Если вы выбрали вариант применения кофигурации **Show commands for manual execution**, то примените конфигурацию вручную с помощью команды helm.

Настройка с помощью командной строки

1. Подключитесь к кластеру, на котором развернута база данных.
2. Выполните команду настройки конфигурации в зависимости от сценария использования:
   - Только ClickHouse:

```
evoctl instance backup enable \
  --namespace <namespace> \
  --ch-backup \
  --s3-endpoint <endpoint> \
  --s3-bucket <bucket> \
  --s3-access-key <access_key> \
  --s3-secret-key <secret_key>
```

   - Только PostgreSQL:

```
evoctl instance backup enable \
  --namespace <namespace> \
  --pg-backup \
  --s3-endpoint <endpoint> \
  --s3-bucket <bucket> \
  --s3-access-key <access_key> \
  --s3-secret-key <secret_key>
```

   - Обе базы данных через один S3-bucket:

```
evoctl instance backup enable \
  --namespace <namespace> \
  --both \
  --s3-endpoint <endpoint> \
  --s3-bucket <bucket> \
  --s3-region <region> \
  --s3-access-key <access_key> \
  --s3-secret-key <secret_key> \
  --auto-upgrade
```

   - Обе базы данных через разные S3-bucket:

```
evoctl instance backup enable \
  --namespace <namespace> \
  --both \
  --s3-endpoint <endpoint> \
  --s3-access-key <access_key> \
  --s3-secret-key <secret_key> \
  --ch-bucket <ch_bucket> \
```

```
  --pg-bucket <pg_bucket>
```

- Обе базы данных через разные S3-совместимые хранилища:
  - Настройка ClickHouse:

```
evoctl instance backup enable \
  --namespace <namespace> \
  --ch-backup \
  --s3-endpoint <endpoint> \
  --s3-bucket <ch_bucket> \
  --s3-access-key <access_key> \
  --s3-secret-key <secret_key>
```

  - Настройка PosgreSQL:

```
evoctl instance backup enable \
  --namespace <namespace> \
  --pg-backup \
  --s3-endpoint <endpoint> \
  --s3-bucket <pg_bucket> \
  --s3-access-key <access_key> \
  --s3-secret-key <secret_key>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<endpoint>` — URL-адрес S3-совместимого хранилища, например, `http://minio:9000`
- `<region>` — код региона, в котором развернуты и выполняются ваши сервисы и функци AWS, например, `ru-central1-a`.
- `<access_key>` — идентификатор ключа доступа к S3-совместимому хранилищу.
- `<secret_key>` — секретный ключ доступа к S3-совместимому хранилищу.
- `<bucket>` — название контейнера (bucket) для хранения резервных копий.
- `<ch_bucket>` — название контейнера (bucket) для хранения резервных копий ClickHouse.
- `<pg_bucket>` — название контейнера (bucket) для хранения резервных копий PostgreSQL.

Работа с резервным копированием

Доступные операции:

- Изменение расписания
- Изменение данных для подключения к S3-совместимому хранилищу
- Проверка статуса резервного копирования
- Отключение резервного копирования
- Ручной запуск резервного копирования



Все действия необходимо производить в кластере Kubernetes, в котором развернута база данных. Подробности о подключении к кластеру приведены в разделе Подключение к кластеру.

Изменение расписания



Не рекомендуется настраивать резервное копирование чаще одного раза в сутки.

Чтобы изменить расписание резервного копирования, выполните команду смены расписания в зависимости от типа базы данных:

- ClickHouse:

```
evoctl instance backup configure \
  --namespace <namespace> \
  --ch-only \
  --schedule "<schedule>"
```

- PostgreSQL:

```
evoctl instance backup configure \
  --namespace <namespace> \
  --pg-only \
  --schedule "<schedule>"
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<schedule>` — новое расписание в cron-формате. Например, `30 4 * * *` — ежедневно в 4:30.

Изменение данных для подключения к S3-совместимому хранилищу

Чтобы изменить access key и secret key, выполните команду для изменения данных для подключения к S3-совместимому хранилищу:

```
evoctl instance backup configure \
  --namespace <namespace> \
  --s3-access-key <new_access_key> \
  --s3-secret-key <new_secret_key>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<new_access_key>` — идентификатор ключа доступа к S3-совместимому хранилищу.
- `<new_secret_key>` — секретный ключ доступа к S3-совместимому хранилищу.



Вы можете изменить другие параметры подключения к S3-совместимому хранилищу, указав их в виде флагов в команде `evoctl instance backup configure`. Например:

```
evoctl instance backup configure \
  --namespace <namespace> \
  --s3-endpoint <endpoint>
  --s3-bucket <bucket>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<endpoint>` — URL-адрес S3-совместимого хранилища.
- `<bucket>` — имя корзины (bucket) для хранения резервных копий.

Проверка статуса резервного копирования

Чтобы отобразить логи резервного копирования, выполните команду:

```
evoctl instance backup info -n <namespace>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Отключение резервного копирования

Чтобы отключить резервное копирование, выполните команду в зависимости от сценария:

- Отключить резервное копирование полностью:

  ```
  evoctl instance backup disable -n <namespace>
  ```

- Отключить только резервное копирование ClickHouse:

  ```
  evoctl instance backup disable -n <namespace> --ch-only
  ```

- Отключить только резервное копирование PostgreSQL:

  ```
  evoctl instance backup disable -n <namespace> --pg-only
  ```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете открыть справку по команде отключения резервного копирования, выполнив команду:

```
evoctl instance backup disable --help
```

Ручной запуск резервного копирования

Чтобы запустить резервное копирование вручную:

1. Выполните резервное копирование вручную с помощью одной из команд:
   - Для ClickHouse:

     ```
     kubectl create job \
       --from=cronjob/clickhouse-backup-shard0 \
       manual-ch-$(date +%s) \
       -n <namespace>
     ```

   - Для PostgreSQL:

     ```
     kubectl create job \
       --from=cronjob/postgresql-backup-client \
       manual-pg-$(date +%s) \
       -n <namespace>
     ```

2. Проверьте статус задания (job) резервного копирования, выполнив команду:

   ```
   kubectl get jobs -n <namespace> | grep backup
   ```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Восстановление из резервной копии

Восстановление из резервной копии описано в разделе Восстановление из резервных копий.

Работа с Backup Wrapper

В данном разделе описан процесс установки и настройки сервиса Backup Wrapper. Сервис Backup Wrapper позволяет автоматически создавать резервные копии для баз данных PostgreSQL и CLickHouse с загрузкой их в S3-совместимое хранилище.

Для создания резервных копий Backup Wrapper использует следующие утилиты:

- wal-g — для PostgreSQL.
- clickhouse-backup — для ClickHouse.



В рамках раздела описаны необходимые шаги по настройке конфигурационных файлов утилит wal-g и clickhouse-backup. Дополнительно с возможносятми утилит wal-g вы можете ознакомиться в официальных документациях:

- wal-g.
- clickhouse-backup.

Возможности Backup Wrapper:

- Автоматически запускает утилиты резервного копирования.
- Сохраняет резервные копии в облачные S3-совместимые хранилища, например: MinIO, AWS S3, Yandex Object Storage.
- Отслеживает статус, поддерживает отмену и повторные попытки резервного копирования.



Сервис Backup Wrapper поддерживает только создание резервных копий. Для восстановления используйте напрямую утилиты wal-g и clickhouse-backup. Подробности приведены в разделе Восстановление из резервных копий.

Установка и настройка Backup Wrapper состоит из следующих этапов:

- Настройка PostgreSQL для работы с Backup Wrapper
- Настройка ClickHouse для работы с Backup Wrapper
- Настройка многоуровневого хранилища ClickHouse
- Установка и настройка Backup Wrapper

Настройка PostgreSQL для работы с Backup Wrapper

Backup Wrapper использует утилиту wal-g для создания резервных копий баз данных PostgreSQL. Для корректной работы сервиса предварительно необходимо настроить базу данных.



Утилита wal-g устанавливается вместе с Backup Wrapper. Подробности приведены в разделе Установка утилиты wrapper.

Настройка базы данных PostgreSQL состоит из следующих этапов:

1. Создание пользователя для работы с резервными копиями
2. Настройка файла конфигурации `postgresql.conf`
3. Настройка подключения к S3-совместимому хранилищу
4. Настройка файла `pg_hba.conf`
5. Перезапуск PostgreSQL и проверка настроек
6. Подготовка директории для хранения логов



Все действия необходимо производить на физическом сервере или виртуальной машине, на которых развернута база данных PostgreSQL.

Создание пользователя для работы с резервными копиями

Чтобы создать пользователя PostgreSQL:

1. Подключитесь к PostgreSQL как суперпользователь, выполнив команду:

   ```
   psql -U postgres
   ```

2. Создайте пользователя для работы с резервными копиями баз данных и выдайте ему необходимые права, выполнив SQL-запросы:

   ```
   CREATE USER backup_user WITH PASSWORD '<password>' REPLICATION;
   GRANT pg_read_all_data TO backup_user;
   ```

   Здесь:

   - `<password>` — пароль нового пользователя.

Настройка файла конфигурации `postgresql.conf`

Файл конфигурации `postgresql.conf` определяет основные параметры работы экземпляра базы данных.

Чтобы настроить файл конфигурации:

1. Откройте файл `postgresql.conf`.

   Вы можете узнать расположение файла `postgresql.conf`, выполнив SQL-запрос:

   ```
   SHOW config_file
   ```

2. Добавьте в файл следующие строки в зависимости от сценария работы:

   WAL-архивирование

   В данном сценарии происходит WAL-архивирование резервной копии базы данных. Это позволяет корректно производить восстановление данных из резервной копии. Данный сценарий подходит для итогового решения.

   ```
   wal_level = replica
   max_wal_senders = 3
   archive_mode = on
   archive_timeout = 300
   archive_command = '/usr/bin/wal-g wal-push %p >> /var/log/pgsql/archive.log
   2>&1'
   max_wal_size = 4GB
   ```

Без WAL-архивирования

В данном сценарии не происходит WAL-архивирование резервной копии базы данных. Из-за этого невозможно восстановить данные с помощью инструмента wal-g из резервной копии. Данный сценарий подходит для тестирования работы с резервными копиями.

```
wal_level = replica
max_wal_senders = 3
archive_mode = on
archive_command = '/bin/true'
max_wal_size = 4GB
```



В дальнейшем для смены сценария работы достаточно выполнить SQL-запрос:

```
ALTER SYSTEM SET archive_command = '<scenario_type>';
SELECT pg_reload_conf();
```

Здесь:

- `<scenario_type>` — тип работы wal-g:
  - `/usr/bin/wal-g wal-push %p` — с WAL-архивированием.
  - `/bin/true` — без WAL-архивирования, только для тестирования.

Настройка подключения к S3-совместимому хранилищу

Чтобы настроить подключение к S3-совместимому хранилищу:

1. Создайте файл конфигурации `.walg.json` от имени пользователя postgres, выполнив команду:

```
su - postgres -c 'cat <<EOF >/<path_to_postgresql>/.walg.json
{
  "PGDATA": "<pgdata>",
  "PGHOST": "<pghost>",
  "AWS_ACCESS_KEY_ID": "<access_key>",
  "AWS_SECRET_ACCESS_KEY": "<secret_key>",
  "AWS_REGION": "<aws_region>",
  "AWS_ENDPOINT": "<aws_endpoint>",
  "AWS_S3_FORCE_PATH_STYLE": "true",
  "WALG_S3_PREFIX": "<walg_s3_prefix>",
  "WALG_UPLOAD_CONCURRENCY": "2",
  "WALG_DOWNLOAD_CONCURRENCY": "4",
  "WALG_UPLOAD_DISK_CONCURRENCY": "2",
  "WALG_DELTA_MAX_STEPS": "7",
  "WALG_COMPRESSION_METHOD": "lz4"
}
```

```
EOF'
```

Здесь:

- `<path_to_postgresql>` — путь до корневой директории PostgreSQL, например, `/var/lib/pgsql/`.
- `<pgdata>` — путь до директории с данными PostgreSQL (PGDATA), например, `/var/lib/pgsql/16/data/`.
- `<pghost>` — имя хоста сервера базы данных для подключения. Например, `/var/run/postgresql`, если подключение происходит через UNIX-сокет.
- `<access_key>` — идентификатор ключа доступа к S3-совместимому хранилищу.
- `<secret_key>` — секретный ключ доступа к S3-совместимому хранилищу.
- `<aws_region>` — код региона, в котором развернуты и выполняются ваши сервисы и функции AWS, например, `ru-central1-a`.
- `<aws_endpoint>` — URL-адрес для подключения к S3-совместимому хранилищу, например, `http://minio:9000`.
- `<walg_s3_prefix>` — путь до хранилища, в котором будут храниться резервные копии, например, `s3://backups/postgresql/wal`.

2. Установите права доступа к файлу `.walg.json`, выполнив команды:

```
chmod 600 /<path_to_postgresql>/.walg.json
chown postgres:postgres /<path_to_postgresql>/.walg.json
```

Здесь:

- `<path_to_postgresql>` — путь до корневой директории PostgreSQL, например, `/var/lib/pgsql/`.

Настройка файла `pg_hba.conf`

Файл конфигурации `pg_hba.conf` позволяет управлять доступом к серверу баз данных.

Чтобы настроить файл `pg_hba.conf`:

1. Откройте для редактирования файл конфигурации `pg_hba.conf`.



Вы можете узнать расположение файла `pg_hba.conf`, выполнив SQL-запрос:

```
SHOW hba_file;
```

2. Вставьте в файл `pg_hba.conf` следующие строки:

```
local   replication     backup_user                             trust
host    replication     backup_user     127.0.0.1/32            md5
host    replication     backup_user     ::1/128                 md5
```

Перезапуск PostgreSQL и проверка настроек

Для применения всех внесенных изменений необходимо перезапустить PostgreSQL. Чтобы это сделать:

1. Перезагрузите конфигурацию менеджера системы, выполнив команду:

```
sudo systemctl daemon-reload
```

2. Выполните команду перезапуска PostgreSQL:

```
sudo systemctl restart <postgresql>
```

   Здесь:

   - `<postgresql>` — название сервиса postgresql, например `postgresql` или `postgresql-16`.



   Чтобы получить название сервиса, выполните следующую команду:

```
systemctl list-units | grep postgres
```

3. Проверьте, что настройки файла конфигурации `postgresql.conf` выставлены корректно, с помощью команды:

```
sudo -u postgres psql -c "SELECT name, setting FROM pg_settings WHERE name IN
('wal_level', 'max_wal_senders', 'archive_mode', 'archive_command');"
```

4. Проверьте, что WAL-архивирование работает корректно, с помощью команды:

```
sudo -u postgres psql -c "SELECT archived_count, failed_count FROM
pg_stat_archiver;"
```

   После выполнения команды подождите количество секунд, которое вы указали в параметре `archive_timeout` ранее.

Подготовка директории для хранения логов

Чтобы подготовить директорию для хранения логов:

1. Создайте директорию `pgsql`, выполнив команду:

```
mkdir -p /var/log/pgsql
```

2. Установите права доступа к директории `pgsql`, выполнив команды:

```
chown postgres:postgres /var/log/pgsql
chmod 750 /var/log/pgsql
```

Мониторинг объема директории с WAL-файлами

Для мониторинга объема директории с WAL-файлами доступны следующие команды:

- Вывод размера директории `pg_wal`:

  ```
  du -sh /<path_to_pgdata>/pg_wal/
  ```

  Здесь:

  - `<path_to_pgdata>` — путь до директории с данными PostgreSQL (PGDATA), например, `/var/lib/pgsql/16/data/`.

- Вывод количества WAL-файлов, ожидающих архивации:

  ```
  ls -la /<path_to_pgdata>/pg_wal/archive_status/*.ready 2>/dev/null | wc -l
  ```

  Здесь:

  - `<path_to_pgdata>` — путь до директории с данными PostgreSQL (PGDATA), например, `/var/lib/pgsql/16/data/`.

- Вывод статистики WAL-архиватора:

  ```
  sudo -u postgres psql -c "SELECT * FROM pg_stat_archiver;"
  ```

  ▼ *Интерпретация полей* `pg_stat_archiver`

  - `archived_count` — количество успешно архивированных WAL-файлов с момента последнего сбора статистики.
  - `failed_count` — количество неудачных попыток архивации.
  - `last_archived_wal` — имя последнего успешно архивированного WAL-файла.
  - `last_archived_time` — время завершения последней успешной архивации.
  - `last_failed_wal` — имя WAL-файла, при архивации которого произошла последняя ошибка.
  - `last_failed_time` — время последней ошибки.
  - `stats_reset` — время последнего сброса статистики.

  Количество неудачных попыток архивации (`failed_count`) больше нуля приемлемо при кратковременных сбоях S3-совместимого хранилища. PostgreSQL автоматически перезапускает процесс WAL-архивации.

  Сценарии, при которых необходимо обратить внимание:

- количество неудачных попыток архивации (`failed_count`) постоянно растет;
- с последней ошибки архивации (`last_failed_time`) прошло немного времени и количество успешно архивированных WAL-файлов (`archived_count`) не увеличивается.

Настройка ClickHouse для работы с Backup Wrapper

Backup Wrapper использует утилиту clickhouse-backup для работы с резервными копиями. Для корректной работы сервиса предварительно необходимо настроить базу данных.



Утилита clickhouse-backup устанавливается вместе с Backup Wrapper. Подробности приведены в разделе Установка утилиты wrapper.

Сервис Backup Wrapper использует подход с созданием резервных DIFF-копий (через параметр `--diff-from-remote`). Благодаря этому подходу сервис создает одну полную резервную копию базы данных, а все последующие копии содержат только изменения относительно корневой резервной копии.

Вы можете настроить процесс создания резервных DIFF-копий. Например, в первый день недели создавать полную копию базы данных, а в каждый последующий день создавать DIFF-копию. При этом каждая DIFF-копия создается от полной копии базы данных, а не от предыдущей DIFF-копии. С началом новой недели цикл повторяется.

Преимущества данного подхода:

- Упрощается восстановление базы данных: нужна только полная копия и последняя DIFF-копия.
- Исключаются проблемы с разрывом цепочки, так как каждая DIFF-копия независима от других DIFF-копий.



Рекомендуется хранить минимум два полных цикла резервных копий: две полные копии и DIFF-копии, созданные от них.

Настройка ClickHouse

Настройка базы данных ClickHouse состоит из следующих этапов:

1. Создание пользователя для работы с резервными копиями
2. Настройка утилиты clickhouse-backup



Все действия необходимо производить на физическом сервере или виртуальной машине, на которых развернута база данных ClickHouse.

Создание пользователя для работы с резервными копиями

1. Подключитесь к консольному клиенту ClickHouse, выполнив команду:

```
clickhouse-client
```

2. Создайте пользователя для работы с резервными копиями и выдайте ему необходимые права, выполнив следующие SQL-запросы:

```
-- Создание пользователя.
CREATE USER backup_user IDENTIFIED BY '<password>';

-- Выдача прав на чтение данных и метаданных.
GRANT SELECT ON *.* TO backup_user;
GRANT SHOW TABLES ON *.* TO backup_user;
GRANT SHOW DATABASES ON *.* TO backup_user;
GRANT SHOW COLUMNS ON *.* TO backup_user;
GRANT SHOW DICTIONARIES ON *.* TO backup_user;

-- Выдача права на создание локальной копии указанной партиции таблицы ClickHouse
(FREEZE).
GRANT ALTER FREEZE PARTITION ON *.* TO backup_user;

-- Выдача доступа к системным таблицам.
GRANT SELECT ON system.* TO backup_user;

-- Выдача прав на восстановление данных.
GRANT INSERT ON *.* TO backup_user;
GRANT CREATE TABLE ON *.* TO backup_user;
GRANT CREATE DATABASE ON *.* TO backup_user;
GRANT ALTER TABLE ON *.* TO backup_user;
GRANT DROP TABLE ON *.* TO backup_user;
```

Здесь:

- `<password>` — пароль нового пользователя.



Вы можете проверить выданные права, выполнив SQL-запрос:

```
SHOW GRANTS FOR backup_user;
```

Настройка утилиты clickhouse-backup

1. Создайте файл `config.yml` в директории `/etc/clickhouse-backup/`.
2. Добавьте в файл `config.yml` следующие строки:

```
general:
  disable_progress_bar: true
  max_file_size: 1073741824  # 1 ГБ.
  backups_to_keep_local: 0   # Резервные копии не хранятся локально (Backup Wrapper
управляет этим).
  backups_to_keep_remote: 0  # Резервные копии не удаляются автоматически.
  remote_storage: s3
```

```
clickhouse:
  host: localhost
  port: 9000
  username: backup_user
  password: "<password>"
  skip_tables:
    - system.*
    - INFORMATION_SCHEMA.*
    - information_schema.*
  skip_disks:
    - cold_disk_1
  timeout: 5m
  freeze_by_part: false
  freeze_by_part_where: ""

# S3 настройки передаются через переменные из файла окружения wrapper.
s3:
  access_key: "${S3_ACCESS_KEY}"
  secret_key: "${S3_SECRET_KEY}"
  bucket: "${S3_BUCKET}"
  endpoint: "${S3_ENDPOINT}"
  region: "${S3_REGION}"
  path: "${S3_PATH}"
  compression_format: tar
  compression_level: 1
  sse: ""
  disable_ssl: false
  force_path_style: true  # Для хранилища Minio.
```

Здесь:

- `<password>` — пароль пользователя backup_user, созданного ранее.

Многоуровневое хранения данных

Для высоконагруженных инсталляций рекомендуется разделение хранилища на уровни (hot/cold). Это позволяет:

- Оптимизировать ресурсы, затрачиваемые на хранение резервных копий.
- Ускорить резервное копирование, так как копируются только свежие данные.
- Ускорить восстановление данных из резервной копии.

Подробности по настройке многоуровневого хранилища ClickHouse приведены в разделе Настройка многоуровневого хранилища ClickHouse.

Настройка многоуровневого хранилища ClickHouse

База данных ClickHouse поддерживает подход многоуровневого хранилища: горячие данные хранятся на быстрых носителях (SSD), холодные данные — на медленных носителях (HDD).

Рекомендуется настроить многоуровневое хранилище, если у вас высоконагруженная инсталляция.

Данный подход выглядит следующим образом:

Уровень	**Тип диска**	**Данные**	**Резервное копирование**
Cold	SATA HDD	Данные старше 7 дней	Вручную
Hot	NVMe SSD	Данные за последние 7 дней	Ежедневно

Подход многоуровневого хранилища позволяет:

- Оптимизировать ресурсы, затрачиваемые на хранение резервных копий.
- Ускорить резервное копирование, так как копируются только свежие данные.
- Ускорить восстановление данных из резервной копии.
- Охватить почти все запросы приложений, так как большинство запросов происходит к горячим данным.

Цель данного подхода: создавать резервные копии только свежих данных в случае аварии.



Данный подход не подразумевает резервное копирование всей базы данных. Данные с дисков холодного хранения (cold) будут утеряны при восстановлении.

Рекомендуется предварительно настроить резервное копирование данных с дисков холодного хранения.



Особенности восстановления с многоуровневым хранилищем ClickHouse:

- Данные восстанавливаются на диск, установленный по умолчанию, или диск горячего хранения, даже если резервная копия была сделана с диска холодного хранения.
- После восстановления начнется миграция данных согласно TTL-правилам.
- При восстановлении данные с дисков холодного хранения будут утеряны.
- Используется сценарий Disaster Recovery, который позволяет оперативно восстановить работоспособность системы из свежих данных. Подразумевается, что при аварии данные с диска холодного хранения уже находятся на безопасном отдельном диске, который можно восстановить из архивов позже.

Подробности о восстановлении из резервных копий ClickHouse приведены в разделе Восстановление из резервных копий.

Настройка многоуровневого хранилища ClickHouse состоит из следующих этапов:

- Настройка дисков в ClickHouse
- Настройка clickhouse-backup для работы с многоуровневым хранилищем
- Настройка TTL для автоматической миграции данных

Все действия необходимо производить на физическом сервере или виртуальной машине, на которых развернута база данных ClickHouse.

Настройка дисков в ClickHouse

1. Подключитесь к консольному клиенту ClickHouse, выполнив команду:

```
clickhouse-client
```

2. Проверьте диски в ClickHouse, выполнив SQL-запрос:

```
SELECT name, path, type, total_space, free_space
FROM system.disks;
```

▼ *Пример вывода*

```
┌─name──────────────┬─path──────────────────────────────────────────────
──────────┬─type──────┬─total_space─┬─free_space──┐
│ cold_disk_1   │ /var/lib/clickhouse/cold-1/   │ Local   │ 103894020096   │
74152300544   │
│ default       │ /var/lib/clickhouse/          │ Local   │ 103894020096   │
74152300544   │
│ hot_disk_1    │ /var/lib/clickhouse/hot-1/    │ Local   │ 103894020096   │
74152300544   │
└───────────────┴──────────────────────────────────────────────────────
──────────┴───────────┴─────────────┴─────────────┘
```

3. Проверьте, какие таблицы расположены на дисках, с помощью SQL-запроса:

```
SELECT
    database,
    table,
    disk_name,
    sum(bytes_on_disk) as size
FROM system.parts
WHERE active
GROUP BY database, table, disk_name
ORDER BY database, table, disk_name;
```

▼ *Пример вывода*

```
┌─database──────┬─table─────────────────────┬─disk_name──────┬─
──────────────size─────┐
│ default       │ example_table    │ default       │ 148217891135   │
│ default       │ logs             │ default       │ 199151132155   │
│ system        │ query-log        │ cold_disk_1   │ 103895353096   │
```

```
  | system        | trace-log        | cold_disk_1   | 102882828226   |
  | analytics     | events           | hot_disk_1    | 103894020096   |
  | analytics     | sessions         | hot_disk_1    | 103894020096   |
  └─────────────────────────────────┴──────────────────────────────────────┴──────────
 ──────────────────────────┴──────────────────────────────────────┘
```

4. Создайте файл `storage.xml` в директории `/etc/clickhouse-server/config.d/`.
5. Добавьте в файл `storage.xml` данные о разделенных дисках. Например:

```
<clickhouse>
  <storage_configuration>
    <disks>
      <!-- HOT: быстрый диск SSD для свежих данных. -->
      <hot_disk_1>
        <path>/var/lib/clickhouse/hot-1/</path>
        <keep_free_space_bytes>10737418240</keep_free_space_bytes> <!-- 10GB
резерв. -->
      </hot_disk_1>

      <!-- COLD: медленный диск HDD для архивных данных. -->
      <cold_disk_1>
        <path>/var/lib/clickhouse/cold-1/</path>
        <keep_free_space_bytes>53687091200</keep_free_space_bytes> <!-- 50GB
резерв. -->
      </cold_disk_1>
    </disks>
    <policies>
      <siem_storage_policy>
        <volumes>
          <!-- При перечислении томов (volume) важен порядок, так как новые данные
пишутся в первый том. -->
          <hot_volume>
            <disk>hot_disk_1</disk>
          </hot_volume>
          <cold_volume>
            <disk>cold_disk_1</disk>
            <!-- Отключение merge на диске HDD для снижения I/O. -->
            <prefer_not_to_merge>true</prefer_not_to_merge>
          </cold_volume>
        </volumes>
        <!-- Перемещение партиции на диск холодного хранения (cold), если диск
горячего хранения (hot) заполнен более чем на 90%. -->
        <move_factor>0.1</move_factor>
      </siem_storage_policy>
    </policies>
  </storage_configuration>
</clickhouse>
```



Вы можете добавить новые диски в теге <disks>. Для новых дисков также нужно добавить новые данные в теге <volumes>.

Настройка clickhouse-backup для работы с многоуровневым хранилищем

Для корректной работы утилиты clickhouse-backup необходимо исключить диски холодного хранения из файла конфигурации утилиты. Для этого:

1. Откройте файл config.yml, выполнив команду:

```
sudo vi /etc/clickhouse-backup/config.yml
```

2. Добавьте следующие строки в файл в зависимости от сценария:

Исключение конкретных дисков

Укажите названия дисков холодного хранения, чтобы исключить их из процедуры резервного копирования:

```
skip_disks:
  - <disk_name>
```

Здесь:

- <disk_name> — имя диска холодного хранения, созданного ранее, например, cold_disk_1 или s3_archive.

Исключение по типам дисков

Укажите тип диска, чтобы исключить конкретные типы дисков:

```
skip_disk_types:
  - <disk_type>
```

Здесь:

- <disk_type> — тип диска, например, s3 или hdfs.

▼ *Пример заполненного файла* config.yml

```
general:
disable_progress_bar: true
backups_to_keep_local: 0
backups_to_keep_remote: 0
remote_storage: s3

clickhouse:
host: localhost
```

```
port: 9000
username: backup_user
password: "secure_password"

  skip_tables:
    - system.*
    - INFORMATION_SCHEMA.*
    - information_schema.*

  skip_disks:
    - cold_disk_1
    - s3_archive

  # skip_disk_types:
  #   - s3
  #   - hdfs

  timeout: 5m
```

Настройка TTL для автоматической миграции данных

TTL (time-to-live) в ClickHouse — это механизм управления жизненным циклом данных. Вы можете задать правила, по которым данные могут удаляться или перемещаться между дисками по истечении определенного времени.

Чтобы настроить TTL:

1. Подключитесь к консольному клиенту ClickHouse, выполнив команду:

   ```
   clickhouse-client
   ```

2. Создайте таблицу с TTL или добавьте настройки TTL в существующую таблицу. Примеры SQL-запросов для создания таблицы с TTL или настройки TTL в существующей таблице:

   Создание таблицы

   ```
   CREATE TABLE events ON CLUSTER '{cluster}'
   (
     event_time DateTime64(3) CODEC(Delta, ZSTD(1)),
     event_date Date DEFAULT toDate(event_time),
     event_type LowCardinality(String),
     severity LowCardinality(String),
     source_ip IPv4,
     dest_ip IPv4,
     message String CODEC(ZSTD(3)),
     raw_event String CODEC(ZSTD(5))
   )
   ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/events',
   '{replica}')
   ```

```
PARTITION BY toYYYYMMDD(event_date)
ORDER BY (event_date, event_type, event_time)

-- Настройка TTL-правил. Без них данные не будут перемещаться.
TTL
  -- Перемещает данные в том с диском холодного хранения через 7 дней.
  event_time + INTERVAL 7 DAY TO VOLUME '<volume_name>',
  -- Удаляет данные через 90 дней. Необходимо раскомментировать параметр для
применения.
  -- event_time + INTERVAL 90 DAY DELETE.
SETTINGS
  storage_policy = '<storage_policy>',
  -- Проверка TTL раз в сутки (по умолчанию 24 часа).
  merge_with_ttl_timeout = 86400,
  -- Удаление партиций без перезаписи данных.
  ttl_only_drop_parts = 1;
```

Здесь:

- `<volume_name>` — название тома с дисками холодного хранения из файла `storage.xml`, например, `cold_volume`.
- `<storage_policy>` — название политики хранилища из файла `storage.xml`, например, `siem_storage_policy`.

Изменение существующей таблицы

```
-- Шаг 1: Добавить настройки TTL.
ALTER TABLE <table_name> ON CLUSTER '{cluster}'
MODIFY TTL
  -- Перемещает данные в том с диском холодного хранения через 7 дней.
  event_time + INTERVAL 7 DAY TO VOLUME '<volume_name>',
  -- Удаляет данные через 90 дней. Для применения необходимо
раскомментировать параметр ниже.
  -- event_time + INTERVAL 90 DAY DELETE;

-- Шаг 2: Принудительно применить TTL (опционально). Данное действие создает
нагрузку на базу данных.
-- Без этого шага TTL применится при следующем слиянии (merge). Для
применения необходимо раскомментировать строки ниже.
-- ALTER TABLE <table_name> ON CLUSTER '{cluster}'
-- MATERIALIZE TTL;

-- Шаг 3: Мониторинг прогресса миграции.
SELECT
  partition,
  disk_name,
  formatReadableSize(sum(bytes_on_disk)) as size,
  sum(rows) as rows
FROM system.parts
WHERE table = '<table_name>' AND active
```

```
GROUP BY partition, disk_name
ORDER BY partition DESC;
```

Здесь:

- `<volume_name>` — название тома с дисками холодного хранения из файла `storage.xml`, например, `cold_volume`.
- `<table_name>` — название таблицы ClickHouse.

3. Ограничьте хранение системных логов, выполнив следующие SQL-запросы:

```
ALTER TABLE system.query_log MODIFY TTL event_date + INTERVAL 7 DAY;
ALTER TABLE system.trace_log MODIFY TTL event_date + INTERVAL 3 DAY;
ALTER TABLE system.metric_log MODIFY TTL event_date + INTERVAL 3 DAY;
ALTER TABLE system.text_log MODIFY TTL event_date + INTERVAL 7 DAY;
ALTER TABLE system.asynchronous_metric_log MODIFY TTL event_date + INTERVAL 3 DAY;
```

Установка и настройка Backup Wrapper



Сервис Backup Wrapper необходимо устанавливать, если управление резервными копиями планируется производить вне кластера системы. Если вы планируете управлять процессом резервного копирования из кластера системы, то можете перейти сразу к инструкции Настройка резервного копирования из кластера системы.

Требования для установки

Общие требования:

- Дистрибутивы ОС Linux, поддерживающие 64-битную архитектуру. Например, РЕД ОС, CentOS, Ubuntu, Debian.
- S3-совместимое хранилище для хранения резервных копий, например: MinIO, AWS S3, Yandex Object Storage.

Требования для PostgreSQL:

- Актуальная версия базы данных PostgreSQL.
- Версия утилиты wal-g: 3.0 и выше.

Требования для ClickHouse:

- Актуальная версия базы данных ClickHouse.
- Версия утилиты clickhouse-backup: 2.4 и выше.

Установка Backup Wrapper

Программа Backup Wrapper устанавливается как systemd-служба на физический сервер или

виртуальную машину. Процесс установки состоит из следующих шагов:

1. Установка утилиты wrapper на сервере, на котором расположена база данных.
2. Установка утилиты client на сервере, который будет запускать резервное копирование.

Установка утилиты wrapper

Чтобы установить утилиту wrapper:

1. Подключитесь к серверу, на котором расположена база данных.
2. Загрузите файлы установщика, выполнив команду:

   ```
   wget https://files.rvision.ru/platform/6dfaa56c92a665a855a9/backup-wrapper-
   1.0.0.tar.gz
   tar -xzvf backup-wrapper-1.0.0.tar.gz
   ```

3. Перейдите в директорию с установщиком, выполнив команду:

   ```
   cd backup-wrapper-1.0.0
   ```

4. Запустите установщик, выполнив команду:

   ```
   sudo ./install.sh wrapper <db_type>
   ```

   Здесь:

   - `<db_type>` — тип базы данных:
     - `postgres` — база данных PostgreSQL.
     - `clickhouse` — база данных ClickHouse.

   Вместе с установкой утилиты wrapper также происходит установка вспомогательных утилит wal-g и clickhouse-backup.

5. Откройте файл конфигурации `backup-wrapper.env`, выполнив команду:

   ```
   sudo vi /etc/backup-wrapper/backup-wrapper.env
   ```

6. Заполните обязательные параметры подключения:
   - `LISTEN_ADDR` — адрес и порт, на котором развернут сервис, например: `127.0.0.1:8080`.
   - `DB_PATH` — путь до директории данных утилиты wrapper, например: `/var/lib/backup-wrapper/backup.db`.
   - `EXECUTOR_TYPE` — тип базы данных, для которой необходимо сделать резервную копию:
     - `postgres` — PostgreSQL.

- `clickhouse` — ClickHouse.

7. Измените параметры аутентификации:
   - `AUTH_USER` — логин пользователя Backup Wrapper.
   - `AUTH_PASSWORD` — пароль пользователя Backup Wrapper.



Обязательно замените значения по умолчанию в полях `AUTH_USER` и `AUTH_PASSWORD`.

8. Настройте параметры для работы с резверными копиями PostgreSQL:
   - `WALG_PATH` — путь до исполняемого файла утилиты wal-g. По умолчанию: `/usr/local/bin/wal-g`.
   - `POSTGRES_DATA_PATH` — путь до директории с данными PostgreSQL (PGDATA). По умолчанию путь для версии PostgreSQL 16: `/var/lib/postgresql/16/data`.
9. Настройте параметр `CLICKHOUSE_BACKUP_PATH` для работы с резервными копиями ClickHouse. Параметр позволяет указать путь до исполняемого файла утилиты clickhouse-backup. По умолчанию: `/usr/local/bin/clickhouse-backup`.
10. Настройте уровень логирования, заполнив параметр `LOG_LEVEL` значением `info`.
11. Запустите утилиту wrapper, выполнив следующую команду:

```
sudo systemctl start backup-wrapper
```

12. Включите автозапуск утилиты при загрузке системы, выполнив команду:

```
sudo systemctl enable backup-wrapper
```



Для работы с утилитой wrapper также доступны следующие команды:

- Проверка статуса:

```
sudo systemctl status backup-wrapper
```

- Просмотр логов:

```
sudo journalctl -u backup-wrapper -f
```

- Перезапуск после изменения конфигурации:

```
sudo systemctl restart backup-wrapper
```

- Остановка утилиты:

```
sudo systemctl stop backup-wrapper
```

Установка утилиты client

Чтобы установить утилиту client:

1. Подключитесь к серверу, на котором будет запускаться процесс резервного копирования.
2. Загрузите файлы установщика, выполнив команду:

```
wget https://files.rvision.ru/platform/6dfaa56c92a665a855a9/backup-wrapper-
1.0.0.tar.gz
tar -xzvf backup-wrapper-1.0.0.tar.gz
```

3. Перейдите в директорию с установщиком, выполнив команду:

```
cd backup-wrapper-1.0.0
```

4. Запустите установщик, выполнив команду:

```
sudo ./install.sh client
```

5. Откройте для редактирования файл конфигурации `backup-client.env`, выполнив команду:

```
sudo vi /etc/backup-wrapper/backup-client.env
```

6. Настройте подключение к утилите wrapper, заполнив следующие параметры:
   - `WRAPPER_ADDR` — адрес подключения к утилите wrapper, указанный при настройке конфигурации утилиты wrapper, например, `localhost:8080`.
   - `AUTH_USER` — логин, указанный при настройке конфигурации утилиты wrapper.
   - `AUTH_PASSWORD` — пароль, указанный при настройке конфигурации утилиты wrapper.
7. Настройте подключение к S3-совместимому хранилищу, заполнив следующие параметры:
   - `S3_ENDPOINT` — URL-адрес S3-совместимого хранилища, например, `http://minio:9000`
   - `S3_BUCKET` — название контейнера (bucket) для хранения резервных копий.
   - `S3_PATH` — путь до директории внутри контейнера (bucket), где будут храниться резервные копии.
   - `S3_ACCESS_KEY` — идентификатор ключа доступа к S3-совместимому хранилищу.
   - `S3_SECRET_KEY` — секретный ключ доступа к S3-совместимому хранилищу.
8. Проверьте, что конфигурация утилиты client настроена корректно:
   a. Запустите утилиту вручную, выполнив команду:

```
sudo systemctl start backup-client
```

b. Проверьте логи выполнения утилиты, выполнив команду:

```
sudo journalctl -u backup-client -f
```

9. Включите таймер для автоматического резервного копирования, выполнив команду:

```
sudo systemctl enable --now backup-client.timer
```

10. Чтобы изменить расписание таймера, выполните следующую команду:

```
sudo systemctl edit backup-client.timer --full
```

11. Измените значение параметра `OnCalendar`.
12. Перезагрузите конфигурацию менеджера системы, выполнив команду:

```
sudo systemctl daemon-reload
```

13. Перезапустите таймер `backup-client.timer`, выполнив команду:

```
sudo systemctl restart backup-client.timer
```



Для управления задачами резервного копирования используется контроллер CronJob, если вы настраиваете Backup Wrapper из кластера, в котором развернута система.



Для работы с утилитой client также доступны следующие команды:

- Проверка расписания таймера:

```
systemctl list-timers backup-client.timer
```

- Отключение автоматического резервного копирования:

```
sudo systemctl disable --now backup-client.timer
```

- Просмотр истории запусков за последние 7 дней:

```
journalctl -u backup-client --since "7 days ago"
```

Проверка работы Backup Wrapper

Перед полной автоматизацией работы с резервными копиями рекомендуется проверить корректность заданных настроек.

Проверка настройки ClickHouse

Чтобы проверить корректность настроек ClickHouse:

1. Подключитесь к серверу, на котором развернута база данных ClickHouse.
2. Проверьте список таблиц баз данных, доступных для резервного копирования, с помощью команды:

   ```
   clickhouse-backup tables
   ```

3. Запустите тестовое резервное копирование, выполнив команду:

   ```
   clickhouse-backup create_remote test-only
   ```

4. Проверьте созданные резервные копии, выполнив команду:

   ```
   clickhouse-backup list remote
   ```



Если вы настраивали многоуровневое хранилище ClickHouse, то в выводе должны отобразиться резервные копии данных с диска горячего хранения (hot).

Проверка настройки Backup Wrapper

Чтобы проверить работоспособность сервиса:

1. Отправьте запрос на хост утилиты wrapper, выполнив команду:

   ```
   curl -u <auth_user>:<auth_password> http://<ip_port>/health
   ```

2. Создайте резервную копию через API-запрос, выполнив команду:

   ```
   curl -u <auth_user>:<auth_password> -X POST http://<ip_port>/api/v1/backup
   ```

3. Создайте резервную копию через утилиту client, выполнив команду:

   ```
   /usr/local/bin/backup-client --env-file=/etc/backup-wrapper/backup-client.env
   ```

```
--wait
```

4. Посмотрите список созданных резервных копий, выполнив команду:

```
curl -u <auth_user>:<auth_password> http://<ip_port>/api/v1/backups
```

Здесь:

- `<auth_user>` — логин для подключения к утилите wrapper.
- `<auth_password>` — пароль подключения к утилите wrapper.
- `<ip_port>` — IP-адрес и порт, на котором развернута утилита wrapper, например `127.0.0.1:8080`.

Удаление Backup Wrapper

Процесс удаления Backup Wrapper состоит из следующих этапов:

- Удаление утилиты wrapper
- Удаление утилиты client

Удаление утилиты wrapper

Чтобы удалить утилиту wrapper:

1. Подключитесь к серверу, на котором расположена база данных.
2. Перейдите в директорию установщика, выполнив команду:

```
cd <path_to_directory>
```

   Здесь:

   - `<path_to_directory>` — путь до директории, в которую был загружен установщик утилиты.

3. Выполните удаление утилиты с помощью одной из команд:
   - Без удаления данных и резервных копий:

```
sudo ./uninstall.sh
```

   - С удалением всех данных:

```
sudo ./uninstall.sh --purge
```

Удаление утилиты client

Чтобы удалить утилиту client:

1. Подключитесь к серверу, на котором запускается процесс резервного копирования.
2. Перейдите в директорию установщика, выполнив команду:

   ```
   cd <path_to_directory>
   ```

   Здесь:

   - `<path_to_directory>` — путь до директории, в которую был загружен установщик утилиты.

3. Выполните удаление утилиты с помощью одной из команд:
   - Без удаления данных и резервных копий:

     ```
     sudo ./uninstall.sh
     ```

   - С удалением всех данных:

     ```
     sudo ./uninstall.sh --purge
     ```

Резервное копирование внешних баз данных

Резервное копирование внешних баз данных производится с помощью сервиса Backup Wrapper. Доступны следующие варианты управления сервисом Backup Wrapper и резервным копированием:

- Управление из кластера Kubernetes, в котором установлена система. Сервис Backup Wrapper устанавливается вместе с системой. Работа с резервными копиями внешних баз данных происходит с помощью контроллера CronJob и секретов с данными подключения к S3-совместимому хранилищу.
- Управление с сервера, на котором установлен сервис Backup Wrapper.

Настройка резервного копирования из кластера системы

Данный сценарий подойдет, если резервное копирование внешних баз данных настраивается из кластера системы.

Требования

- Настроены базы данных PostgreSQL и ClickHouse для работы с резервными копиями.
- S3-совместимое хранилище для хранения резервных копий, например: MinIO, AWS S3, Yandex Object Storage.
- Доступ к утилитам kubectl и evoctl.

Настройка резервного копирования

1. Подключитесь к кластеру.
2. Подключитесь к master-узлу или машине, с которой производилась установка системы.

3. Настройте резервное копирование с помощью одной из команд в зависимости от сценария:

   - ClickHouse на одном сервере:

     ```
     evoctl instance backup enable \
       --namespace <namespace> \
       --external \
       --name ch-main \
       --ch-backup \
       --wrapper-addr "<wrapper_addr>" \
       --auth-user <auth_user> \
       --auth-password '<auth_password>' \
       --s3-endpoint "<s3_endpoint>" \
       --s3-bucket <s3_bucket> \
       --s3-access-key <s3_access_key> \
       --s3-secret-key '<s3_secret_key>'
     ```

   - ClickHouse с несколькими шардами. Для каждого шарда ClickHouse включите резервное копирование, выполнив команду:

     ```
     evoctl instance backup enable \
       --namespace <namespace> \
       --external \
       --name <shard_name> \
       --ch-backup \
       --wrapper-addr "<wrapper_addr>" \
       --auth-user <auth_user> \
       --auth-password '<auth_password>' \
       --s3-endpoint "<s3_endpoint>" \
       --s3-bucket <s3_bucket> \
       --s3-access-key <s3_access_key> \
       --s3-secret-key '<s3_secret_key>'
     ```

     Для каждого шарда ClickHouse укажите свой адрес подключения к утилите wrapper.

   - PostgreSQL:

     ```
     evoctl instance backup enable \
       --namespace <namespace> \
       --external \
       --name pg-main \
       --pg-backup \
       --wrapper-addr "<wrapper_addr>" \
       --auth-user <auth_user> \
       --auth-password '<auth_password>' \
       --s3-endpoint "<s3_endpoint>" \
       --s3-bucket <s3_bucket> \
     ```

```
  --s3-access-key <s3_access_key> \
  --s3-secret-key '<s3_secret_key>'
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

- `<shard_name>` — название шарда ClickHouse.
- `<wrapper_addr>` — адрес подключения к утилите wrapper, указанный при настройке ее конфигурации, например, `http://ch-server.company.local:8080`.
- `<auth_user>` — логин, указанный при настройке конфигурации утилиты wrapper.
- `<auth_password>` — пароль, указанный при настройке конфигурации утилиты wrapper.
- `<s3_endpoint>` — URL-адрес S3-совместимого хранилища, например, `http://minio:9000`.
- `<s3_bucket>` — название контейнера (bucket) для хранения резервных копий.
- `<s3_access_key>` — идентификатор ключа доступа к S3-совместимому хранилищу.
- `<s3_secret_key>` — секретный ключ доступа к S3-совместимому хранилищу.

После выполнения команды создадутся контроллер CronJob для управления резервными копированием и секрет с данными подключения к S3-совместимому хранилищу.

4. Проверьте, что контроллер CronJob был корректно создан, с помощью команды:

```
kubectl get cronjob -n <namespace> -l app.kubernetes.io/name=external-backup
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

5. Проверьте, что секрет был корректно создан, с помощью команды:

```
kubectl get secret -n <namespace> -l app.kubernetes.io/name=external-backup
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Работа с резервными копиями баз данных

Доступные операции:

- Просмотр статуса резервного копирования
- Ручной запуск резервного копирования

- Отключение резервного копирования



Все команды утилиты evoctl необходимо выполнять на master-узле или машине, с которой производилась установка системы.

Просмотр статуса резервного копирования

Чтобы посмотреть статус резервного копирования, выполните команду:

```
evoctl instance backup info -n <namespace>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Ручной запуск резервного копирования

Чтобы запустить резервное копирование вручную:

1. Подключитесь к кластеру.
2. Создайте задачу (job) из контроллера CronJob резервного копирования, выполнив команду:

   ```
   kubectl create job \
     --from=cronjob/<cronjob_name> \
     <backup_name>-$(date +%Y%m%d-%H%M%S) \
     -n <namespace>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<cronjob_name>` — название контроллера CronJob, созданного на этапе настройки.
   - `<backup_name>` — название задачи резервного копирования.

3. Проверьте статус созданной задачи, выполнив команду:

   ```
   kubectl get jobs -n <namespace> -l app.kubernetes.io/name=external-backup
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

4. При необходимости проверьте логи задачи резервного копирования, выполнив команду:

   ```
   kubectl logs job/<job_name> -n <namespace>
   ```

   Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<job_name>` — название созданной задачи (job).

Отключение резервного копирования

Чтобы отключить резервное копирование, выполните одну из команд:

- Отключить резервное копирование всех внешних баз данных:

```
evoctl instance backup disable -n <namespace> --external --name all
```

- Отключить резервное копирование конкретной внешней базы данных:

```
evoctl instance backup disable -n <namespace> --external --name <db_name>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<db_name>` — название базы данных.

Настройка резервного копирования вне кластера системы

Данный сценарий подойдет, если резервное копирование внешних баз данных настраивается на сервере вне кластера системы.

Требования

- Установлен и настроен сервис Backup Wrapper.
- Настроены базы данных PostgreSQL и ClickHouse для работы с резервными копиями.
- S3-совместимое хранилище для хранения резервных копий, например: MinIO, AWS S3, Yandex Object Storage.

Настройка резервного копирования

Чтобы настроить резервное копирование:

1. Подключитесь к серверу, с которого планируется запуск резервного копирования.
2. Проверьте, что подключение к утилите wrapper настроено корректно, с помощью команд:

```
curl -u <auth_user>:<auth_password> <wrapper_addr>/health
curl -u <auth_user>:<auth_password> <wrapper_addr>/ready
```

Здесь:

- `<auth_user>` — логин, указанный при настройке конфигурации утилиты wrapper.
- `<auth_password>` — пароль, указанный при настройке конфигурации утилиты wrapper.

- `<wrapper_addr>` — адрес подключения к утилите wrapper, указанный при настройке ее конфигурации, например, `http://localhost:8080`.

3. Запустите автоматическое резервное копирование, выполнив команду:

```
./backup-wrapper backup --wait
```



Также вы можете выполнить принудительное резервное копирование, выполнив команду в зависимости от типа резервной копии:

- Создание полной резервной копии:

```
./backup-wrapper backup --type=full --wait
```

- Создание резервной Incremental-копии (только для PostgreSQL):

```
./backup-wrapper backup --type=incremental --wait
```

4. Проверьте, что резервные копии были созданы корректно, с помощью команды:

PostgreSQL

```
wal-g backup-list
```

ClickHouse

```
clickhouse-backup list remote
```

5. Настройте автоматизацию резервного копирования в зависимости от сценария:

Через Cron

Для автоматизации вы можете использовать встроенный планировщик задач Cron в Unix-подобных ОС. Для этого создайте файл `backup-wrapper` в директории `/etc/cron.d/` со следующим содержимым:

```
<schedule> backup-user /bin/bash -c 'source /etc/backup-wrapper-
v1.0.0/backup-client.env && /usr/local/bin/backup-wrapper backup --wait' >>
/var/log/backup.log 2>&1
```

Здесь:

- `<schedule>` — расписание резервного копирования в Cron-формате. Например,

`0 3 * * *` — резервные копии будут создаваться ежедневно в 03:00.



Чтобы составить cron-выражение, можно воспользоваться генератором, например, crontab.guru.

Через Systemd Timer

Вы можете автоматизировать резервное копирование с помощью механизма Systemd Timer, так как сервис Backup Wrapper устанавливается как systemd-служба. Для этого:

1. Создайте файл `backup-wrapper.timer` в директории `/etc/systemd/system/` со следующим содержимым:

```
[Unit]
Description=Daily backup timer

[Timer]
OnCalendar=<schedule>
Persistent=true
RandomizedDelaySec=300

[Install]
WantedBy=timers.target
```

   Здесь:

   - `<schedule>` — задайте расписание таймера в формате Calendar Events. Например: `*-*-* 02:00:00` — резервное копирование будет запускаться каждый день в два часа ночи.

2. Создайте файл `backup-wrapper.service` в директории `/etc/systemd/system/` со следующим содержимым:

```
[Unit]
Description=Run backup via backup-wrapper

[Service]
Type=oneshot
EnvironmentFile=/etc/backup-wrapper-v1.0.0/backup-client.env
ExecStart=/usr/local/bin/backup-wrapper backup --wait
User=backup-user
StandardOutput=journal
StandardError=journal
```

3. Перезагрузите конфигурацию менеджера системы, выполнив команду:

```
sudo systemctl daemon-reload
```

4. Включите Systemd Timer, выполнив команду:

```
sudo systemctl enable --now backup-wrapper.timer
```

5. Проверьте, что Systemd Timer запущен и работает корректно, с помощью команды:

```
systemctl list-timers | grep backup
```

6. Проверьте, что задачи резервного копирования корректно создаются, с помощью следующих команд:
   - Вывести список последних задач:

```
curl -u admin:password http://localhost:8080/api/v1/jobs
```

   - Вывести задачи со статусом failed:

```
curl -u admin:password http://localhost:8080/api/v1/jobs?status=failed
```

Восстановление из резервных копий

В данном разделе описаны способы восстановления баз данных и сервисов платформы.

Содержание раздела:

- Восстановление внутренних баз данных
- Восстановление внешних баз данных
  - Восстановление базы данных PostgreSQL
  - Восстановление базы данных ClickHouse
  - Устранение проблем при восстановлении внешних баз данных
- Восстановление сервисов платформы

Восстановление внутренних баз данных

Требования

- База данных развернута внутри кластера Kubernetes, в котором установлена система.
- S3-совместимое хранилище для хранения резервных копий, например: MinIO, AWS S3, Yandex Object Storage.
- Остановлены все приложения, которые записывали данные в восстанавливаемую базу данных.
- Доступ к утилитам kubectl, helm и evoctl.

Восстановление базы данных PostgreSQL

Доступны следующие сценарии восстановления базы данных PostgreSQL:

- Восстановление из последней резервной копии
- Восстановление из конкретной резервной копии



При восстановлении все текущие данные в базе данных будут удалены. Рекомендуется предварительно проверить восстановление на тестовом кластере.

Восстановление из последней резервной копии

Чтобы восстановить базу данных:

1. Подключитесь к кластеру, в котором развернута база данных.
2. Запустите процесс восстановления, выполнив команду:

   ```
   helm upgrade postgresql /opt/r-vision/infra/helm/postgresql \
     --reuse-values \
     --set backupWrapper.restore.enabled=true --set primary.replicaCount=0 \
     --set backupWrapper.restore.cleanDataDir=true \
     --timeout 30m
   ```

   После применения команды все текущие данные в базе данных PostgreSQL будут удалены.

   Если у вас большое количество данных, рекомендуется увеличить значение параметра `--timeout`.

3. После восстановления базы данных запустите под PostgreSQL и уберите флаги восстановления. Для этого выполните команду:

   ```
   helm upgrade postgresql /opt/r-vision/infra/helm/postgresql \
     --reuse-values \
     --set primary.replicaCount=1 \
     --set backupWrapper.restore.enabled=false
   ```

4. Дождитесь восстановления пода базы данных PostgreSQL.

Восстановление из конкретной резервной копии

Чтобы восстановить базу данных:

1. Подключитесь к кластеру, в котором развернута база данных.
2. Посмотрите список резервных копий, выполнив команду:

```
kubectl exec -it <postgresql> -c backup-wrapper -- wal-g backup-list
```

Здесь:

- `<postgresql>` — имя пода, на котором развернута база данных PostgreSQL. Например, `postgresql-0`.

3. Восстановите базу данных, выполнив команду:

```
helm upgrade <postgresql> . --reuse-values \
  --set primary.replicaCount=0 \
  --set backupWrapper.restore.enabled=true \
  --set backupWrapper.restore.backupName=<backup_name> \
  --timeout 30m
```



После применения команды все текущие данные в базе данных PostgreSQL будут удалены.



Если у вас большое количество данных, рекомендуется увеличить значение параметра `--timeout`.

Здесь:

- `<postgresql>` — имя пода, на котором развернута база данных PostgreSQL. Например, `postgresql-0`.
- `<backup_name>` — название резервной копии, например, `base_000000010000000000000005`.

4. После восстановления запустите под PostgreSQL и уберите флаги восстановления. Для этого выполните команду:

```
helm upgrade <postgresql> --reuse-values \
  --set primary.replicaCount=1 \
  --set backupWrapper.restore.enabled=false
```

Здесь:

- `<postgresql>` — имя пода, на котором развернута база данных PostgreSQL. Например, `postgresql-0`.

5. Дождитесь восстановления пода базы данных PostgreSQL.

Восстановление базы данных ClickHouse



При восстановлении все текущие данные в базе данных будут удалены. Рекомендуется предварительно проверить восстановление на тестовом кластере.

Чтобы восстановить базу данных:

1. Подключитесь к кластеру, в котором развернута база данных.
2. Укажите переменные окружения для задачи (job) восстановления, выполнив команду:

```
JOB_NAME="restore-ch-$(date +%s)"
NAMESPACE="<namespace>"
```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

3. Создайте новую задачу из существующего контроллера CronJob, выполнив команду:

```
kubectl get cronjob clickhouse-backup-shard0 -n "$NAMESPACE" -o json | \
  jq --arg name "$JOB_NAME" '{
    apiVersion: "batch/v1",
    kind: "Job",
    metadata: {name: $name, namespace: .metadata.namespace},
    spec: .spec.jobTemplate.spec
  }
  | .spec.template.spec.containers[0].command = ["/bin/sh", "-c", "echo READY &&
sleep 7200"]
  | .spec.template.spec.containers[0].args = []
  | .spec.activeDeadlineSeconds = 7200
  | .spec.template.spec.restartPolicy = "Never"
  ' | kubectl apply -n "$NAMESPACE" -f -
```

4. Дождитесь запуска задания (job). Для отслеживания статуса выполните команду:

```
kubectl wait --for=condition=Ready pod -l "job-name=$JOB_NAME" \
  -n "$NAMESPACE" --timeout=120s
```

5. Подключитесь к поду заданий (job), выполнив команду:

```
kubectl exec -it -n "$NAMESPACE" "job.batch/$JOB_NAME" -- /bin/sh
```

6. Задайте в переменной `S3_PATH` путь до директории S3-совместимого хранилища, где хранятся резервные копии. Для этого выполните команду:

```
export S3_PATH="<path_to_backups>"
```

   Здесь:

- <path_to_backups> — путь до директории S3-совместимого хранилища с резервными копиями. Например, backups/clickhouse/shard0/2026/02/09.

7. Откройте список доступных резервных копий, выполнив команду:

```
clickhouse-backup list remote
```

8. Восстановите базу данных, выполнив команду:

```
clickhouse-backup restore_remote <backup_name>
```

Здесь:

- <backup_name> — название резервной копии.



После применения команды все текущие данные в базе данных ClickHouse будут удалены.



Для восстановления конкретной таблицы в базе данных выполните команду:

```
clickhouse-backup restore_remote <backup_name> --tables
"<database>.<table_pattern>"
```

Здесь:

- <backup_name> — название резервной копии.
- <database> — название базы данных.
- <table_pattern> — название или шаблон названия таблицы.

9. Проверьте, что восстановление прошло успешно, с помощью команды:

```
clickhouse-backup list local
```

10. После завершения отключитесь от пода, выполнив команду:

```
exit
```

11. Удалите задание (job) восстановления базы данных, выполнив команду:

```
kubectl delete job "$JOB_NAME" -n "$NAMESPACE"
```

Восстановление внешних баз данных

Восстановление баз данных производится вручную при помощи утилит wal-g и clickhouse-backup.



Сервис Backup Wrapper не выполняет восстановление баз данных. Данное ограничение предусмотрено в целях безопасности системы.

Сценарии, при которых требуется восстановление данных

Сценарий	Рекомендуемый подход восстановления
Потеря данных	Из последней резервной копии
Повреждение базы данных	Из последней рабочей версии резервной копии
Откат ошибочных изменений	Откат базы данных на определенный момент времени (PITR)
Восстановление после масштабного сбоя (Disaster Recovery)	На резервный сервер
Тестирование или аудит	В изолированную среду



Все действия необходимо производить на физическом сервере или виртуальной машине, на которых развернута база данных.

Требования

- Доступ к S3-совместимому хранилищу с резервными копиями, например: MinIO, AWS S3, Yandex Object Storage.
- Остановлены все приложения, которые записывали данные в восстанавливаемую базу данных.
- Сделана резервная копия текущего состояния базы данных, если это возможно.
- Процедура восстановления проверена в тестовой среде.
- На сервере базы данных достаточно дискового пространства.
- Подготовлен план отката изменений.

Содержание раздела:

- Восстановление базы данных PostgreSQL
- Восстановление базы данных ClickHouse
- Устранение проблем при восстановлении внешних баз данных

Восстановление базы данных PostgreSQL

Восстановление базы данных PostgreSQL состоит из следующих этапов:

1. Проверка соединения утилиты wal-g
2. Восстановление базы данных
3. Проверка после восстановления

Проверка соединения утилиты wal-g

Чтобы проверить соединение:

1. Установите переменные подключения к S3-совместимому хранилищу, если они не были заданы на этапе настройки Backup Wrapper. Для этого выполните команды:

   ```
   export WALG_S3_PREFIX="<walg_s3_prefix>"
   export AWS_ACCESS_KEY_ID="<aws_access_key>"
   export AWS_SECRET_ACCESS_KEY="<aws_secret_key>"
   export AWS_ENDPOINT="<aws_endpoint>"
   export AWS_S3_FORCE_PATH_STYLE=true
   export AWS_REGION="<aws_region>"
   ```

   Здесь:

   - `<walg_s3_prefix>` — путь к хранилищу, в котором будут храниться резервные копии, например, `s3://backups/postgresql/wal`.
   - `<aws_access_key>` — идентификатор ключа доступа к S3-совместимому хранилищу.
   - `<aws_secret_key>` — секретный ключ доступа к S3-совместимому хранилищу.
   - `<aws_endpoint>` — URL-адрес S3-совместимого хранилища, например, `http://minio:9000`
   - `<aws_region>` — код региона, в котором развернуты и выполняются ваши сервисы и функции AWS, например, `ru-central1-a`.

2. Проверьте доступ утилиты wal-g к S3-совместимому хранилищу, выполнив команду:

   ```
   wal-g backup-list
   ```

Восстановление базы данных

Возможные сценарии восстановления базы данных PostgreSQL:

- Полное восстановление из последней резервной копии
- Восстановление из конкретной резервной копии
- Откат базы данных на определенный момент времени (PITR)

Полное восстановление из последней резервной копии

Чтобы восстановить базу данных:

1. Остановите сервис PostgreSQL, выполнив команду:

   ```
   systemctl stop postgresql
   ```

2. Очистите директорию с данными PostgreSQL (PGDATA), выполнив команду:

```
rm -rf /<path_to_pgdata>/*
```



После очистки все текущие данные в PostgreSQL будут утеряны. Предварительно проверьте восстановление из выбранной резервной копии в тестовой среде. Это снизит риск полной потери данных.

Здесь:

- `<path_to_pgdata>` — путь к директории с данными PostgreSQL, например, `/var/lib/pgsql/16/data`.

3. Восстановите базу данных из последней резервной копии, выполнив команду:

```
wal-g backup-fetch /<path_to_pgdata> LATEST
```

Здесь:

- `<path_to_pgdata>` — путь к директории с данными PostgreSQL, например, `/var/lib/pgsql/16/data`.

4. Создайте файл `recovery.signal` для применения WAL-архивации, выполнив команду:

```
touch /<path_to_pgdata>/recovery.signal
```

Здесь:

- `<path_to_pgdata>` — путь к директории с данными PostgreSQL, например, `/var/lib/pgsql/16/data`.

5. Настройте параметр `restore_command` в файле конфигурации `postgresql.conf`, выполнив команду:

```
echo "restore_command = 'wal-g wal-fetch %f %p'" >>
/<path_to_pgdata>/postgresql.conf
```

Здесь:

- `<path_to_pgdata>` — путь к директории с данными PostgreSQL, например, `/var/lib/pgsql/16/data`.

6. Запустите сервис PostgreSQL, выполнив команду:

```
systemctl start postgresql
```

7. Дождитесь окончания процесса восстановления. Для отслеживания логов восстановления в командной строке можно выполнить команду:

```
tail -f /var/log/postgresql/postgresql-*.log | grep -i recovery
```

Дождитесь появления записи:

```
database system is ready to accept connections
```

Восстановление из конкретной резервной копии

Чтобы восстановить базу данных:

1. Остановите сервис PostgreSQL, выполнив команду:

```
systemctl stop postgresql
```

2. Очистите директорию с данными PostgreSQL (PGDATA), выполнив команду:

```
rm -rf /<path_to_pgdata>/*
```



После очистки все текущие данные в PostgreSQL будут утеряны. Предварительно проверьте восстановление из выбранной резервной копии в тестовой среде. Это снизит риск полной потери данных.

Здесь:

- `<path_to_pgdata>` — путь к директории с данными PostgreSQL, например, `/var/lib/pgsql/16/data`.

3. Посмотрите список резервных копий, выполнив команду:

```
wal-g backup-list
```

▼ *Пример вывода*

```
name                          last_modified        wal_segment_backup_start
base_000000010000000000000005 2026-01-27T10:00:00Z 000000010000000000000005
base_000000010000000000000010 2026-01-28T10:00:00Z 000000010000000000000010
```

4. Восстановите базу данных из нужной резервной копии, выполнив команду:

```
wal-g backup-fetch /<path_to_pgdata> <backup_name>
```

Здесь:

- `<path_to_pgdata>` — путь к директории с данными PostgreSQL, например, `/var/lib/pgsql/16/data`.
- `<backup_name>` — название резервной копии, например, `base_000000010000000000000010`.

5. Настройте параметр `restore_command` в файле конфигурации `postgresql.conf`, выполнив команду:

```
echo "restore_command = 'wal-g wal-fetch %f %p'" >>
/<path_to_pgdata>/postgresql.conf
```

Здесь:

- `<path_to_pgdata>` — путь к директории с данными PostgreSQL, например, `/var/lib/pgsql/16/data`.

6. Запустите сервис PostgreSQL, выполнив команду:

```
systemctl start postgresql
```

7. Дождитесь окончания процесса восстановления. Для отслеживания логов восстановления в командной строке можно выполнить команду:

```
tail -f /var/log/postgresql/postgresql-*.log | grep -i recovery
```

Дождитесь появления записи:

```
database system is ready to accept connections
```

Откат базы данных на определенный момент времени (PITR)

Чтобы восстановить базу данных:

1. Остановите сервис PostgreSQL, выполнив команду:

```
systemctl stop postgresql
```

2. Очистите директорию с данными PostgreSQL (PGDATA), выполнив команду:

```
rm -rf /<path_to_pgdata>/*
```



После очистки все текущие данные в PostgreSQL будут утеряны. Предварительно проверьте восстановление из выбранной резервной копии в тестовой среде. Это снизит риск полной потери данных.

Здесь:

- <path_to_pgdata> — путь к директории с данными PostgreSQL, например, /var/lib/pgsql/16/data.

3. Восстановите базу данных из резервной копии, которая была создана до нужного момента времени. Для этого выполните команду:

```
wal-g backup-fetch /<path_to_pgdata> <backup_name>
```

Здесь:

- <path_to_pgdata> — путь к директории с данными PostgreSQL, например, /var/lib/pgsql/16/data.
- <backup_name> — название резервной копии, например, base_000000010000000000000010.

4. Настройте PITR в файле конфигурации postgresql.conf, выполнив команду:

```
cat >> /<path_to_pgdata>/postgresql.conf << 'EOF' restore_command = 'wal-g wal-
fetch %f %p' recovery_target_time = '<target_time>' recovery_target_action =
'promote' EOF
```

Здесь:

- <path_to_pgdata> — путь к директории с данными PostgreSQL, например, /var/lib/pgsql/16/data.
- <target_time> — дата и время, до которых необходимо откатить базу данных, например, 2026-01-28 10:00:00.

5. Создайте файл recovery.signal для применения WAL-архивации, выполнив команду:

```
touch /<path_to_pgdata>/recovery.signal
```

Здесь:

- <path_to_pgdata> — путь к директории с данными PostgreSQL, например, /var/lib/pgsql/16/data.

6. Запустите сервис PostgreSQL, выполнив команду:

```
systemctl start postgresql
```

7. Дождитесь окончания процесса восстановления. Для отслеживания логов восстановления в командной строке можно выполнить команду:

```
tail -f /var/log/postgresql/postgresql-*.log | grep -i recovery
```

Дождитесь появления записи:

```
database system is ready to accept connections
```

Проверка после восстановления

Чтобы проверить, что восстановление базы данных прошло корректно:

1. Проверьте логи на наличие ошибок. Для вывода логов с ошибками выполните команду:

   ```
   grep -i error /var/log/postgresql/postgresql-*.log
   ```

2. Проверьте статус репликации при наличии реплик баз данных. Для этого выполните команду:

   ```
   psql -c "SELECT * FROM pg_stat_replication;"
   ```

3. Проверьте целостность данных, выполнив команду:

   ```
   psql -c "SELECT count(*) FROM <table_name>;"
   ```

   Здесь:

   - `<table_name>` — название таблицы базы данных.

4. Проверьте, что параметр резервного копирования `archive_command` включен для будущих восстановлений, с помощью команды:

   ```
   psql -c "SHOW archive_mode; SHOW archive_command;"
   ```

Восстановление базы данных ClickHouse

Восстановление базы данных ClickHouse состоит из следующих этапов:

1. Проверка соединения с S3-совместимым хранилищем
2. Восстановление базы данных
3. Проверка после восстановления базы данных

Проверка соединения с S3-совместимым хранилищем

Чтобы проверить соединение:

1. Установите переменные подключения к S3-совместимому хранилищу, если они не были заданы на этапе настройки Backup Wrapper. Для этого выполните команды:

   ```
   export REMOTE_STORAGE=s3
   ```

```
export S3_ENDPOINT="<aws_endpoint>"
export S3_BUCKET="<bucket>"
export S3_PATH="<path_to_backups>"
export S3_ACCESS_KEY="<access_key>"
export S3_SECRET_KEY="<secret_key>"
export S3_FORCE_PATH_STYLE=true
export S3_DISABLE_SSL=true
```

Здесь:

- `<aws_endpoint>` — URL-адрес S3-совместимого хранилища, например, `http://your-s3-endpoint:9000`
- `<bucket>` — название контейнера (bucket) для хранения резервных копий.
- `<path_to_backups>` — путь к директории внутри контейнера (bucket), где будут храниться резервные копии.
- `<aws_access_key>` — идентификатор ключа доступа к S3-совместимому хранилищу.
- `<aws_secret_key>` — секретный ключ доступа к S3-совместимому хранилищу.

2. Проверьте доступ утилиты wal-g к S3-совместимому хранилищу, выполнив команду:

```
clickhouse-backup list remote
```

Восстановление базы данных

Доступны следующие сценарии восстановления базы данных ClickHouse:

- Восстановление из последней резервной копии
- Восстановление из конкретной резервной копии
- Восстановление данных в другую базу данных
- Восстановление данных в другую таблицу

Восстановление из последней резервной копии

Чтобы восстановить базу данных из последней резервной копии, выполните команду:

```
clickhouse-backup restore_remote LATEST
```

Восстановление из конкретной резервной копии

Чтобы восстановить базу данных:

1. Откройте список резервных копий, выполнив команду:

```
clickhouse-backup list remote
```

2. Восстановите базу данных. Способ восстановления зависит от типа резервной копии:

FULL

Для восстановления резервной FULL-копии выполните команду:

```
clickhouse-backup restore_remote <full_backup_name>
```

Здесь:

- `<full_backup_name>` — название резервной FULL-копии, например, `full-2026-01-27T10-00-00`.

DIFF

Для восстановления резервной DIFF-копии выполните команду:

```
clickhouse-backup restore_remote <diff_backup_name>
```

Здесь:

- `<diff_backup_name>` — название резервной DIFF-копии, например, `inc-2026-01-28T10-00-00`.

Восстановление данных в другую базу данных

Чтобы восстановить данные в новую базу данных:

1. Откройте список резервных копий, выполнив команду:

```
clickhouse-backup list remote
```

2. Восстановите данные в новую базу данных, выполнив команду:

```
clickhouse-backup restore_remote --restore-database-mapping <old_db>:<new_db>
<backup_name>
```

Здесь:

- `<backup_name>` — название резервной копии.
- `<old_db>` — название существующей базы данных.
- `<new_db>` — название новой базы данных.

Восстановление данных в другую таблицу

Чтобы восстановить данные в новую таблицу:

1. Откройте список таблиц необходимой резервной копии, выполнив команду:

   ```
   clickhouse-backup tables --remote-backup <backup_name>
   ```

   Здесь:

   - `<backup_name>` — название резервной копии.

2. Восстановите данные в новую таблицу, выполнив команду:

   ```
   clickhouse-backup restore_remote --restore-table-mapping <old_table>:<new_table>
   <backup_name>
   ```

   Здесь:

   - `<backup_name>` — название резервной копии.
   - `<old_table>` — название существующей таблицы.
   - `<new_table>` — название новой таблицы.

Проверка после восстановления базы данных

Чтобы провести проверку:

1. Проверьте целостность восстановленных данных, выполнив команды:

   ```
   clickhouse-client -q "SELECT count() FROM <database_table>"
   ```

   Здесь:

   - `<database_table>` — название восстанавливаемой базы данных и таблицы, например, `your_database.your_table`.

2. Проверьте данные в системных таблицах, выполнив команду:

   ```
   clickhouse-client -q "SELECT count() FROM system.parts WHERE active"
   clickhouse-client -q "SELECT database, table, count() FROM system.parts WHERE
   active GROUP BY database, table"
   ```

3. Проверьте наличие ошибок в логах. Отобразить логи с ошибками можно с помощью команды:

   ```
   grep -i error /var/log/clickhouse-server/clickhouse-server.log | tail -50
   ```

4. При необходимости оптимизируйте восстановленные таблицы, осуществив принудительное слияние (merge) всех частей данных. Для этого выполните команду:

```
clickhouse-client -q "OPTIMIZE TABLE <table> FINAL"
```

Здесь:

- `<table>` — название таблицы базы данных.

Устранение проблем при восстановлении внешних баз данных

При восстановлении внешних баз данных могут возникнуть следующие проблемы:

- PostgreSQL: отсутствуют файлы WAL-архивации
- PostgreSQL: конфликт версий истории базы данных
- PostgreSQL: отказ в доступе к PGDATA
- ClickHouse: разрыв в цепочке резервных копий
- ClickHouse: ошибка восстановления данных в существующую таблицу
- ClickHouse: ошибка отключения таблицы

PostgreSQL: отсутствуют файлы WAL-архивации

Проявление: при попытке восстановления базы данных выводится следующее сообщение:

```
FATAL: could not locate a valid checkpoint record
```

Причина: отсутствуют файлы WAL-архивации, которые необходимы для корректного восстановления базы данных.

Решение: проверьте, что параметр `restore_command` настроен корректно.

PostgreSQL: конфликт версий истории базы данных

Проявление: при попытке восстановления базы данных выводится следующее сообщение:

```
FATAL: requested timeline X is not a child of this server's history
```

Причина: утилита wal-g не может использовать выбранную резервную копию, так как данная копия не является предком текущей базы данных.

Решение: добавьте в файл конфигурации `postgresql.conf` следующую строку:

```
recovery_target_timeline = 'latest'
```

PostgreSQL: отказ в доступе к PGDATA

Проявление: при попытке восстановления базы данных выводится следующее сообщение:

```
permission denied
```

Причина: для директории файлов PostgreSQL (PGDATA) указан неверный владелец.

Решение: переопределите владельца директории файлов PostgreSQL, выполнив команду:

```
chown -R postgres:postgres /<path_to_pgdata>
```

Здесь:

- `<path_to_pgdata>` — путь к директории с данными PostgreSQL, например, `/var/lib/pgsql/16/data`.

ClickHouse: разрыв в цепочке резервных копий

Проявление: при восстановлении базы данных из резервной FULL- или связанной DIFF-копии выводится следующее сообщение:

```
no such file or directory
```

Причина: произошел разрыв цепочки резервных копий: повреждена FULL-копия или связывающая DIFF-копия.

Решение: доступные способы решения проблемы:

- Если повреждена резервная FULL-копия, то восстановите базу данных из предыдущей рабочей FULL-копии.
- Если повреждена связывающая DIFF-копия, то восстановите базу данных вручную последовательно.

Чтобы решить проблему с поврежденной резервной FULL-копией:

1. Проверьте список резервных копий, выполнив команду:

   ```
   clickhouse-backup list remote
   ```

   ▼ *Пример списка резервных копий*

   ```
   full-2026-01-27   remote                          all:14.35GiB
   inc-2026-01-28    remote   +full-2026-01-27       all:2.43GiB
   inc-2026-01-29    remote   +full-2026-01-27       all:2.28GiB
   ```

   В третьем столбце отображается название резервной FULL-копии, с которой связана DIFF-копия. Например, `+full-2026-01-27`.

2. Если резервная FULL-копия отсутствует, то все связанные DIFF-копии становятся недействительными. Необходимо восстановить базу данных из предыдущей резервной FULL-

копии.

Чтобы решить проблему с поврежденной DIFF-копией в цепочке:

1. Очистите локальные резервные копии, выполнив команду:

   ```
   clickhouse-backup delete local --all
   ```

2. Проверьте список резервных копий, выполнив команду:

   ```
   clickhouse-backup list remote
   ```

   ▼ *Пример списка резервных копий*

   ```
   full-2026-01-27   remote                         all:14.35GiB
   inc-2026-01-28    remote   +full-2026-01-27      all:2.43GiB
   inc-2026-01-29    remote   +full-2026-01-27      all:2.28GiB
   ```

   В третьем столбце отображается название резервной FULL-копии, с которой связана DIFF-копия. Например, `+full-2026-01-27`.

3. Восстановите базу данных по порядку из каждой резервной копии, начиная с FULL-копии. Пример пошагового восстановления:

   ```
   clickhouse-backup restore_remote full-2026-01-27T10-00-00
   clickhouse-backup restore_remote inc-2026-01-28T10-00-00
   clickhouse-backup restore_remote inc-2026-01-29T10-00-00
   ```

ClickHouse: ошибка восстановления данных в существующую таблицу

Проявление: при попытке восстановить данные в конкретную таблицу базы данных выводится следующее сообщение:

```
DB::Exception: Table already exists
```

Причина: таблица, в которую восстанавливаются данные, уже существует.

Решение: доступны следующие варианты решения:

- Удалить таблицу вручную.
- Добавить к команде восстановления базы данных флаг `--rm`. Например:

  ```
  clickhouse-backup restore_remote --rm <backup_name>
  ```

  Здесь:

- <backup_name> — название резервной копии.

ClickHouse: ошибка отключения таблицы

Проявление: при попытке восстановления базы данных выводится следующее сообщение:

```
Code: 243. DB::Exception: Cannot detach
```

Причина: таблица восстанавливаемой базы данных используется сторонним приложением.

Решение: остановите все приложения, которые посылают запросы к таблицам восстанавливаемой базы данных.

Восстановление сервисов платформы

В данном разделе описаны шаги для восстановления сервисов платформы MinIO, NATS и Dkron.



Все работы по восстановлению производятся с учетом, что настроены соответствующие секреты и задачи (job) резервного копирования. Подробности приведены в разделе Резервное копирование сервисов платформы.

Восстановление MinIO

Чтобы восстановить MinIO:

1. Подключитесь к кластеру Kubernetes.
2. Задайте необходимые переменные, выполнив команду:

   ```
   export NAMESPACE="<namespace>"
   export IMAGE="harbor.rvision.pro/tools/backup-wrapper:v1.0.0"
   ```

   Здесь:

   - <namespace> — название пространства имен, в котором установлена система.

   Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

3. Запустите под для восстановления MinIO, выполнив команду:

   ```
   kubectl run restore-minio -n "$NAMESPACE" --rm -it \
     --image="$IMAGE" --restart=Never \
     --overrides='{
       "spec": {
         "containers": [{
           "name": "restore",
           "image": "'"$IMAGE"'",
           "command": ["/bin/ash"],
   ```

```
        "stdin": true, "tty": true,
        "envFrom": [{"secretRef": {"name": "backup-minio"}}]
      }]
    }
  }'
```

4. Подключитесь к созданному поду, выполнив команду:

```
kubectl exec -it restore-minio -n "$NAMESPACE" -- /bin/sh
```

5. Настройте сокращенные имена подключения к S3-совместимому хранилищу (aliases), выполнив команду:

```
mc alias set source "$SOURCE_ENDPOINT" "$SOURCE_ACCESS_KEY" "$SOURCE_SECRET_KEY"
mc alias set target "$S3_ENDPOINT" "$S3_ACCESS_KEY" "$S3_SECRET_KEY"
```

6. Проверьте список доступных резервных копий MinIO, выполнив команду

```
mc ls "target/${S3_BUCKET}/backups/minio/"
```



Также вы можете посмотреть объем памяти, который занимают резервные копии MinIO. Для этого выполните команду:

```
mc du "target/${S3_BUCKET}/backups/minio/"
```

7. Выполните следующий скрипт восстановления в командной строке:

```
IFS=,
for bucket in $BUCKETS; do
  echo "Восстановление: $bucket"
  mc mb --ignore-existing "source/${bucket}"
  mc mirror --preserve --overwrite \
    "target/${S3_BUCKET}/backups/minio/${bucket}" \
    "source/${bucket}"
done
```

8. Дождитесь успешного восстановления каждого контейнера (bucket) MinIO.

Восстановление NATS

Чтобы восстановить NATS:

1. Подключитесь к кластеру Kubernetes.
2. Задайте необходимые переменные, выполнив команду:

```
export NAMESPACE="<namespace>"
export IMAGE="harbor.rvision.pro/tools/backup-wrapper:v1.0.0"
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

3. Запустите под для восстановления NATS, выполнив команду:

```
kubectl run restore-nats -n "$NAMESPACE" --rm -it \
  --image="$IMAGE" --restart=Never \
  --overrides='{
    "spec": {
      "containers": [{
        "name": "restore",
        "image": "'"$IMAGE"'",
        "command": ["/bin/ash"],
        "stdin": true, "tty": true,
        "envFrom": [{"secretRef": {"name": "backup-nats"}}]
      }]
    }
  }'
```

4. Подключитесь к созданному поду, выполнив команду:

```
kubectl exec -it restore-nats -n "$NAMESPACE" -- /bin/sh
```

5. Настройте сокращенное имя подключения к S3-совместимому хранилищу (alias), выполнив команду:

```
mc alias set target "$S3_ENDPOINT" "$S3_ACCESS_KEY" "$S3_SECRET_KEY"
```

6. Проверьте список доступных резервных копий NATS, выполнив команду:

```
mc ls "target/${S3_BUCKET}/backups/nats/"
```

7. Скопируйте необходимую резервную копию в локальную директорию `/tmp/`, выполнив команду:

```
mc cp "target/${S3_BUCKET}/backups/nats/<backup_name>" /tmp/
```

Здесь:

- `<backup_name>` — название файла резервной копии, например, `nats-backup.tar`.

8. Создайте директорию для восстановления и распакуйте данные резервной копии, выполнив команды:

```
mkdir -p /tmp/nats-restore
tar -xf /tmp/nats-backup.tar -C /tmp/nats-restore
```

9. Настройте переменные подключения к NATS, выполнив команду:

```
NATS_OPTS="--server=${NATS_URL}"
case "${NATS_AUTH_MODE:-none}" in
  token)    NATS_OPTS="$NATS_OPTS --token=${NATS_TOKEN}" ;;
  userpass) NATS_OPTS="$NATS_OPTS --user=${NATS_USER} --password=${NATS_PASSWORD}"
;;
  creds)    NATS_OPTS="$NATS_OPTS --creds=/etc/nats/nats.creds" ;;
  nkey)     NATS_OPTS="$NATS_OPTS --nkey=/etc/nats/nats.nkey" ;;
esac
```

10. Проверьте подключение к NATS, выполнив команду:

```
nats $NATS_OPTS account info
```

11. Создайте резервную копию нынешнего состояния NATS, выполнив команды:

```
mkdir -p /tmp/pre-restore
for s in $(nats $NATS_OPTS stream ls --json -a 2>/dev/null \
  | tr -d '[]"," | sed '/^$/d; s/^[[:space:]]*//' ); do
  nats $NATS_OPTS stream backup "$s" "/tmp/pre-restore/$s" --no-progress
2>/dev/null || true
done
```

12. Восстановите каждый стрим NATS, выполнив скрипт в командной строке:

```
BACKUP_DIR=$(find /tmp/nats-restore -mindepth 1 -maxdepth 1 -type d | head -1)
for stream_dir in "$BACKUP_DIR"/*/; do
  stream_name="${stream_dir%/}"
  stream_name="${stream_name##*/}"
  # В восстановлении участвуют только каталоги. JSON-метаданные пропускаются.
  [ ! -d "$stream_dir" ] && continue

  echo "Восстановление: $stream_name"
  nats $NATS_OPTS stream rm "$stream_name" --force 2>/dev/null || true
  # Имя стрима берётся из метаданных внутри резервной копии.
  nats $NATS_OPTS stream restore "${stream_dir%/}"
```

```
done
```

13. Проверьте, что все стримы NATS восстановились корректно, с помощью команды:

```
nats $NATS_OPTS stream ls
```

Восстановление Dkron

Чтобы восстановить Dkron:

1. Подключитесь к кластеру Kubernetes.
2. Задайте необходимые переменные, выполнив команду:

```
export NAMESPACE="<namespace>"
export IMAGE="harbor.rvision.pro/tools/backup-wrapper:v1.0.0"
```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

3. Запустите под для восстановления NATS, выполнив команду:

```
kubectl run restore-dkron -n "$NAMESPACE" --rm -it \
  --image="$IMAGE" --restart=Never \
  --overrides='{
    "spec": {
      "containers": [{
        "name": "restore",
        "image": "'"$IMAGE"'",
        "command": ["/bin/ash"],
        "stdin": true, "tty": true,
        "envFrom": [{"secretRef": {"name": "backup-dkron"}}]
      }]
    }
  }'
```

4. Подключитесь к созданному поду, выполнив команду:

```
kubectl exec -it restore-dkron -n "$NAMESPACE" -- /bin/sh
```

5. Настройте имя подключения к S3-совместимому хранилищу (alias), выполнив команду:

```
mc alias set target "$S3_ENDPOINT" "$S3_ACCESS_KEY" "$S3_SECRET_KEY"
```

6. Проверьте список доступных резервных копий, выполнив команду:

```
mc ls "target/${S3_BUCKET}/backups/dkron/"
```

7. Скачайте необходимую резервную копию Dkron в локальную директорию `/tmp/`, выполнив команду:

```
mc cp "target/${S3_BUCKET}/backups/dkron/jobs.json" /tmp/dkron-restore.json
```

8. Проверьте содержимое резервной копии, выполнив команду:

```
grep -c '"name"' /tmp/dkron-restore.json
```

9. Разделите хост и порт из переменной `DKRON_ENDPOINT`, выполнив команду:

```
DKRON_HOST="${DKRON_ENDPOINT%%:*}"
DKRON_PORT="${DKRON_ENDPOINT##*:}"
[ "$DKRON_PORT" = "$DKRON_HOST" ] && DKRON_PORT=8080
```

10. Восстановите Dkron, выполнив скрипт в командной строке:

```
tr -d '\n\r' < /tmp/dkron-restore.json \
  | sed 's/^\[{/{/; s/}\]$/}/' \
  | sed 's/},{/}\n{/g' \
  | while IFS= read -r job; do
    _name=$(echo "$job" | sed -n
's/.*"name"[[:space:]]*:[[:space:]]*"\([^"]*\)".*/\1/p')
    _len=$(printf '%s' "$job" | wc -c)
    echo "Восстановление job: ${_name:-?}"
    {
      printf "POST /v1/jobs HTTP/1.1\r\n"
      printf "Host: %s\r\n" "$DKRON_HOST"
      printf "Content-Type: application/json\r\n"
      printf "Content-Length: %d\r\n" "$_len"
      printf "Connection: close\r\n"
      printf "\r\n"
      printf '%s' "$job"
    } | nc "$DKRON_HOST" "$DKRON_PORT"
    echo ""
done
```

11. Проверьте, что Dkron корректно восстановился, с помощью команды:

```
wget -qO- "http://${DKRON_HOST}:${DKRON_PORT}/v1/jobs" | head -c 500
```

Архивирование и восстановление исторических событий

В данном разделе описана настройка автоматизированного архивирования исторических событий с их последующим удалением из основного хранилища ClickHouse, а также порядок восстановления данных из архивов.



Описываемый механизм предназначен для архивного хранения исторических данных и не является средством полного резервного копирования кластера.

Данная инструкция описывает процесс настройки архивирования событий в экземпляре ClickHouse, расположенном на **одном выделенном сервере**.

Полное резервное копирование базы данных ClickHouse описано в следующих разделах:

- Резервное копирование ClickHouse без Backup Wrapper
- Резервное копирование внешних баз данных
- Резервное копирование внутренних баз данных

Не рекомендуется настраивать резервное копирование исторических событий, если вы настроили резервное копирование по инструкциям выше.

Содержание раздела:

- Настройка утилиты для архивирования и восстановления исторических событий
- Работа с утилитой для архивирования исторических событий

Настройка утилиты для архивирования и восстановления исторических событий



Полное резервное копирование базы данных ClickHouse описано в следующих разделах:

- Резервное копирование ClickHouse без Backup Wrapper
- Резервное копирование внешних баз данных
- Резервное копирование внутренних баз данных

Не рекомендуется настраивать резервное копирование исторических событий, если вы настроили резервное копирование по инструкциям выше.

Для архивирования и восстановления исторических событий используется утилита clickhouse-backup. Чтобы установить и настроить ее в вашей инсталляции ClickHouse, выполните действия, указанные ниже.



Все команды в инструкции необходимо выполнять от имени пользователя root.

1. Откройте в браузере список релизов clickhouse-backup на GitHub.
2. Чтобы выбрать пакет релиза для установки, определите архитектуру и семейство ОС узла, где установлена СУБД ClickHouse.

   a. Выполните команду для вывода архитектуры:

   ```
   uname -m
   ```

   Пример вывода команды `uname -m`

   ```
   x86_64
   ```

   b. Выполните команду для вывода файла информации об ОС:

   ```
   cat /etc/os-release
   ```

   Пример вывода команды `cat /etc/os-release`

   ```
   NAME="RED OS"
   VERSION="MUROM (7.3.2)"
   PLATFORM_ID="platform:el7"
   ID="redos"
   ID_LIKE="rhel centos fedora"
   VERSION_ID="7.3.2"
   PRETTY_NAME="RED OS MUROM (7.3.2)"
   ANSI_COLOR="0;31"
   CPE_NAME="cpe:/o:redos:redos:7"
   HOME_URL="http://red-soft.ru/ru/main_products.html#redos"
   BUG_REPORT_URL="http://redos-support.red-soft.ru"
   EDITION="Standard"
   ```

   Если параметр `ID_LIKE` имеет значение `rhel centos fedora`, вам нужно скачать файл clickhouse-backup с расширением `.rpm`, а если `ubuntu` — с расширением `.deb`.

   Согласно вышеуказанному примеру, для ОС с параметром `ID_LIKE="rhel centos fedora"` и архитектурой `x86_64` нужно скачать пакет релиза clickhouse-backup `clickhouse-backup-2.6.41-1.x86_64.rpm`.

3. Скачайте необходимый пакет релиза и скопируйте его на узел, на котором установлена СУБД ClickHouse.
4. Установите пакет с помощью менеджера пакетов, доступного на узле, например:

Debian

```
dpkg -i <package_name>
```

Здесь:

- <package-name> — имя пакета, например, clickhouse-backup_2.6.41_amd64.deb.

РЕД ОС

```
dnf install -y <package_name>
```

Здесь:

- <package-name> — имя пакета, например, clickhouse-backup-2.6.41-1.x86_64.rpm.

5. Убедитесь, что утилита clickhouse-backup успешно установлена, с помощью следующей команды:

```
clickhouse-backup --version
```

Пример вывода версии clickhouse-backup

```
Version: 2.6.41
Git Commit: b52aa4fc973f0025707e09dce2f275257f068ff1
Build Date: 2025-12-02
```

6. Создайте пользователя ClickHouse, от имени которого будет работать clickhouse-backup. Для этого:
   a. Подключитесь к консольному клиенту ClickHouse, выполнив следующую команду:

   ```
   clickhouse-client
   ```

   b. Создайте пользователя для clickhouse-backup, выполнив следующий запрос:

   ```
   CREATE USER <backup_user> IDENTIFIED WITH plaintext_password BY
   '<backup_password>';
   ```

   Здесь и далее:

   - <backup_user> — имя пользователя для clickhouse-backup.
   - <backup_password> — пароль пользователя для clickhouse-backup.

   c. Выдайте пользователю <backup_user> необходимые права для корректной работы clickhouse-backup, выполнив следующий запрос:

   ```
   GRANT SELECT, INSERT, ALTER, DROP, CREATE TABLE, BACKUP, SHOW ON *.* TO
   <backup_user>;
   GRANT SYSTEM ON *.* TO <backup_user>;
   ```

d. Проверьте, что пользователь <backup_user> обладает необходимыми правами, с помощью следующего запроса:

```
SHOW GRANTS FOR <backup_user>;
```

Результат выполнения запроса должен содержать следующую строку:

```
GRANT SHOW, SELECT, INSERT, ALTER, CREATE TABLE, DROP, BACKUP, SYSTEM ON *.* TO
<backup_user>
```

7. Настройте директорию для хранения архивов исторических событий.



Далее в инструкции используется директория /opt/clickhouse/backup, однако при необходимости вы можете ее изменить.

a. Создайте директорию для архивов исторических событий и выдайте права на работу с ней пользователю и группе clickhouse:

```
mkdir -p /opt/clickhouse/backup
chown clickhouse:clickhouse /opt/clickhouse/backup
chmod 755 /opt/clickhouse/backup
```

b. Для обеспечения доступа ClickHouse к разделу на диске, в котором будут создаваться архивы событий, создайте конфигурационный файл /etc/clickhouse-server/config.d/backup_disk.xml со следующим содержимым:

```
<clickhouse>
    <storage_configuration>
        <disks>
            <backups>
                <type>local</type>
                <path>/opt/clickhouse/backup/</path>
            </backups>
        </disks>
    </storage_configuration>
    <backups>
        <allowed_disk>backups</allowed_disk>
        <allowed_path>/opt/clickhouse/backup/</allowed_path>
    </backups>
</clickhouse>
```

c. Для применения конфигурационного файла перезагрузите службы ClickHouse:

```
systemctl restart clickhouse-*
```

d. Когда службы ClickHouse будут перезапущены, убедитесь, что в логах перезапуска отображается информация о прочтении файла `/etc/clickhouse-server/config.d/backup_disk.xml`. Для этого выполните следующую команду:

```
systemctl status clickhouse-*
```

▼ *Пример лога с информацией о прочтении файла* `/etc/clickhouse-server/config.d/backup_disk.xml`

```
clickhouse-server.service- ClickHouse Server (analytic DBMS for big data)
   Loaded: loaded (/usr/lib/systemd/system/clickhouse-server.service;
enabled; vendor preset: disabled)
   Active: active (running) since Fri 2025-11-28 13:55:06 UTC: 3 days ago
 Main PID: 385936 (clickhouse-serv)
    Tasks: 963 (limit: 14304)
    Memory: 8.6G
    CPU: 2d 49min 11.507s
 CGroup: /system.slice/clickhouse-server.service
         ├─── 385933 clickhouse-watchdog "" ** ** ** ** *"""
--config=/etc/clickhouse-server/config.xml --pid-file=/run/clickhouse
-server/clickhouse-server.pid
         └─── 385936 /usr/bin/clickhouse-server --config=/etc/clickhouse
-server/config.xml --pid-file=/run/clickhouse-server/clickhouse-server.pid

dec 01 13:55:02 clc04.evo.local systemd[1]: Starting ClickHouse Server
(analytic DBMS for big data)...
dec 01 13:55:02 clc04.evo.local clickhouse-server [385933]: Processing
configuration file '/etc/clickhouse-server/config.xml'.
dec 01 13:55:02 clc04.evo.local clickhouse-server [385933]: Merging
configuration file '/etc/clickhouse-server/config.d/backup_disk.xml'.
dec 01 13:55:02 clc04.evo.local clickhouse-server [385933]: Logging warning
to /var/log/clickhouse-server/clickhouse-server.log
dec 01 13:55:02 clc04.evo.local clickhouse-server [385933]: Logging errors to
/var/log/clickhouse-server/clickhouse-server.err.log
dec 01 13:55:02 clc04.evo.local systemd[1]: clickhouse-server.service:
Supervising process 385936 which is not our child. We'll most likely not
notice when it exits.
dec 01 13:55:06 clc04.evo.local systemd[1]: Started ClickHouse Server
(analytic DBMS for big data).
```

8. Создайте конфигурационный файл `/etc/clickhouse-backup/config.yml`, содержащий настройки утилиты clickhouse-backup:

```
general:
    remote_storage: none
    backups_to_keep_local: <backups_to_keep_local>
    log_level: info
clickhouse:
    username: <backup_user>
```

```
    password: <backup_password>
    host: localhost
    port: 9000
    disk_mapping:
      backups: /opt/clickhouse/backup/
    skip_tables:
        - system.*
        - INFORMATION_SCHEMA.*
        - information_schema.*
    timeout: <timeout>
    use_embedded_backup_restore: true
    embedded_backup_disk: "backups"
    debug: false
```

Здесь:

- <backups_to_keep_local> — максимальное количество архивов событий, хранимых на диске. Целое число. Следует настроить его в зависимости от размера доступного дискового пространства.
- <backup_user> — имя пользователя для clickhouse-backup.
- <backup_password> — пароль пользователя для clickhouse-backup.
- <timeout> — максимальное допустимое время на создание архива, после чего процесс архивирования исторических событий будет досрочно остановлен. Указывается в часах, например, 24h.

9. Создайте скрипт архивирования исторических событий /usr/local/bin/archive_events.sh:

```
#!/bin/bash

# archive_events.sh - архивация партиций таблиц default.EventStorage_*_local старше
N дней и удаление их из ClickHouse.
#
# Использование: ./archive_events.sh [DAYS].
#   DAYS - события старше скольких дней нужно архивировать (по умолчанию 7).
#
# Требования: bash, clickhouse-client, clickhouse-backup.

set -euo pipefail

# Перенаправление stdout и stderr в лог-файл.
LOG_DIR=/opt/clickhouse/log
LOG_FILE="$LOG_DIR/archive_events.log"

mkdir -p "$LOG_DIR"
chown clickhouse:clickhouse "$LOG_DIR"
chmod 0755 "$LOG_DIR"
touch "$LOG_FILE"
chown clickhouse:clickhouse "$LOG_FILE"
chmod 0644 "$LOG_FILE"
```

```
exec >>"$LOG_FILE" 2>&1

# Пути к утилитам clickhouse-backup и clickhouse-client.
CHB_BIN=/usr/bin/clickhouse-backup
CH_CLIENT=/usr/bin/clickhouse-client

# Вызов справки по ключу --help.
if [[ "${1:-}" =~ ^(-h|--help)$ ]]; then
  cat <<'EOF'
Использование: ./archive_events.sh [DAYS].
  DAYS - события старше скольких дней нужно архивировать (по умолчанию 7).
EOF
  exit 0
fi

RETENTION_DAYS="${1:-7}"

# Паттерн таблиц и путь к конфигурационному файлу.
TABLE_PATTERN='default.EventStorage_*_local'
CHB_CFG='/etc/clickhouse-backup/config.yml'

# Проверка наличия утилиты clickhouse-backup.
if ! command -v "$CHB_BIN" &>/dev/null; then
  echo "ERROR: clickhouse-backup не найден по $CHB_BIN" >&2
  exit 1
fi

# Проверка наличия утилиты clickhouse-client.
if ! command -v "$CH_CLIENT" &>/dev/null; then
  echo "ERROR: clickhouse-client не найден по $CH_CLIENT" >&2
  exit 1
fi

# Чтение учетных данных из конфигурационного файла.
read -r CHB_USER CHB_PASS CHB_HOST CHB_PORT < <(
  awk '/^\s*username:/ {u=$2}
       /^\s*password:/ {p=$2}
       /^\s*host:/     {h=$2}
       /^\s*port:/     {po=$2}
       END {
         print u, p, h, po
       }' "$CHB_CFG"
)

# Проверка, что учетные данные успешно получены.
if [[ -z "$CHB_USER" || -z "$CHB_PASS" || -z "$CHB_HOST" || -z "$CHB_PORT" ]]; then
  echo "ERROR: не удалось прочитать username/password/host/port из $CHB_CFG" >&2
  exit 1
fi
```

```
# Проверка наличия и выбор кластера ClickHouse. По умолчанию - default.
CLUSTER=$("$CH_CLIENT" \
  --host="$CHB_HOST" --port="$CHB_PORT" \
  --user="$CHB_USER" --password="$CHB_PASS" \
  --query="SELECT any(cluster) FROM system.clusters FORMAT TSV")
CLUSTER="${CLUSTER:-default}"

# Вычисление пороговой даты.
CUTOFF=$(date -d "-${RETENTION_DAYS} days" +%Y%m%d)

# Сбор списка партиций.
mapfile -t PARTS < <(
  "$CH_CLIENT" \
    --host="$CHB_HOST" --port="$CHB_PORT" \
    --user="$CHB_USER" --password="$CHB_PASS" \
    --query="
      SELECT DISTINCT partition
      FROM system.parts
      WHERE active
        AND database='default'
        AND table LIKE 'EventStorage_%_local'
        AND partition <= '$CUTOFF'
      ORDER BY partition
      FORMAT TSV"
)

if [ "${#PARTS[@]}" -eq 0 ]; then
  echo "Нет партиций ≤ $CUTOFF для архивации, процесс отменен."
  exit 0
fi

# Формирование CSV-файла и имени архива с перечислением партиций.
{
  IFS=','
  PARTS_CSV="${PARTS[*]}"
}

PARTS_UNDERSCORED="${PARTS_CSV//,/_}"
BACKUP_NAME="archive_events_${PARTS_UNDERSCORED}"

echo "→ Архивация партиций [$PARTS_CSV] в архив '$BACKUP_NAME'…"
"$CHB_BIN" create \
  --config "$CHB_CFG" \
  --tables="$TABLE_PATTERN" \
  --partitions="$PARTS_CSV" \
  "$BACKUP_NAME"

PARTS_SQL=$(printf "'%s'," "${PARTS[@]}")
PARTS_SQL=${PARTS_SQL%,}

echo "→ Удаление партиций из ClickHouse (кластер '$CLUSTER')…"
```

```
# Сбор списка таблиц, из которых надо удалить партиции.
mapfile -t TABLES < <(
  "$CH_CLIENT" \
    --host="$CHB_HOST" --port="$CHB_PORT" \
    --user="$CHB_USER" --password="$CHB_PASS" \
    --query="
      SELECT DISTINCT concat(database, '.', table)
      FROM system.parts
      WHERE active
        AND database='default'
        AND table LIKE 'EventStorage_%_local'
        AND partition IN (${PARTS_SQL})
      FORMAT TSV"
)

for tbl in "${TABLES[@]}"; do
  for part in "${PARTS[@]}"; do
    echo "  • ALTER TABLE $tbl ON CLUSTER '$CLUSTER' DROP PARTITION '$part'"
    "$CH_CLIENT" \
      --host="$CHB_HOST" --port="$CHB_PORT" \
      --user="$CHB_USER" --password="$CHB_PASS" \
      --query="ALTER TABLE $tbl ON CLUSTER '$CLUSTER' DROP PARTITION '$part'"
  done
done

echo "Готово. Архив: $BACKUP_NAME, удалены партиции ≤ $CUTOFF."
```

10. Сделайте скрипт исполняемым и выдайте на него права пользователю и группе clickhouse:

```
chown clickhouse:clickhouse /usr/local/bin/archive_events.sh
chmod 750 /usr/local/bin/archive_events.sh
```

11. Перейдите к разделу Работа с утилитой для архивирования исторических событий.

Работа с утилитой для архивирования исторических событий

Данный раздел описывает действия по архивированию и восстановлению исторических событий:

- Просмотр информации о партициях и дисках ClickHouse
- Настройка расписания архивирования исторических событий
  - Примеры настройки расписания архивирования исторических событий
- Просмотр журнала архивирования исторических событий
- Восстановление событий из архива

Просмотр информации о партициях и дисках ClickHouse

Для оценки дискового пространства, занимаемого данными о событиях в ClickHouse, а также размера будущих архивов исторических событий вы можете выполнить запросы, которые отобразят

информацию о партициях и дисках ClickHouse. Для этого:

1. Подключитесь к терминалу узла, на котором установлена СУБД ClickHouse.
2. Подключитесь к консольному клиенту ClickHouse, выполнив следующую команду:

```
clickhouse-client
```

3. Выполните один из следующих запросов:
   - Просмотр информации о партициях ClickHouse:

```
SELECT
    `table`,
    partition,
    formatReadableSize(sum(bytes_on_disk)) AS size
FROM system.parts
WHERE (database = 'default') AND (`table` LIKE 'EventStorage_%_local')
GROUP BY
    `table`,
    partition
ORDER BY
    `table` ASC,
    partition ASC
```

▼ *Пример вывода информации о партициях*

```
Query id: 0725d2cb-6121-4b6c-a979-11aca297a73f


 ┌─table─────────────────────────────────────────────┬─partition─┬─size──────┐
 1. │ EventStorage_088b6376_bc2c_4ac7_9f3e_099f67876ec1_local │ 20250605
 │ 16.45 KiB  │
 2. │ EventStorage_088b6376_bc2c_4ac7_9f3e_099f67876ec1_local │ 20250620
 │ 16.21 KiB  │
 3. │ EventStorage_0e87bb0d_5784_4765_9aad_bac8e6c9ec52_local │ 20240722
 │ 72.01 KiB  │
 4. │ EventStorage_0e87bb0d_5784_4765_9aad_bac8e6c9ec52_local │ 20240723
 │ 137.46 KiB │
 5. │ EventStorage_0e87bb0d_5784_4765_9aad_bac8e6c9ec52_local │ 20240724
 │ 230.35 KiB │
 6. │ EventStorage_0e87bb0d_5784_4765_9aad_bac8e6c9ec52_local │ 20240725
 │ 55.72 KiB  │
 7. │ EventStorage_0e87bb0d_5784_4765_9aad_bac8e6c9ec52_local │ 20240802
 │ 4.95 MiB   │
 8. │ EventStorage_0e87bb0d_5784_4765_9aad_bac8e6c9ec52_local │ 20240803
 │ 10.55 MiB  │
 9. │ EventStorage_0e87bb0d_5784_4765_9aad_bac8e6c9ec52_local │ 20240804
```

```
 │ 10.56 MiB   │
10. │ EventStorage_0e87bb0d_5784_4765_9aad_bac8e6c9ec52_local │ 20240805
 │ 10.57 MiB   │

 └──────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────┴──────────────────────────
─┴─────────────────────────┘

10 rows in set. Elapsed: 0.132 sec. Processed 2.08 thousand rows, 218.09 KB
(15.73 thousand rows/s., 1.65 MB/s.)
```

◦ Просмотр информации о томах ClickHouse:

```
SELECT * FROM system.disks
```

▼ *Пример вывода информации о томах ClickHouse*

```
Query id: b0c7375a-ca73-4d06-93c3-313940468377


 ┌─name──────────────┬─path──────────────────────────────────────────
────┬────free_space────┬─────total_space──┬─unreserved_space──┬─keep_f
ree_space──┬─type────┬─object_storage_type─┬─metadata_type─┬─is_e
ncrypted─┬─is_read_only─┬─is_write_once─┬─is_remote─┬─is_broken─
──┬─cache_path─┐
1. │ cold_disk_1 │ /opt/clickhouse/cold-1/   │ 133619982336 │ 738952110080
 │     133619982336 │               0 │ Local │ None                │
None          │            0 │            0 │             0 │         0
 │         0 │             │
2. │ default     │ /opt/clickhouse/          │ 133619982336 │ 738952110080
 │     133619982336 │               0 │ Local │ None                │
None          │            0 │            0 │             0 │         0
 │         0 │             │
3. │ hot_disk_1  │ /opt/clickhouse/hot-1/    │ 133619982336 │ 738952110080
 │     133619982336 │               0 │ Local │ None                │
None          │            0 │            0 │             0 │         0
 │         0 │             │
4. │ backups     │ /opt/clickhouse/backups/  │ 133619982336 │ 738952110080
 │     133619982336 │               0 │ Local │ None                │
None          │            0 │            0 │             0 │         0
 │         0 │             │

 └─────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────
──
```

```
4 rows in set. Elapsed: 0.003 sec.
```

Настройка расписания архивирования исторических событий

Чтобы настроить расписание архивирования исторических событий, выполните следующие действия:

1. Настройте периоды горячего и холодного хранения событий (hot_storage и cold_storage) в разделе **Ресурсы → Хранилища событий** веб-интерфейса системы, расширив период холодного хранения на дополнительное количество дней, за которое необходимо архивировать события.
2. Настройте запуск утилиты clickhouse-backup по расписанию. Для этого выполните следующие действия на узле, на котором установлена СУБД ClickHouse:
   a. Откройте файл расписания crontab. Например, это можно сделать с помощью текстового редактора nano, выполнив команду:

   ```
   EDITOR=nano crontab -e
   ```

   b. Добавьте в расписание строку для запуска clickhouse-backup следующего вида:

   ```
   <cron_expression> sudo -u clickhouse /usr/local/bin/archive_events.sh <n_days>
   ```

   Здесь:

   - `<cron_expression>` — cron-выражение, определяющее расписание запусков clickhouse-backup.

     Чтобы составить cron-выражение, можно воспользоваться генератором, например, crontab.guru.

   - `<n_days>` — события старше скольких дней нужно архивировать.

   c. Сохраните изменения в файле расписания.

Примеры настройки расписания архивирования исторических событий

- Пусть в системе настроено горячее хранение событий в течение 1 дня, а холодное — в течение 2 дней. Необходимо ежедневно выполнять архивацию событий старше 2 дней.

  Тогда строка для запуска clickhouse-backup может выглядеть следующим образом:

  ```
  0 2 * * * sudo -u clickhouse /usr/local/bin/archive_events.sh 2
  ```

  Утилита clickhouse-backup будет запускаться ежедневно в 02:00, выполнять архивацию событий старше 2 дней (за третий день) и удалять их из холодного хранилища ClickHouse. События за 2 последних дня останутся в ClickHouse.

- Пусть в системе настроено горячее хранение событий в течение 30 дней, а холодное — в течение 187 дней. Необходимо еженедельно выполнять архивацию событий за самую раннюю неделю в холодном хранилище, то есть событий старше 210 дней.

  Тогда строка для запуска clickhouse-backup может выглядеть следующим образом:

  ```
  0 2 * * 0 sudo -u clickhouse /usr/local/bin/archive_events.sh 210
  ```

  Утилита clickhouse-backup будет запускаться по воскресеньям в 02:00, выполнять архивацию событий старше 210 дней (211—217 дни) и удалять их из холодного хранилища ClickHouse. События за 210 последних дней останутся в ClickHouse.

Просмотр журнала архивирования исторических событий

Чтобы просмотреть журнал архивирования исторических событий:

1. Подключитесь к терминалу ClickHouse.
2. Выполните следующую команду:

   ```
   tail -f -n <n_lines> /opt/clickhouse/log/archive_events.log
   ```

   Здесь:

   - `<n_lines>` — количество последних строк журнала, которые необходимо отобразить.

Восстановление событий из архива

Чтобы восстановить события из архива, выполните следующие действия на узле, на котором установлена СУБД ClickHouse:

1. Получите список имеющихся архивов событий, выполнив следующую команду:

   ```
   clickhouse-backup list
   ```

2. Выполните команду для восстановления событий из необходимого вам архива:

   ```
   clickhouse-backup restore <archive_events_name> --data
   ```

   Здесь:

   - `<archive_events_name>` — имя архива событий, например, `archive_events_20251201`.

3. При необходимости удалите архив, из которого восстановлены события, выполнив следующую команду:

   ```
   clickhouse-backup delete local <archive_events_name>
   ```

Здесь:

- `<archive_events_name>` — имя архива событий, например, `archive_events_20251201`.

Управление лицензией

Данный раздел описывает процесс работы с лицензией в системе. Работа осуществляется в разделе **Настройки** → **Лицензия** веб-интерфейса системы.



Раздел доступен только учетным записям со статусом **Суперадминистратор**.

О лицензии

Лицензия предоставляет доступ к функциям системы, а также определяет ресурсные ограничения и срок использования системы.

Система непрерывно отслеживает параметры лицензии и отображает уведомления при возникновении следующих ситуаций:

- Срок действия лицензии истекает в ближайшие 30 дней.
- Срок действия лицензии истек. Требуется активация нового ключа лицензии.
- Превышены ограничения хотя бы по одному параметру лицензии.

Соответствующие предупреждения также отображаются при просмотре сведений о лицензии.

Интерфейс раздела

Панель инструментов раздела включает кнопку **Активировать лицензию**, которая позволяет задать ключ лицензии в системе.

При наличии активированной лицензии в рабочей области раздела отображаются ее подробные сведения.

Работа с лицензией

Доступные операции над лицензией:

- Просмотр лицензии
- Активация лицензии

Просмотр лицензии

Чтобы просмотреть сведения о лицензии в системе, перейдите в раздел **Настройки** → **Лицензия**.

Если активированная лицензия отсутствует, в верхней части экрана отображается соответствующее предупреждение.

При наличии активированной лицензии в рабочей области раздела отображаются следующие сведения:

- **Статус** — текущий статус лицензии:
  - **Активна** — лицензия активна, и ее срок действия не истек.
  - **Истекает** — лицензия активна, и ее срок действия истекает в ближайшие 30 дней.

    При данном статусе в верхней части сведений о лицензии отображается предупреждение о скором истечении срока лицензии. Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.

  - **Истекла** — срок действия лицензии истек.

    При данном статусе в верхней части экрана отображается предупреждение об истечении срока лицензии и необходимости ее повторной активации. Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.

  - **Превышение параметров** — превышено хотя бы одно ограничение лицензии: **EPS в точках входа** или **Входной EPS коррелятров**.

    При данном статусе в верхней части экрана отображается предупреждение о превышении параметров лицензии. Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.

- **Server ID** — уникальный идентификатор сервера, для которого выдается лицензия.

  Справа от значения поля расположена кнопка , позволяющая скопировать идентификатор сервера в буфер обмена.

- **Срок действия лицензии** — даты начала и окончания действия лицензии.
- **Срок действия технической поддержки** — даты начала и окончания срока, в течение которого техническая поддержка R-Vision может оказывать помощь пользователям системы с текущей лицензией.
- **EPS в точках входа** — ограничение по среднему количеству событий в секунду (EPS), которое может обрабатывать система в точках входа коллекторов с текущей лицензией.
  - Если показатель **EPS в точках входа** не ограничен лицензией, отображается надпись **Неограниченно**.
  - Если система использует от 75% до 100% доступных ресурсов, показатель **EPS в точках входа** выделяется желтым цветом.
  - Если система использует более 100% доступных ресурсов, показатель **EPS в точках входа** выделяется красным цветом, а в верхней части экрана отображается предупреждение о превышении параметров лицензии.

    Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.

- **Входной EPS коррелляторов** — ограничение по среднему количеству событий в секунду (EPS), которое может поступать на входы коррелляторов с текущей лицензией.

  - Если показатель **Входной EPS коррелляторов** не ограничен лицензией, отображается надпись **Неограниченно**.
  - Если система использует от 75% до 100% доступных ресурсов, показатель **Входной EPS коррелляторов** выделяется желтым цветом.
  - Если система использует более 100% доступных ресурсов, показатель **Входной EPS коррелляторов** выделяется красным цветом, а в верхней части экрана отображается предупреждение о превышении параметров лицензии.

    Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.



Если лицензия не совпадает с версией продукта, то в верхней части экрана отображается соответствующее предупреждение.

Особенности расчета EPS для лицензии

Расчет показателей EPS для лицензии осуществляется по следующему алгоритму:

1. Система определяет временной диапазон от текущего момента времени минус 7 дней. Данный диапазон разбивается на интервалы по 4 часа.
2. Для каждого интервала высчитывается сумма событий по всем точкам входа или коррелляторам в системе, в зависимости от вычисляемой метрики.
3. Суммируется общее количество событий по всем ненулевым интервалам. Полученная сумма делится на количество ненулевых интервалов.
4. Полученное среднее количество событий в интервале делится на количество секунд в интервале. В результате получается среднее количество событий в секунду (EPS).

Активация лицензии

Чтобы активировать лицензию:

1. Перейдите в раздел **Настройки** → **Лицензия**. Система отобразит сведения о текущей лицензии в системе.
2. Нажмите на кнопку **Активировать лицензию**. Откроется окно активации лицензии.
3. Нажмите на кнопку **Выбрать файл** (⊕) и выберите файл ключа лицензии в формате `.lic` на своем компьютере.

   При необходимости вы можете удалить загруженный файл с помощью кнопки 🗑.
4. Нажмите на кнопку **Активировать**. Система активирует лицензию и отобразит соответствующее уведомление.

Управление пространствами

Система включает в себя решение для управления кластерами продукта, а также оркестрации запускающихся на них компонентов и организации взаимодействия между ними.

Кластеры и пространства

Развертывание компонентов продукта в одном Kubernetes-кластере не всегда удовлетворяет бизнес-требованиям по работе с удаленным ресурсами, особенно если ресурсы расположены в DMZ-зоне. В таких случаях возникает необходимость развертывать компоненты в отдельных кластерах, которые могут быть размещены в разных сегментах сети. Система позволяет использовать такую топологию и управлять сервисами, которые запущены в этих кластерах.

Кластер Kubernetes, в котором разворачивается основной экземпляр системы, называется **центральным кластером**. Часть компонентов продукта может быть развернута в дополнительных удаленных кластерах — **кластерах сателлитов**. В отличие от основного экземпляра системы, сателлиты не обладают полным спектром функциональности продукта и могут работать только с теми компонентами, которые в них запущены.

Для подключения к различным кластерам продукта, а также развертывания и запуска его компонентов в этих кластерах используется специальная логическая среда — **пространство**. По умолчанию в системе всегда имеется **центральное пространство**, предназначенное для работы с компонентами в центральном кластере.

Программные компоненты, разворачиваемые в пространствах, называются **сервисами**. Каждый сервис поставляется в виде Docker-образов при установке продукта и представляет собой набор контейнеров, приложений и их конфигураций.

В системе доступны следующие типы сервисов:

- **Коллектор** — сервис, который осуществляет сбор событий из источников, их обработку и дальнейшую отправку на хранение или во внешние системы.
- **Шлюз** — сервис, позволяющий управлять потоками данных между агентами на конечных устройствах и центральным кластером.

Таким образом, система предлагает функционал для управления пространствами в различных кластерах продукта и оркестрации его сервисов.

Работа с пространствами и сервисами осуществляется в разделе **Настройки → Пространства и сервисы** веб-интерфейса системы.

Конфигурация сервиса

- **Инфраструктурная конфигурация (Infra)** — содержит сведения о компонентах, которые должны быть развернуты в кластере. Infra-конфигурация не хранит в себе конфигурацию самих приложений, а только описывает необходимую для их запуска инфраструктуру: например, список контейнеров и переменные окружения.
- **Конфигурация приложений (Runtime)** — содержит сведения о поведении запускаемых приложений. Runtime-конфигурация описывает параметры, специфические для конкретного типа сервиса. Такие конфигурации создаются и предоставляются менеджерами конкретного типа

сервисов. Runtime-конфигурация не является обязательной для работы сервиса и может отсутствовать.

Компоненты модуля управления пространствами

Модуль управления пространствами состоит из следующих компонентов:

1. **Менеджер пространств** — предназначен для создания пространств, подключения к ним и управления ими как объектами системы.
2. **Контроллер пространства** — разворачивается в каждом пространстве и предназначен для контроля за ресурсами пространства и за состоянием его сервисов:
   - **Контроль за ресурсами**: при попытке запуска сервиса контроллер определяет, достаточно ли в пространстве системных ресурсов для его развертывания. При достаточном объеме ресурсов контроллер запустит сервис.
   - **Контроль за сервисами**: контроллер регулярно отслеживает состояние сервисов, собирает логи об их изменении и следит, чтобы все запущенные сервисы работали. Так, например, если какой-нибудь сателлит будет перезапущен, контроллер заново развернет все необходимые сервисы в этом сателлите.
3. **Менеджер сервисов** — представляет собой конкретную реализацию менеджера определенного типа сервисов. Отвечает за формирование Runtime-конфигурации нужного типа сервиса и взаимодействие этого сервиса с системой.
4. **Шина** — предназначена для взаимодействия компонентов системы между собой посредством обмена командами и их результатами.

Работа с пространствами

Данный раздел описывает процесс работы с пространствами в системе. Работа осуществляется в разделе **Настройки → Пространства и сервисы → Пространства** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О пространствах

Пространство — это логическая среда в рамках кластера Kubernetes, предназначенная для развертывания и запуска сервисов в этом кластере.

Пространства могут быть созданы как в **центральном кластере**, в котором развернуты все основные компоненты продукта, так и в **сателлитах** — удаленных кластерах, на которых развернута лишь часть компонентов.

В системе доступны пространства следующих типов:

- **Системные пространства** — создаются системой автоматически. Их нельзя изменять или удалять.
- **Пользовательские пространства** — создаются вручную.

Существует системное **центральное пространство**, предназначенное для работы с компонентами в центральном кластере.

Компоненты для работы с пространствами

Для работы с пространствами используются следующие компоненты:

- **Менеджер пространств** — предназначен для создания пространств, подключения к ним и управления ими как объектами системы.
- **Контроллер пространства** — разворачивается в каждом пространстве и предназначен для контроля за ресурсами пространства и за состоянием его сервисов:
  - **Контроль за ресурсами**: при попытке запуска сервиса контроллер определяет, достаточно ли в пространстве системных ресурсов для его развертывания. При достаточном объеме ресурсов контроллер запустит сервис.
  - **Контроль за сервисами**: контроллер регулярно отслеживает состояние сервисов, собирает логи об их изменении и следит, чтобы все запущенные сервисы работали. Так, например, если какой-нибудь сателлит будет перезапущен, контроллер заново развернет все необходимые сервисы на этом сателлите.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать подключение к новому пространству.
- Поле **Поиск** предназначено для быстрого поиска подключений к пространствам в отображаемом списке по названию, IP-адресу или домену.

В рабочей области отображается таблица имеющихся подключений к пространствам. Таблица содержит следующие столбцы:

- **Название** — название подключения к пространству, используемое для его идентификации в системе.
- **Статус** — текущий статус подключения к пространству: **Активный**, **Ошибка** или **Нет соединения**.
- **IP/Домен** — IP-адрес или домен пространства.
- **Тип** — тип подключения к пространству:
  - **Системное** — подключение к центральному пространству;
  - **Пользовательское** — подключение, добавленное вручную.

При работе с таблицей подключений к пространствам доступны следующие операции:

- Поиск подключений по полям **Название** и **IP/Домен**.
- Сортировка подключений по полям **Название** и **IP/Домен**.

При выборе конкретного подключения к пространству в правой части рабочей области отображается его карточка с детальной информацией.

Работа с подключением к пространству

Доступные операции над подключениями к пространствам:

- Создание подключения к пространству
- Просмотр подключения к пространству
- Изменение подключения к пространству
- Удаление подключения к пространству

Создание подключения к пространству

Чтобы создать подключение к пространству:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания подключения.
3. Введите уникальный идентификатор подключения.
4. Введите название подключения.
5. Введите IP-адрес или домен, по которому расположено пространство.



Домен сателлита должен быть уникальным в рамках системы. Создание нескольких подключений к одному пространству с одинаковым доменом недоступно.

6. Введите порт для подключения к пространству. Допустимые значения: 1—65535.
7. При необходимости введите описание подключения к пространству.
8. Нажмите на кнопку **Создать**. Система создаст подключение к пространству и отобразит соответствующее уведомление. Подключение отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Пространства**.

Просмотр подключения к пространству

Чтобы просмотреть подключение к пространству:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.

В верхней части карточки отображается название подключения к пространству.

Карточка подключения содержит сведения, сгруппированные по следующим секциям:

- **Статистика** — метрики ресурсов, занимаемых пространством:
  - **CPU** — количество ядер и уровень загруженности процессора в пространстве.
  - **RAM** — объем и уровень загруженности оперативной памяти в пространстве.

- **Информация** — основная информация о подключении к пространству:
  - **ID** — уникальный идентификационный код подключения, генерируемый автоматически при его создании.
  - **Название** — имя подключения, используемое для его идентификации в системе.
  - **IP-адрес/домен** — IP-адрес или домен кластера, в котором расположено пространство.
  - **Статус** — текущий статус подключения: **Активный**, **Ошибка** или **Нет соединения**.
  - **Порт** — порт кластера, в котором расположено пространство.
  - **Тенант** — название тенанта, к которому относится подключение.
  - **Описание** — краткое описание функциональности пространства.
  - **Дата создания** — дата и время создания подключения.
  - **Создал** — пользователь, инициировавший создание подключения.
  - **Дата изменения** — дата и время последнего изменения подключения.
  - **Изменил** — пользователь, выполнивший последнее изменение подключения.
- **Сервисы** — список сервисов в пространстве. Список представлен в виде таблицы со следующими столбцами:
  - **Активен** — переключатель состояния сервиса.
  - **Название** — название сервиса.
  - **CPU** — уровень загруженности процессора сервиса. При наведении курсора на данный столбец отображаются полные сведения о количестве ядер и уровне загруженности процессора для текущего сервиса.
  - **RAM** — уровень загруженности оперативной памяти сервиса. При наведении курсора на данный столбец отображаются полные сведения об объеме и уровне загруженности оперативной памяти для текущего сервиса.

  В правой части строки сервиса в таблице имеется кнопка →, при нажатии на которую осуществляется переход в раздел **Настройки → Пространства и сервисы → Сервисы** и открывается карточка выбранного сервиса.

Изменение подключения к пространству

Вы можете изменить подключение к пространству через кнопку в его карточке.



Изменение доступно только для пользовательских подключений к пространствам без запущенных сервисов.

Удаление подключения к пространству

Вы можете удалить подключение к пространству через кнопку в его карточке.



Удаление недоступно для центрального пространства.



- При удалении подключения осуществляется только разрыв соединения с пространством, само пространство физически не удаляется и впоследствии

может быть использовано повторно для создания нового подключения.

- Если в пространстве имеются запущенные сервисы, то при удалении подключения эти сервисы станут недоступны для работы или удаления. Чтобы восстановить доступ к сервисам, необходимо заново создать подключение к их пространству.

Работа с сервисами

Данный раздел описывает процесс работы с сервисами в системе. Работа осуществляется в разделе **Настройки** → **Пространства и сервисы** → **Сервисы** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О сервисах

Сервис — это программный компонент, который может разворачиваться в продукте. Сервис создается и запускается в рамках пространства — специальной логической среды, осуществляющей подключение к какому-либо кластеру продукта.

Каждый сервис поставляется в виде Docker-образов при установке продукта и представляет собой набор контейнеров, приложений и их конфигураций.

В системе доступны следующие типы сервисов:

- **Коллектор** — сервис, который осуществляет сбор событий из источников, их обработку и дальнейшую отправку на хранение или во внешние системы.
- **Шлюз** — сервис, позволяющий управлять потоками данных между агентами на конечных устройствах и центральным кластером.

Конфигурация сервиса:

- **Инфраструктурная конфигурация (Infra)** — содержит сведения о компонентах, которые должны быть развернуты в кластере. Infra-конфигурация не хранит в себе конфигурацию самих приложений, а только описывает необходимую для их запуска инфраструктуру: например, список контейнеров и переменные окружения.
- **Конфигурация приложений (Runtime)** — содержит сведения о поведении запускаемых приложений. Runtime-конфигурация описывает параметры, специфические для конкретного типа сервиса. Такие конфигурации создаются и предоставляются менеджерами конкретного типа сервисов. Runtime-конфигурация не является обязательной для работы сервиса и может отсутствовать.

Компоненты для работы с сервисами

Для работы с сервисами используются следующие компоненты:

- **Контроллер пространства** — разворачивается в каждом пространстве и предназначен для контроля за ресурсами пространства и за состоянием его сервисов:

- **Контроль за ресурсами**: при попытке запуска сервиса контроллер определяет, достаточно ли в пространстве системных ресурсов для его развертывания. При достаточном объеме ресурсов контроллер запустит сервис.
- **Контроль за сервисами**: контроллер регулярно отслеживает состояние сервисов, собирает логи об их изменении и следит, чтобы все запущенные сервисы работали. Так, например, если какой-нибудь сателлит будет перезапущен, контроллер заново развернет все необходимые сервисы на этом сателлите.

- **Менеджер сервисов** — представляет собой конкретную реализацию менеджера определенного типа сервисов. Отвечает за формирование Runtime-конфигурации нужного типа сервиса и взаимодействие этого сервиса с системой.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Поле **Поиск** предназначено для быстрого поиска сервисов в отображаемом списке по названию или пространству.
- Выпадающий список **Тип** позволяет настроить отображение списка сервисов в соответствии с их выбранными типами.

В рабочей области отображается таблица имеющихся сервисов, созданных в соответствующих разделах системы. Таблица содержит следующие столбцы:

- **Активен** — переключатель статуса сервиса.



Переключатель доступен, если сервис находится в одном из следующих статусов:

- **Запущен**;
- **Остановлен**;
- **Ошибка**;
- **Неизвестная ошибка**.

- **Название** — название сервиса, используемое для его идентификации в системе.
- **Тип** — тип сервиса.
- **Пространство** — название подключения к пространству, в котором создан сервис.
- **Статус** — текущий статус сервиса:
  - **Неустановленный** — сервис создан и еще не запускался.
  - **Ожидание** — сервис находится в очереди на запуск.
  - **Запускается** — сервис находится в процессе запуска.
  - **Запущен** — сервис запущен и находится в активном состоянии.
  - **Останавливается** — сервис находится в процессе остановки.
  - **Остановлен** — сервис остановлен и недоступен для работы.

- **Ошибка** — ошибка при запуске сервиса.
- **Неизвестная ошибка** — неизвестная ошибка при работе сервиса.

При работе с таблицей сервисов доступны следующие операции:

- Поиск сервисов по полям **Название** и **Пространство**.
- Фильтрация сервисов по полю **Тип**. Тип выбирается из выпадающего списка. Доступен множественный выбор.
- Сортировка сервисов по полям **Название** и **Пространство**.

При выборе конкретного сервиса в правой части рабочей области отображается его карточка с детальной информацией.

Работа с сервисом

Доступные операции над сервисами:

- Просмотр сервиса
- Запуск сервиса
- Остановка сервиса

Просмотр сервиса

Чтобы просмотреть сервис:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.



Вы также можете открыть карточку сервиса из карточки подключения к пространству:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Нажмите на кнопку → в правой части строки нужного сервиса в списке сервисов. Система осуществит переход в раздел **Настройки** → **Пространства и сервисы** → **Сервисы** и откроет карточку выбранного сервиса.

В верхней части карточки отображается название сервиса.

В карточке расположен переключатель состояния сервиса.



Переключатель доступен, если сервис находится в одном из следующих статусов:

- **Запущен**;

- **Остановлен**;
- **Ошибка**;
- **Неизвестная ошибка**.

Карточка содержит основные сведения о сервисе:

- Метрики сервиса:
  - **CPU** — количество ядер и уровень загруженности процессора в сервисе.
  - **RAM** — объем и уровень загруженности оперативной памяти в сервисе.
- Информация о сервисе:
  - **ID** — уникальный идентификационный код сервиса, генерируемый автоматически при его создании.
  - **Название** — имя сервиса, используемое для его идентификации в системе.
  - **Тип** — тип сервиса.
  - **Статус** — текущий статус сервиса:
    - **Неустановленный** — сервис создан и еще не запускался.
    - **Ожидание** — сервис находится в очереди на запуск.
    - **Запускается** — сервис находится в процессе запуска.
    - **Запущен** — сервис запущен и находится в активном состоянии.
    - **Останавливается** — сервис находится в процессе остановки.
    - **Остановлен** — сервис остановлен и недоступен для работы.
    - **Ошибка** — ошибка при запуске сервиса.
    - **Неизвестная ошибка** — неизвестная ошибка при работе сервиса.

    В строке поля также отображается кнопка  , по которой можно инициировать обновление статуса.
  - **Пространство** — название подключения к пространству, в котором создан сервис.
  - **Тенант** — название тенанта, к которому относится сервис.
  - **Дата создания** — дата и время создания сервиса.
  - **Создал** — пользователь, инициировавший создание сервиса.
  - **Дата изменения** — дата и время последнего изменения сервиса.
  - **Изменил** — пользователь, выполнивший последнее изменение сервиса.
  - **Статус конфигурации** — статус Runtime-конфигурации сервиса: **Сохранен**, **Применен** или **Ошибка**.

В нижней части карточки также содержится кнопка **Посмотреть логи**, нажатие на которую позволяет просмотреть список логов сервиса.

Запуск сервиса

Для того чтобы сервис мог начать работу, его нужно запустить.

Чтобы запустить сервис:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.
3. Убедитесь, что в карточке сервиса открыта вкладка **Информация**.
4. Установите переключатель в верхней части карточки в активное положение и подтвердите запуск сервиса. Система запустит сервис и отобразит соответствующее уведомление.

Вы также можете запустить сервис из списка сервисов. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Установите переключатель слева от строки сервиса в списке в активное положение и подтвердите запуск сервиса. Система запустит сервис и отобразит соответствующее уведомление.



Запуск сервиса также возможен через карточку подключения к пространству, в котором создан сервис. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Установите переключатель слева от строки сервиса в карточке подключения в активное положение и подтвердите запуск сервиса. Система запустит сервис и отобразит соответствующее уведомление.

Остановка сервиса

Чтобы остановить работающий сервис:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.
3. Убедитесь, что в карточке сервиса открыта вкладка **Информация**.
4. Установите переключатель в верхней части карточки в неактивное положение и подтвердите остановку сервиса. Система остановит сервис и отобразит соответствующее уведомление.



Вы также можете остановить сервис через список сервисов. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**.

Система отобразит сведения об имеющихся сервисах.

2. Установите переключатель слева от строки сервиса в списке в неактивное положение и подтвердите остановку сервиса. Система остановит сервис и отобразит соответствующее уведомление.

Остановка сервиса также возможна через карточку подключения к пространству, в котором запущен сервис. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Установите переключатель слева от строки сервиса в карточке подключения в неактивное положение и подтвердите остановку сервиса. Система остановит сервис и отобразит соответствующее уведомление.

Работа с логами сервиса

Логирование сервисов обеспечивает запись событий и состояний сервисов, что позволяет администраторам отслеживать и анализировать работу системы.

Функционал просмотра логов доступен через карточку сервиса.

Доступные операции над логами сервиса:

- Просмотр логов сервиса
- Фильтрация списка логов сервиса
- Копирование логов сервиса

Просмотр логов сервиса

Чтобы просмотреть логи сервиса:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.
3. Нажмите на кнопку **Посмотреть логи** в нижней части карточки. Система отобразит окно со списком логов сервиса.

В левом верхнем углу окна отображается ID просматриваемого сервиса.

Доступны два режима вывода логов:

- **Режим мониторинга** — вывод логов в реальном времени.
- **Режим поиска по временному периоду** — вывод логов в рамках заданного периода. Максимальный период запроса — 30 дней.

Переключатель **Мониторинг** позволяет управлять режимом вывода логов.



Время событий отображается в логах в формате `YYYY-MM-DDThh:mm:ss.sssZ` независимо от настроек интерфейса.

Фильтрация списка логов сервиса



Просмотр логов за конкретный временной интервал недоступен в режиме мониторинга.

Чтобы отфильтровать список логов сервиса:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.
3. Нажмите на кнопку **Посмотреть логи** в нижней части карточки. Система отобразит окно со списком логов сервиса.
4. Установите переключатель **Мониторинг** в неактивное положение. Настройки фильтрации станут доступными для редактирования.
5. Настройте период поиска логов. Для этого нажмите на кнопку  и выберите из выпадающего списка способ задания периода:
   - **Последний период** — задайте значение вручную или с помощью кнопок — и +, после чего выберите единицы времени из выпадающего списка. Поиск логов будет выполняться в указанном диапазоне вплоть до нынешнего момента.
   - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск логов.
   - **Часто используемые** — выберите в разделе один из предлагаемых периодов.
6. Нажмите на кнопку **Найти**. Список логов, соответствующий заданному периоду, отобразится в окне.

Копирование логов сервиса

Чтобы скопировать записи журнала логов сервиса:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.
3. Нажмите на кнопку **Посмотреть логи** в нижней части карточки. Система отобразит окно со списком логов сервиса.
4. Нажмите на кнопку **Скопировать** в правой нижней части окна. Система скопирует записи журнала в буфер обмена и отобразит соответствующее уведомление.

Управление доступом

Данный раздел посвящен управлению тенантами и учетными записями пользователей в системе. В подразделах описаны ключевые аспекты и действия, доступные администраторам для работы с тенантами, учетными записями, политиками безопасности, ролями и настройками доступа.

Содержание раздела:

- Тенанты
- Пользователи
- Группы пользователей
- Роли
  - Стандартные роли
- Политики входа и пароля
  - Настройка времени жизни для токенов доступа
- LDAP-соединения
  - Настройка сервисной учетной записи ALD Pro
  - Атрибуты Distinguished Name
- Политики управления доступом к событиям

Тенанты

Данный раздел описывает процесс работы с тенантами в системе. Работа осуществляется в разделе **Настройки** → **Управление доступом** → **Тенанты** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О тенантах

Система поддерживает режим мультитенантности, при котором один экземпляр системы в инфраструктуре организации (далее — **родительский**, или **основной**, тенант) позволяет изолированным частям системы (далее — **дочерние** тенанты) получать, хранить и обрабатывать события независимо друг от друга.

В системе можно работать только в рамках одного тенанта одновременно. Текущий тенант отображается в левом верхнем углу веб-интерфейса под логотипом продукта. В пределах тенанта можно настраивать систему для отдельных групп пользователей, регулируя их права доступа к разделам, доступные операции и доступ к данным хранилища, оставаясь в пределах одной инфраструктуры. События тенантов хранятся отдельно.

В системе доступны следующие типы тенантов:

- **Основной тенант** — тенант, устанавливаемый в системе по умолчанию. Имеет доступ к данным всех дочерних тенантов. Основной тенант нельзя удалить или изменить.

- **Дочерний тенант** — изолированная часть основной инфраструктуры системы, которая содержит ресурсы и данные, выделенные из основного тенанта. Он предназначен для предоставления ограниченного доступа к этим ресурсам и настройкам только определенной группе пользователей. Каждый дочерний тенант работает независимо, обеспечивая доступ и управление только собственными ресурсами и данными.

Доступ к тенантам регулируется в настройках пользователей. Пользователь может иметь доступ к нескольким тенантам одновременно. В каждом тенанте ему могут быть назначены разные роли. Роль состоит из разрешений и действует в указанных тенантах. Если роль не привязана к конкретному тенанту, она доступна во всех тенантах. Один и тот же пользователь может иметь разные роли в разных тенантах.

При формировании API-запроса к системе, работа происходит в том тенанте, который указан в заголовке запроса.

Наследование тенанта

Ресурсы и события системы доступны только в том тенанте, к которому они относятся.

Объекту автоматически присваивается идентификатор тенанта, в рамках которого он был создан. Этот идентификатор указывается в поле `tenantId` объекта. После создания объекта его тенант невозможно изменить.



Текущий тенант отображается в левом верхнем углу веб-интерфейса системы под логотипом программы.

Например, тенант корреляционного события наследуется от создавшего его коллектора. Коллектор получает идентификатор того тенанта, в котором он был создан.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Добавить** (＋) позволяет создать новый тенант.
- Поле **Поиск** предназначено для быстрого поиска тенантов в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся в системе тенантов, которые доступны пользователю. Таблица содержит следующие столбцы:

- **Активен** — текущий статус тенанта: **Включен** или **Выключен**.
- **Имя** — имя тенанта, используемое для его идентификации в системе.
- **Описание** — описание тенанта.

При работе с таблицей тенантов доступны следующие операции:

- Поиск тенантов по полю **Имя**.
- Сортировка тенантов по полю **Имя**.

При выборе конкретного тенанта в правой части рабочей области отображается его карточка с детальной информацией.

Работа с тенантами

Доступные операции над тенантами:

- Добавление тенанта
- Просмотр тенанта
- Включение тенанта
- Отключение тенанта
- Изменение тенанта
- Переключение между тенантами

Добавление тенанта

Чтобы добавить тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Нажмите на кнопку **Добавить** (＋). Отобразится окно добавления тенанта.
3. Введите название тенанта.
4. При необходимости введите описание тенанта.
5. Нажмите на кнопку **Добавить**. Система создаст тенант и отобразит соответствующее уведомление. Новый тенант появится в списке раздела.

Просмотр тенанта

Чтобы просмотреть тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Нажмите на строку тенанта в списке. Система отобразит в правой части экрана карточку этого тенанта с подробной информацией о нем.

В верхней части карточки отображается название тенанта. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранного тенанта.

Карточка содержит основные сведения о тенанте:

- **ID** — уникальный идентификационный код тенанта, генерируемый автоматически при его создании.
- **Родительский тенант** — основной тенант, к которому принадлежит текущий тенант.
- **Имя** — имя тенанта, используемое для его идентификации в системе.
- **Статус** — текущий статус тенанта: **Включен** или **Выключен**.
- **Описание** — описание тенанта.
- **Дата создания** — дата и время создания тенанта.

- **Создал** — пользователь, инициировавший создание тенанта.
- **Дата изменения** — дата и время последнего изменения тенанта.
- **Изменил** — пользователь, выполнивший последнее изменение тенанта.

В верхней части карточки также расположен переключатель статуса тенанта.

Включение тенанта

Включение тенанта делает его доступным для выбора в списке доступных тенантов системы.

Чтобы включить тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Установите переключатель состояния в столбце **Активен** в строке нужного тенанта в активное положение. Система включит тенант и отобразит соответствующее уведомление. Тенант станет доступным для выбора.



Вы также можете включить тенант через его карточку. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Нажмите на строку тенанта в списке. Система отобразит в правой части экрана карточку этого тенанта с подробной информацией о нем.
3. Установите переключатель состояния тенанта в верхней части карточки в активное положение. Система включит тенант и отобразит соответствующее уведомление. Тенант станет доступным для выбора.

Отключение тенанта

Отключение тенанта делает его недоступным для выбора в списке доступных тенантов системы.

Чтобы отключить тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Установите переключатель состояния в столбце **Активен** в строке нужного тенанта в неактивное положение. Система отобразит окно подтверждения действия.
3. Нажмите на кнопку **Отключить**. Система отключит тенант и отобразит соответствующее уведомление. Тенант перестанет быть доступным для выбора.



Вы также можете отключить тенант через его карточку. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Нажмите на строку тенанта в списке. Система отобразит в правой части экрана карточку этого тенанта с подробной информацией о нем.

3. Установите переключатель в верхней части тенанта в неактивное положение. Отобразится окно подтверждения действия.
4. Нажмите на кнопку **Отключить**. Система отключит тенант и отобразит соответствующее уведомление. Тенант перестанет быть доступным для выбора.

Изменение тенанта

Вы можете изменить тенант через меню действий в его карточке.

Переключение между тенантами

Одновременно можно работать только в рамках одного тенанта. Если пользователь имеет доступ к нескольким тенантам, он может выбрать, данные какого тенанта будут отображаться в веб-интерфейсе системы.

Чтобы выбрать тенант:

1. На главной панели веб-интерфейса под логотипом продукта нажмите на название тенанта. Система отобразит список доступных тенантов.
2. Выберите тенант, данные которого необходимо отобразить в веб-интерфейсе системы. В разделах веб-интерфейса отобразятся только данные и аналитика, которые относятся к выбранному тенанту.

Пользователи

Данный раздел описывает процесс работы с учетными записями пользователей в системе. Работа осуществляется в разделе **Настройки** → **Управление доступом** → **Пользователи** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О пользователях

Учетная запись пользователя — это средство идентификации и аутентификации лица в системе, включающее в себя уникальные учетные данные, такие как логин и пароль, а также набор присвоенных ролей, которые определяют доступные для пользователя разделы интерфейса и операции в системе. Метаданные учетной записи могут содержать информацию о ФИО, должности и дополнительное описание.

В системе можно создавать учетные записи следующих типов:

- **Локальный пользователь** — учетная запись для доступа и работы пользователей в системе.
- **Пользователь из домена** — учетная запись, созданная через интеграцию с LDAP-сервером и предназначенная для управления пользователями из внешнего каталога.
- **Бот** — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании учетной записи типа **Бот** ей присваивается API-токен.

Роли и разрешения пользователей

Пользователю могут быть назначены одна или несколько ролей. Роли могут быть назначены как непосредственно учетной записи, так и через группы пользователей, в которых состоит пользователь. В разных тенантах один и тот же пользователь может иметь разные роли. Разрешения из всех ролей пользователя в рамках одного тенанта суммируются. При конфликте разрешений применяется наибольшее из доступных. Пользователь может иметь одну или несколько ролей, однако наличие хотя бы одной роли является обязательным.

Предусмотрена возможность создания пользовательских ролей или использования стандартных.

В системе по умолчанию установлены следующие стандартные роли со своим набором разрешений:

- **Администратор** — отвечает за настройку и поддержку инфраструктуры системы, обновление конфигураций и обеспечение безопасности данных. Имеет полный доступ к административным функциям. Обладает всеми возможными разрешениями в системе, имеет полный контроль над всеми тенантами системы.



При создании учетной записи пользователя, ей можно присвоить статус **Суперадминистратор**. Это предоставит все доступные в системе разрешения.

- **Инженер** — занимается настройкой операционных процессов системы, разработкой решений для оптимизации этих процессов и их внедрением. Отвечает за создание и установку конфигураций конвейера, настройку коллектора. Действия инженера ограничены рамками определенных тенантов.
- **Пользователь** — работает с данными системы в рамках определенных тенантов, выполняет анализ и визуализацию данных для поддержки управленческих решений. Имеет ограниченный доступ к ресурсам системы.

Учетную запись с разрешениями администратора может создать и редактировать только другой администратор. Из системы нельзя удалить последнюю учетную запись со статусом **Суперадминистратор**.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Добавить** (+) позволяет добавить нового пользователя.
- Поле **Поиск** предназначено для быстрого поиска пользователей в отображаемом списке по полям **ФИО** и **Логин**.
- Выпадающий список **Роль** позволяет отфильтровать список пользователей по ролям.
- Выпадающий список **Тип** позволяет отфильтровать список пользователей по типам учетной записи.

В рабочей области отображается таблица имеющихся в системе учетных записей пользователей. Таблица содержит следующие столбцы:

- **Логин** — логин пользователя в системе.
- **Статус** — текущий статус пользователя: **Активен** или **Заблокирован**.

- **Тип пользователя** — тип учетной записи пользователя.
- **ФИО** — полные фамилия, имя и отчество пользователя.
- **Роли** — перечень ролей, назначенных пользователю.

При работе с таблицей учетных записей доступны следующие операции:

- Поиск пользователей по полям **Логин** и **ФИО**.
- Фильтрация пользователей по полям **Роль** и **Тип пользователя**. Роли и типы выбираются из выпадающих списков.
- Сортировка пользователей по полям **Логин**, **Тип пользователя** и **ФИО**.

При выборе конкретной учетной записи в правой части рабочей области отображается ее карточка с детальной информацией.

Работа с пользователем

Доступные операции над пользователем:

- Добавление пользователя
- Просмотр пользователя
- Блокировка пользователя
- Активация пользователя
- Изменение пользователя
- Удаление пользователя

Добавление пользователя

Вы можете добавить учетную запись пользователя с нужным набором ролей и параметров.

В системе можно создавать учетные записи следующих типов:

- **Локальный пользователь** — учетная запись для доступа и работы пользователей в системе.
- **Пользователь из домена** — учетная запись, созданная через интеграцию с LDAP-сервером и предназначенная для управления пользователями из внешнего каталога.
- **Бот** — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании учетной записи типа **Бот** ей присваивается API-токен.

Добавление локального пользователя

Чтобы добавить локального пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на кнопку **Добавить** (+) и выберите из выпадающего списка тип **Локальный пользователь**. Система отобразит окно добавления локального пользователя.
3. Заполните следующие поля:

- **Логин** — задайте уникальный логин для учетной записи пользователя в системе.
- **Пароль** — установите пароль для входа пользователя в систему. Пароль должен удовлетворять заданным требованиям.
- **Группа пользователей** (опционально) — выберите из выпадающего списка группы, в которые должен входить пользователь.
- **ФИО** (опционально) — укажите полное имя пользователя.
- **Должность** — укажите должность пользователя в организации.
- **Номер телефона** (опционально) — укажите номер телефона пользователя.
- **Email** (опционально) — укажите адрес электронной почты пользователя.
- **Описание** (опционально) — добавьте краткое описание учетной записи пользователя.

4. Назначьте роли пользователю, чтобы определить перечень его разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.

   b. Выберите из выпадающего списка роль, которую необходимо назначить пользователю.

   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   > Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.

   > Если активирован переключатель **Суперадминистратор**, учетной записи предоставляются все доступные в системе разрешения.

5. Нажмите на кнопку **Добавить**. Система создаст учетную запись пользователя и отобразит соответствующее уведомление. Новая учетная запись появится в списке раздела.

Добавление пользователя из домена

Пользователь из домена — учетная запись, созданная через интеграцию с LDAP-сервером, для управления пользователями из внешнего каталога. Синхронизируются следующие поля учетных записей: логин, ФИО, должность, статус, электронная почта, номер телефона.

Чтобы добавить пользователя из домена:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на кнопку **Добавить** (+) и выберите из выпадающего списка тип **Пользователь из домена**. Система отобразит окно добавления доменных пользователей.
3. Выберите одно из доступных LDAP-соединений из выпадающего списка **Домен**. Система отобразит каталоги выбранного LDAP-соединения в виде дерева в поле **Каталоги**.
4. Выберите необходимый каталог из дерева. Система отобразит информацию о пользователях из выбранного каталога в поле **Пользователи**.

Для быстрого поиска каталога введите запрос в строку поиска над деревом каталогов.



По умолчанию поиск выполняется с начала названия каталога. Например, если необходимо найти каталог с названием "domain users", то его можно найти по запросу `domain`. По запросу `users` данный каталог найден не будет.

Чтобы выполнить поиск по подстроке, используйте следующий синтаксис: `*подстрока*`. Например, для поиска каталога с названием "domain users" по подстроке "users" используйте запрос `*users*`.

5. Выберите одного или нескольких пользователей, для которых необходимо создать учетную запись. Можно выбрать до 10 пользователей из одного или нескольких каталогов.

   Для быстрого поиска пользователей введите запрос в строку поиска над списком пользователей. В списке отобразятся пользователи, в именах которых встречается введенный запрос.

6. При необходимости выберите из выпадающего списка группы, в которые должны входить доменные пользователи.
7. Нажмите на кнопку **Далее**. Отобразится вкладка для настройки разрешений пользователей.
8. Назначьте роли доменным пользователям, чтобы определить перечень их разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.
   b. Выберите из выпадающего списка роль, которую необходимо назначить пользователю.
   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

      Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы удалить добавленную роль, нажмите на кнопку  в строке настройки роли.



Если активирован переключатель **Суперадминистратор**, учетной записи предоставляются все доступные в системе разрешения.

9. Нажмите на кнопку **Создать**. Система создаст учетные записи для выбранных пользователей и отобразит соответствующее уведомление. При создании учетных записей будут синхронизированы следующие поля: логин, ФИО, должность, статус, email, номер телефона. Новые учетные записи появятся в списке раздела.



Если в систему добавлены пользователи с одинаковыми логинами, но из разных доменов, то при входе им нужно дополнительно указывать домен в поле логина.

Добавление бота

Бот — учетная запись, предназначенная для автоматизированного взаимодействия с системой через

API-запросы. При создании учетной записи типа **Бот** ей автоматически присваивается API-токен.

Чтобы создать бота:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на кнопку **Добавить** (+) и выберите из выпадающего списка тип **Бот**. Система отобразит окно добавления бота.
3. Заполните следующие поля:
   - **Логин** — задайте уникальный логин для учетной записи пользователя в системе.
   - **Группа пользователей** (опционально) — выберите из выпадающего списка группы, в которые должен входить пользователь.
   - **Описание** (опционально) — добавьте краткое описание учетной записи пользователя.
4. Назначьте роли пользователю, чтобы определить перечень его разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.
   b. Выберите из выпадающего списка роль, которую необходимо назначить пользователю.
   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   > Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.

   > Если активирован переключатель **Суперадминистратор**, учетной записи предоставляются все доступные в системе разрешения.

5. Нажмите на кнопку **Добавить**. Система создаст бота и отобразит соответствующее уведомление. Новый бот появится в списке раздела.

> При создании бота система автоматически генерирует API-токен, который используется для авторизации в API-запросах.

Просмотр пользователя

Чтобы просмотреть пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на строку пользователя в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.

В верхней части карточки отображается логин учетной записи. Справа от логина расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранной учетной записи.

Карточка учетной записи содержит следующие вкладки:

- **Основное** — основная информация об учетной записи пользователя.
- **Роли** — список ролей, назначенных пользователю.
- **Разрешения** — список разрешений пользователя.

Для учетной записи типа **Бот** в нижней части карточки отображаются следующие кнопки:

- **Скопировать токен** ( ) — копирует токен в буфер обмена.
- **Перевыпустить токен** ( ) — обновляет API-токен бота, делая предыдущий токен недействительным.

Вкладка Основное

На вкладке **Основное** отображается основная информация об учетной записи. Вкладка содержит следующие поля:

- **Логин** — уникальный идентификатор пользователя в системе.
- **Статус** — текущий статус пользователя: **Активен** или **Заблокирован**.
- **Тип** — тип учетной записи пользователя: локальный, доменный или бот.
- **Описание** — краткое описание учетной записи пользователя.
- **Дата создания** — дата и время создания учетной записи.
- **Создал** — пользователь, инициировавший создание учетной записи.
- **Дата изменения** — дата и время последнего изменения учетной записи.
- **Изменил** — пользователь, выполнивший последнее изменение учетной записи.
- **ФИО** — полное имя пользователя.
- **Должность** — должность пользователя в организации.
- **Суперадминистратор** — указывает, обладает ли пользователь статусом **Суперадминистратор**.
- **Номер телефона** — номер телефона пользователя.
- **Email** — адрес электронной почты пользователя.

В верхней части вкладки также содержится переключатель статуса пользователя.

Вкладка Роли



Для пользователей со статусом **Суперадминистратор** эта вкладка не отображается.

На вкладке **Роли** приведен перечень ролей, назначенных пользователю.

По умолчанию для каждой роли отображается только ее название. Чтобы получить более подробные сведения о роли, нажмите на кнопку в правой части ее строки. Для роли отобразятся следующие сведения:

- **Источники** — список источников роли:

- Если роль была добавлена вручную при создании учетной записи, в качестве источника отображается **Собственная роль**.
- Если роль была наследована от группы, в которой содержится учетная запись, в качестве источника отображается название этой группы.

- **Тенанты** — список тенантов, в которых действует роль.

Вкладка Разрешения



Для пользователей со статусом **Суперадминистратор** эта вкладка не отображается.

На вкладке **Разрешения** приведен полный перечень разрешений, которыми обладает пользователь.

Список разрешений представлен в виде таблицы со следующими столбцами:

- **Разрешение** — название разрешения.
- **Раздел** — раздел системы, который затрагивается разрешением. Путь до раздела указывается через /.
- **Модуль** — функциональный модуль платформы, к которому относится разрешение.
- **Тенанты и организации** — список тенантов, в рамках которых выдано разрешение.

При работе с таблицей разрешений доступны следующие операции:

- Поиск разрешений по всем полям.
- Сортировка разрешений по полям **Разрешение** и **Модуль**.

Блокировка пользователя

При необходимости вы можете заблокировать учетную запись пользователя, чтобы запретить ему авторизоваться и работать с системой.

Чтобы заблокировать учетную запись пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на строку пользователя в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.
3. Убедитесь, что в карточке открыта вкладка **Основное**.
4. В верхней части карточки установите переключатель статуса в неактивное положение. Система заблокирует пользователя и отобразит соответствующее уведомление.

Активация пользователя

Активация заблокированной учетной записи возобновляет возможность авторизоваться и работать с системой через эту учетную запись.



Заблокированные учетные записи могут быть активированы только администраторами, пользователями с соответствующими разрешениями в составе

своих ролей и учетными записями со статусом **Суперадминистратор**.

Чтобы активировать учетную запись пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на строку пользователя в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.
3. Убедитесь, что в карточке открыта вкладка **Основное**.
4. В верхней части карточки установите переключатель статуса в активное положение. Система активирует пользователя и отобразит соответствующее уведомление.

Изменение пользователя

Вы можете изменить пользователя через меню действий в его карточке.



При изменении списка ролей пользователя доступные ему разделы меню обновятся после перезагрузки страницы или повторного входа в систему.

Удаление пользователя

Вы можете удалить пользователя следующими способами:

- через меню действий в карточке пользователя;
- через массовое удаление пользователей.



Если пользователь из домена был добавлен в систему автоматически при создании или синхронизации доменной группы, то он автоматически удаляется из системы в следующих случаях:

- при удалении доменной группы;
- при удалении пользователя из доменной группы и ее повторной синхронизации.

Группы пользователей

Данный раздел описывает процесс работы с группами пользователей в системе. Работа осуществляется в разделе **Настройки** → **Управление доступом** → **Группы пользователей** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О группах пользователей

Группа пользователей объединяет учетные записи для назначения общих прав управления системой. В системе поддерживаются следующие типы групп пользователей:

- **Локальная группа** — группа, куда могут быть включены любые учетные записи в системе. Ее состав регулируется вручную через веб-интерфейс системы.
- **Доменная группа** — группа, связанная с каталогом LDAP. Ее состав и данные учетных записей автоматически синхронизируются с указанным LDAP-каталогом.

Группам пользователей могут быть назначены роли, которые определяют права пользователей группы в системе. Одной группе можно присвоить сразу несколько ролей.



Группе не может быть присвоен статус **Суперадминистратор** — этот статус может быть присвоен только индивидуальным пользователям.

Назначение ролей в группах пользователей является необязательным. Группу можно создать и сохранить без ролей, в этом случае пользователи группы не получат дополнительных прав. Группы пользователей не привязаны к тенантам системы.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Добавить** (+) позволяет добавить новую группу пользователей.
- Выпадающий список **Тип** позволяет отфильтровать список групп пользователей по типу.
- Поле **Поиск** предназначено для быстрого поиска пользователей в отображаемом списке по полям **Название** и **Ключ**.

В рабочей области отображается таблица имеющихся в системе групп пользователей. Таблица содержит следующие столбцы:

- **Ключ** — уникальный идентификатор группы.
- **Название** — наименование группы пользователей.
- **Тип группы** — тип группы: локальная или доменная.
- **Описание** — описание группы.
- **Дата создания** — дата и время создания группы.
- **Дата изменения** — дата и время последнего изменения группы.

При работе с таблицей групп пользователей доступны следующие операции:

- Поиск групп пользователей по полям **Название** и **Ключ**.
- Фильтрация групп пользователей по полю **Тип группы**. Тип группы выбирается из выпадающего списка.
- Сортировка групп пользователей по полям **Ключ**, **Название**, **Дата создания** и **Дата изменения**.

При выборе конкретной группы в правой части рабочей области отображается ее карточка с детальной информацией.

Работа с группой пользователей

Доступные операции над группой пользователей:

- Создание группы пользователей
- Просмотр группы пользователей
- Изменение группы пользователей
- Изменение состава локальной группы пользователей
- Синхронизация группы пользователей
- Удаление группы пользователей

Создание группы пользователей

В системе можно создавать группы пользователей следующих типов:

- **Локальная группа** — группа, куда могут быть включены любые пользователи системы.
- **Доменная группа** — группа, связанная с каталогом LDAP.

Создание локальной группы пользователей



При необходимости вы можете сохранить группу без пользователей.

Чтобы создать локальную группу пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит сведения о доступных группах.
2. Нажмите на кнопку **Создать** (+) и выберите из выпадающего списка тип **Локальная группа**. Система отобразит окно добавления локальной группы.
3. Заполните следующие поля:
   - **Название** — задайте уникальное название для группы пользователей.
   - **Ключ** — задайте уникальный идентификатор для группы. Ключ может содержать цифры (1–9), латинские буквы (a–z) и специальные символы (-_).
   - **Описание** (опционально) — добавьте краткое описание группы.
4. При необходимости назначьте роли группе пользователей, чтобы определить перечень их разрешений в системе.

   По умолчанию в окне уже присутствует строка для настройки роли. Чтобы задать настройки роли:

   a. Выберите из выпадающего списка роль, которую необходимо назначить всем пользователям в группе.

   b. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы добавить новую роль, нажмите на кнопку **Добавить роль** (+). Появится строка настройки новой роли.

Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.



Если требуется создать группу без ролей, удалите все строки настроек ролей, в том числе и строку, которая присутствует в окне по умолчанию.

5. При необходимости в поле **Состав группы** установите флажки слева от пользователей, которые должны быть включены в группу.

   Список пользователей представлен в виде таблицы со следующими столбцами:

   - **Логин** — логин пользователя в системе.
   - **ФИО** — полное имя пользователя.

   При работе с таблицей пользователей доступны следующие операции:

   - Поиск пользователей по всем полям.
   - Сортировка пользователей по всем полям.

   Чтобы отобразить в таблице только уже выбранных пользователей, установите переключатель **Только выбранное** над таблицей в активное положение.

6. Нажмите на кнопку **Создать**. Система создаст локальную группу пользователей и отобразит соответствующее уведомление. Новая группа появится в списке раздела.

Создание доменной группы пользователей

Чтобы создать доменную группу пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит сведения о доступных группах.
2. Нажмите на кнопку **Создать** (+) и выберите из выпадающего списка тип **Доменная группа**. Система отобразит окно добавления доменной группы.
3. Заполните следующие поля:
   - **Название** — задайте уникальное название для группы пользователей.
   - **Ключ** — задайте уникальный идентификатор для группы. Ключ может содержать цифры (1–9), латинские буквы (a–z) и специальные символы (-_).
   - **Описание** (опционально) — добавьте краткое описание группы.
4. Выберите одно из доступных LDAP-соединений из выпадающего списка. Система отобразит каталоги и группы выбранного LDAP-соединения в виде дерева в поле **Группы**.
5. Выберите необходимую группу пользователей из дерева. Система отобразит список входящих в группу пользователей в поле **Пользователи**.

   Для быстрого поиска каталога или группы введите запрос в строку поиска над деревом каталогов.

   По умолчанию поиск выполняется с начала названия каталога или группы. Например, если необходимо найти каталог с названием "domain users", то его

можно найти по запросу `domain`. По запросу `users` данный каталог найден не будет.

Чтобы выполнить поиск по подстроке, используйте следующий синтаксис: `*подстрока*`. Например, для поиска каталога с названием "domain users" по подстроке "users" используйте запрос `*users*`.

6. Нажмите на кнопку **Далее**. Отобразится вкладка для настройки разрешений в группе.
7. Назначьте роли доменным пользователям, чтобы определить перечень их разрешений в системе.

   По умолчанию в окне уже присутствует строка для настройки роли. Чтобы задать настройки роли:

   a. Выберите из выпадающего списка роль, которую необходимо назначить всем пользователям в группе.

   b. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы добавить новую роль, нажмите на кнопку **Добавить роль** (+). Появится строка настройки новой роли.

   Чтобы удалить добавленную роль, нажмите на кнопку  в строке настройки роли.

   Если требуется создать группу без ролей, удалите все строки настроек ролей, в том числе и строку, которая присутствует в окне по умолчанию.

8. При необходимости в поле **Состав группы** установите флажки слева от пользователей, которые должны быть включены в группу.

   Список пользователей представлен в виде таблицы со следующими столбцами:

   - **Логин** — логин пользователя в системе.
   - **ФИО** — полное имя пользователя.

   При работе с таблицей пользователей доступны следующие операции:

   - Поиск пользователей по всем полям.
   - Сортировка пользователей по всем полям.

   Чтобы отобразить в таблице только уже выбранных пользователей, установите переключатель **Только выбранное** над таблицей в активное положение.

9. Нажмите на кнопку **Создать**. Система создаст доменную группу пользователей и отобразит соответствующее уведомление. Новая группа появится в списке раздела.

После создания группы система выполнит синхронизацию, в ходе которой состав группы обновится

в соответствии с указанными параметрами. При этом в систему автоматически будут добавлены учетные записи для пользователей из этой группы.

Просмотр группы пользователей

Чтобы просмотреть группу пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.

В верхней части карточки отображается название группы. Справа от названия расположены кнопки действий, доступных для выбранной группы.

Карточка группы пользователей содержит следующие вкладки:

- **Основное** — основная информация о группе.
- **Состав группы** — список пользователей, входящих в группу.
- **Роли** — список ролей, назначенных группе.
- **Разрешения** — список разрешений группы.

Для доменной группы в нижней части карточки также содержится кнопка **Синхронизировать** (🔄) для синхронизации группы.

Вкладка Основное

На вкладке **Основное** отображается основная информация о группе пользователей. Вкладка содержит следующие поля:

- **Название** — имя группы.
- **Ключ** — уникальный идентификатор группы в системе.
- **Тип группы** — тип группы: локальная или доменная.
- **Описание** — краткое описание группы.
- **Дата создания** — дата и время создания группы.
- **Создал** — пользователь, инициировавший создание группы.
- **Дата изменения** — дата и время последнего изменения группы.
- **Изменил** — пользователь, выполнивший последнее изменение группы.
- **LDAP-домен** — имя LDAP-домена. Поле отображается только для доменных групп.
- **Название в LDAP** — название группы в LDAP-каталоге. Поле отображается только для доменных групп.
- **Синхронизировано** — дата и время последней синхронизации с LDAP. Поле отображается только для доменных групп.

Вкладка Состав группы

На вкладке **Состав группы** отображается список пользователей, входящих в группу.

Список пользователей представлен в виде таблицы со следующими столбцами:

- **Логин** — логин пользователя.
- **ФИО** — полное имя пользователя.

В таблице доступна сортировка пользователей по всем полям.

Для локальных групп на вкладке доступна опция изменения состава группы через кнопки и . Для доменных групп вкладка **Состав группы** отображает список пользователей без возможности его изменения. Состав доменной группы изменяется автоматически при синхронизации с LDAP-каталогом.

Вкладка Роли

На вкладке **Роли** карточки группы пользователей отображается список ролей, которые назначаются пользователям в группе.

По умолчанию для каждой роли отображается только ее название. Чтобы получить более подробные сведения о роли, нажмите на кнопку в правой части ее строки. Для роли отобразится список тенантов, в которых она действует.

Вкладка Разрешения

На вкладке **Разрешения** карточки группы пользователей приведен полный перечень разрешений, которыми обладают пользователи в группе.

Список разрешений представлен в виде таблицы со следующими столбцами:

- **Разрешение** — название разрешения.
- **Раздел** — раздел системы, который затрагивается разрешением. Путь до раздела указывается через /.
- **Модуль** — функциональный модуль платформы, к которому относится разрешение.
- **Тенанты и организации** — список тенантов, в рамках которых выдано разрешение.

При работе с таблицей разрешений доступны следующие операции:

- Поиск разрешений по всем полям.
- Сортировка разрешений по полям **Разрешение** и **Модуль**.

Изменение группы пользователей

Вы можете изменить группу пользователей через кнопку в ее карточке.

Изменение состава локальной группы пользователей



Состав группы можно изменить только в локальной группе. Для доменной группы пользователи добавляются или удаляются автоматически при синхронизации с

LDAP-каталогом.

Вы можете изменить состав локальной группы, добавив в нее новых пользователей или удалив из нее существующих.



Вы также можете добавлять пользователей в группу при их создании.

Чтобы изменить состав локальной группы пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.
3. Перейдите на вкладку **Состав группы**.
4. Нажмите на кнопку **Изменить** () в верхней части вкладки. Откроется окно изменения состава группы.

   В окне отображается список доступных пользователей в системе. Список представлен в виде таблицы со следующими столбцами:

   - **Логин** — логин пользователя.
   - **ФИО** — полное имя пользователя.

   При работе с таблицей пользователей доступны следующие операции:

   - Поиск пользователей по всем полям.
   - Сортировка пользователей по всем полям.

   Чтобы добавить нового пользователя в группу, установите флажок слева от его строки.

   Чтобы удалить существующего пользователя из группы, уберите флажок слева от его строки.

   Чтобы отобразить в таблице только уже выбранных пользователей, установите переключатель **Только выбранное** над таблицей в активное положение.

5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке группы.



Удалять пользователей из группы также можно через таблицу на вкладке **Состав группы**. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.
3. Перейдите на вкладку **Состав группы**.
4. Наведите курсор на строку пользователя, которого необходимо удалить, и

нажмите на появившуюся кнопку в правой части строки. Система удалит пользователя из группы и отобразит соответствующее уведомление.

Синхронизация группы пользователей

Синхронизация доступна только для доменных групп пользователей. Изменение состава локальных групп выполняется вручную.

Синхронизация обновляет состав группы пользователей в системе на основе данных из LDAP-сервера. Синхронизируются следующие поля учетных записей: логин, ФИО, должность, статус, электронная почта, номер телефона. При этом:

- Если в доменной группе появился новый пользователь, в системе для него будет автоматически создана учетная запись.
- Если из доменной группы был удален пользователь, то его учетная запись будет автоматически удалена из системы.

Чтобы синхронизировать данные доменной группы:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.
3. Нажмите на кнопку **Синхронизировать** () в нижней части карточки. Начнется синхронизация данных группы с LDAP-сервером.

   По завершении операции система отобразит уведомление о статусе синхронизации.

Удаление группы пользователей

Вы можете удалить группу пользователей через кнопку в ее карточке.

При удалении доменной группы из системы автоматически будут удалены все пользователи из этой группы.

Роли

Данный раздел описывает процесс работы с ролями пользователей в системе. Работа осуществляется в разделе **Настройки** → **Управление доступом** → **Роли** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О ролях

Роль — это набор разрешений, который определяет доступные для пользователя разделы интерфейса и операции в системе.

Разрешение — это право выполнения определенного действия или наличие доступа к определенной функциональности системы. Разрешения в системе разделены по модулям.

Роли могут быть назначены пользователям индивидуально или через группы пользователей. Пользователь может иметь одну или несколько ролей одновременно. Если роль не привязана к конкретному тенанту, она доступна во всех тенантах системы. В разных тенантах один и тот же пользователь может иметь разные роли. Разрешения из всех ролей пользователя в рамках одного тенанта суммируются. При конфликте разрешений применяется наибольшее из доступных. Пользователь может иметь одну или несколько ролей, однако наличие хотя бы одной роли является обязательным.

Система поддерживает создание пользовательских ролей с произвольными наборами разрешений.

Стандартные роли

В системе по умолчанию установлены следующие стандартные роли с предопределенными наборами разрешений:

- **Администратор** — отвечает за настройку и поддержку инфраструктуры системы, обновление конфигураций и обеспечение безопасности данных. Имеет полный доступ к административным функциям. Обладает всеми возможными разрешениями в системе, имеет полный контроль над всеми тенантами системы.



При создании учетной записи пользователя, ей можно присвоить статус **Суперадминистратор**. Это предоставит все доступные в системе разрешения.

- **Инженер** — занимается настройкой операционных процессов системы, разработкой решений для оптимизации этих процессов и их внедрением. Отвечает за создание и установку конфигураций конвейера, настройку коллектора. Действия инженера ограничены рамками определенных тенантов.
- **Пользователь** — работает с данными системы в рамках определенных тенантов, выполняет анализ и визуализацию данных для поддержки управленческих решений. Имеет ограниченный доступ к ресурсам системы.

Суперадминистраторы

Пользователи со статусом **Суперадминистратор** имеют высший уровень доступа и могут выполнять любые операции в системе, включая создание и управление другими ролями и пользователями.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую роль.
- Поле **Поиск** предназначено для быстрого поиска ролей в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся в системе ролей, которые доступны пользователю. Таблица содержит следующие столбцы:

- **Имя** — имя роли, используемое для ее идентификации в системе.

- **Тип** — тип роли: стандартная или пользовательская.
- **Описание** — описание роли.

При работе с таблицей ролей доступны следующие операции:

- Поиск ролей по полю **Имя**.
- Сортировка ролей по полям **Имя** и **Тип**.

При выборе конкретной роли в правой части рабочей области отображается ее карточка с детальной информацией.

Работа с ролью

Доступные операции над ролью:

- Создание роли
- Просмотр роли
- Изменение роли
- Создание копии роли
- Удаление роли

Создание роли

Вы можете создать роль с нужным набором разрешений.

Чтобы создать роль:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Нажмите на кнопку **Создать** (+). Система отобразит окно создания роли.
3. Введите название роли.
4. При необходимости введите описание роли.
5. В поле **Разрешения** установите флажки слева от разрешений, которыми должна обладать роль. Разрешения представлены в виде дерева и сгруппированы по модулям платформы и разделам системы.



Если у разрешения есть связанные разрешения, необходимые для его корректной работы, при его выборе они выдаются автоматически. При этом связанные разрешения становятся недоступными для отключения.

Так, если выдать разрешение на управление сущностью системы, автоматически будет выдано разрешение и на ее просмотр. Например, при выборе разрешения **Управление секретами** дополнительно выдается разрешение **Просмотр секретов**.

Чтобы выдать для роли сразу все разрешения из конкретного модуля или раздела, установите флажок слева от его заголовка в дереве. При этом, если в будущих версиях продукта в рамках

выбранного модуля или раздела появятся новые разрешения, они будут автоматически выданы для данной роли.

Для быстрого поиска разрешений введите запрос в строку поиска над деревом разрешений. В дереве отобразятся только те разрешения, в названии которых содержится введенный запрос.



Чтобы отобразить в дереве только уже выбранные разрешения, установите переключатель **Только выбранное** над деревом разрешений в активное положение.

6. Нажмите на кнопку **Создать**. Система создаст роль и отобразит соответствующее уведомление. Новая роль появится в списке раздела.

Просмотр роли

Чтобы просмотреть роль:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Нажмите на строку роли в списке. Система отобразит в правой части экрана карточку этой роли с подробной информацией о ней.

В верхней части карточки отображается название роли. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранной роли.

Карточка роли содержит следующие вкладки:

- **Разрешения** — список разрешений роли.
- **Информация** — основная информация о роли.

Вкладка Разрешения

На вкладке **Разрешения** приведен полный перечень разрешений, назначенных роли.

Список разрешений представлен в виде таблицы со следующими столбцами:

- **Разрешение** — название разрешения.
- **Раздел** — раздел системы, который затрагивается разрешением. Путь до раздела указывается через /.
- **Модуль** — функциональный модуль платформы, к которому относится разрешение.

При работе с таблицей разрешений доступны следующие операции:

- Поиск разрешений по всем полям.
- Сортировка разрешений по полям **Разрешение** и **Модуль**.

Вкладка Информация

На вкладке **Информация** отображается основная информация о роли. Вкладка содержит следующие

поля:

- **Имя** — имя роли, используемое для ее идентификации в системе.
- **Тип** — тип роли: стандартная или пользовательская.
- **Описание** — описание роли.
- **Дата создания** — дата и время создания роли.
- **Создал** — пользователь, инициировавший создание роли.
- **Дата изменения** — дата и время последнего изменения роли.
- **Изменил** — пользователь, выполнивший последнее изменение роли.

Изменение роли

Вы можете изменить роль через меню действий в ее карточке.



При изменении разрешений роли, разделы меню, доступные пользователям с этой ролью, обновятся после перезагрузки страницы или повторного входа в систему.

Создание копии роли

Чтобы создать копию роли:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Наведите курсор на строку роли в списке и нажмите на появившуюся кнопку в правой части строки. Отобразится окно создания копии выбранной роли.
3. Внесите требуемые изменения в конфигурацию роли.
4. Нажмите на кнопку **Сохранить**. Система создаст копию выбранной роли и отобразит соответствующее уведомление. Новая роль появится в списке раздела.



Вы также можете создать копию роли из ее карточки. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Нажмите на строку роли в списке. Система отобразит в правой части экрана карточку этой роли с подробной информацией о ней.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Создать копию**. Отобразится окно создания копии выбранной роли.
4. Внесите требуемые изменения в конфигурацию роли.
5. Нажмите на кнопку **Сохранить**. Система создаст копию выбранной роли и отобразит соответствующее уведомление. Новая роль появится в списке раздела.

Удаление роли

Вы можете удалить роль через меню действий в ее карточке.

Стандартные роли

О стандартных ролях

Роль — это набор разрешений, который определяет доступные для пользователя разделы интерфейса и операции в системе.

Стандартные роли — это роли с предопределенным набором разрешений. Стандартные роли не могут быть изменены или удалены.

В системе предустановлены следующие стандартные роли:

- **Администратор** — отвечает за настройку и поддержку инфраструктуры системы, обновление конфигураций и обеспечение безопасности данных. Имеет полный доступ к административным функциям. Обладает всеми возможными разрешениями в системе, имеет полный контроль над всеми тенантами системы.



При создании учетной записи пользователя, ей можно присвоить статус **Суперадминистратор**. Это предоставит все доступные в системе разрешения.

- **Инженер** — занимается настройкой операционных процессов системы, разработкой решений для оптимизации этих процессов и их внедрением. Отвечает за создание и установку конфигураций конвейера, настройку коллектора. Действия инженера ограничены рамками определенных тенантов.
- **Пользователь** — работает с данными системы в рамках определенных тенантов, выполняет анализ и визуализацию данных для поддержки управленческих решений. Имеет ограниченный доступ к ресурсам системы.

Разрешения стандартных ролей пользователей

Разрешения в системе сгруппированы по следующим секциям:

- Системные разрешения
- Разрешения для работы с менеджером агентов
- Разрешения для работы с событиями ИБ
- Разрешения для работы с пространствами

Системные разрешения

Системные разрешения сгруппированы в разделе **Настройки** дерева разрешений. Они определяют, может ли пользователь управлять доступом других пользователей к системе, а также работать с LDAP-соединениями.



Системными разрешениями обладает только стандартная роль **Администратор**.

Группа разрешений	Разрешения
Управление доступом	• Политики пароля и входа ◦ Изменение политик пароля и входа ◦ Просмотр политик пароля и входа
	• Роли ◦ Создание ролей ◦ Удаление ролей ◦ Политики ▪ Создание политик ▪ Удаление политик ▪ Изменение политик ▪ Просмотр политик ◦ Изменение ролей ◦ Просмотр ролей
	• Тенанты ◦ Создание тенантов ◦ Изменение тенантов ◦ Просмотр тенантов
	• Группы пользователей ◦ Создание групп пользователей ◦ Удаление групп пользователей ◦ Изменение групп пользователей ◦ Просмотр групп пользователей
	• Пользователи ◦ Создание пользователей ◦ Удаление пользователей ◦ Изменение пользователей ◦ Просмотр пользователей
LDAP-соединения	• Создание LDAP-соединений • Удаление LDAP-соединений • Запуск синхронизации LDAP-соединения • Изменение LDAP-соединений • Просмотр LDAP-соединений

Разрешения для работы с менеджером агентов

Разрешения для работы с менеджером агентов сгруппированы в разделе **Агенты** дерева разрешений. Они определяют, может ли пользователь устанавливать и настраивать агенты, а также управлять работой агентов, групп агентов и шлюзов.



Разрешениями для работы с менеджером агентов обладает только стандартная роль **Администратор**.

Группа разрешений	Разрешения
Настройка конфигурации группы агентов	• Изменение конфигурации • Просмотр информации о конфигурации
Группы агентов	• Создание/дублирование группы • Удаление группы • Редактирование группы • Получение информации о группах
Агенты	• Назначение группы агенту • Удаление агента • Выполнение действия на агенте (все действия) • Удаление группы у агента • Получение информации об агентах
Шлюзы	• Создание шлюза • Удаление шлюза • Запуск шлюза • Остановка шлюза • Редактирование шлюза • Просмотр информации о шлюзах
Настройки	• Агенты

Разрешения для работы с событиями ИБ

Разрешения для работы с событиями ИБ сгруппированы в разделе **Streams** дерева разрешений. Они определяют, доступны ли для пользователя такие разделы, как **Ресурсы** и **Экспертиза**.



Всеми разрешениями для работы с событиями ИБ обладает только стандартная роль **Администратор**.

Группа разрешений	Разрешения	Инженер	Пользователь
Оповещения	Просмотр оповещений	✓	✓
	Управление оповещениями	✓	✓
Дашборды	Просмотр дашбордов	✓	✓
	Управление дашбордами	✓	✓
Экспертиза	Просмотр элементов экспертизы	✓	✓
	Управление элементами экспертизы	✓	
Отчеты	Просмотр отчетов	✓	✓
	Управление отчетами	✓	✓
Ресурсы → Активные списки	Просмотр активных списков	✓	✓
	Управление активными списками	✓	
	Управление системными активными списками		
Ресурсы → Коллекторы	Просмотр коллекторов	✓	✓
	Управление коллекторами	✓	
Ресурсы → Драйверы БД	Просмотр драйверов	✓	
	Управление драйверами	✓	
Ресурсы → Базы данных событий	Просмотр БД событий	✓	✓
	Управление БД событий	✓	
Ресурсы → Модели событий	Просмотр моделей событий	✓	✓
	Управление моделями событий	✓	
Ресурсы → Хранилища событий	Просмотр хранилищ событий	✓	✓
	Управление хранилищами событий	✓	
Ресурсы → Интеграции	Просмотр интеграций	✓	✓
	Управление интеграциями	✓	
Ресурсы → Секреты	Просмотр секретов	✓	
	Управление секретами	✓	
Поиск	Поиск по событиям	✓	✓
	Добавление записи в исключения		
Настройки сервера	Просмотр настроек сервера	✓	✓
	Изменение настроек сервера	✓	
Инструменты → Политики управления доступа к событиям	Просмотр политик управления доступом к событиям	✓	
	Управление политиками доступом к событиям	✓	

Группа разрешений	Разрешения	Инженер	Пользователь
Инструменты → Мониторинг		✓	✓
Инструменты → Аудит источников	Просмотр аудита источников	✓	✓
	Управление аудитом источников	✓	
Инструменты → Диспетчер запросов	Просмотр записей диспетчера запросов	✓	✓
	Управление записями диспетчера запросов	✓	✓
	Просмотр записей ресурсов пользовательских запросов	✓	
	Управление записями ресурсов пользовательских запросов	✓	
Инструменты → Сниппеты	Просмотр сниппетов	✓	✓
	Управление сниппетами	✓	✓
Инструменты → VRL-песочница		✓	✓
Инструменты → Покрытие MITRE ATT&CK		✓	✓

Разрешения для работы с пространствами

Разрешения для работы с пространствами сгруппированы в разделе **Пространства** дерева разрешений. Они определяют, может ли пользователь управлять пространствами и сервисами в системе.



Разрешениями для работы с пространствами обладают только стандартные роли **Администратор** и **Инженер**.

Группа разрешений	Разрешение
Пространства и сервисы	• Создание пространства • Удаление пространства • Просмотр пространства • Изменение пространства • Просмотр сервиса • Запустить сервис • Остановить сервис

Политики входа и пароля

Данный раздел описывает настройки, применяемые ко всем пользователям системы для создания и изменения учетных данных. Эти настройки управляют как аспектами безопасности пароля, так и правилами доступа к системе. При одновременном изменении политики несколькими пользователями, применятся изменения, внесенные тем пользователем, который последним сохранит изменения.

Настройки пароля и входа в систему задаются в разделе **Настройки** → **Управление доступом** → **Политики входа и пароля** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Настройка политик входа и пароля

Чтобы настроить политики входа и пароля:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Политики входа и пароля**.
2. Задайте следующие настройки:
   - Требования к паролям
   - Срок действия пароля
   - Политики входа
   - Блокирование неиспользуемой учетной записи
   - Блокирование сеанса пользователя после установленного времени бездействия
3. Чтобы сохранить внесенные изменения, нажмите на кнопку **Сохранить** в нижней части раздела.

   Чтобы отменить все несохраненные изменения, нажмите на кнопку **Сбросить** в нижней части раздела.

Требования к паролям

Если пользователь задаст пароль, не соответствующий установленным требованиям, он увидит соответствующее предупреждение. Также, при изменении требований к паролю, пользователь увидит предупреждение о том, что существующий пароль не соответствует новой парольной политике, и сможет изменить пароль в соответствии с ней.

Для включения политики надежных паролей:

1. Установите переключатель **Требовать надежные пароли** в активное положение.
2. Задайте минимальную длину пользовательского пароля и установите флажки напротив требуемых условий для создания пароля.
3. Нажмите на кнопку **Сохранить** в нижней части раздела. Система применит изменения и отобразит соответствующее уведомление.

Политика надежных паролей включает следующие требования:

- **Минимальная длина пароля**: задает минимально допустимое количество символов в пароле.
- **Должен содержать**: задает классы символов, которые должен содержать пароль. Можно включить в требования следующие классы:
  - строчные латинские буквы (a–z);
  - заглавные латинские буквы (A–Z);

- цифры (0–9);
- специальные символы:

```
~"!@#$%^&*_-+=`(){}[]:;'<>,.?/|\
```

Кроме того, можно включить ограничения на содержание пароля:

- запрет на последовательное повторение символов;
- запрет на совпадение пароля с текущим или предыдущим.

Срок действия пароля

Срок действия пароля определяет, на какой срок устанавливается пароль, прежде чем его необходимо будет изменить. Допустимые значения: от 7 до 365 дней. Если переключатель **Срок действия пароля** выключен, пароль действует бессрочно.

Чтобы установить срок действия пароля:

1. Установите переключатель **Срок действия пароля** в активное положение.
2. Задайте срок действия пароля вручную или с помощью кнопок — и +. Срок действия задается в днях.
3. Нажмите на кнопку **Сохранить** в нижней части раздела. Система применит изменения и отобразит соответствующее уведомление.

   За 7 дней до истечения срока действия система отобразит уведомление о необходимости смены пароля.

Политики входа

Чтобы включить блокировку учетных записей после неправильного ввода пароля:

1. Установите переключатель **Политики входа** в активное положение.
2. Настройте блокировку учетной записи при неудачных попытках ввода пароля. Блокировка происходит, если количество неудачных попыток ввода пароля превышает заданный порог в течение указанного периода.
   - **Период ввода попыток** — период времени в минутах, в течение которого подсчитываются неудачные попытки входа. Допустимые значения: от 1 до 60 минут.
   - **Количество попыток до блокировки** — максимальное количество неудачных попыток входа. Допустимые значения: от 3 до 10 попыток.



Пример

Если **Период ввода попыток** равен 5 минутам и **Количество попыток до блокировки** равно 3, то учетная запись пользователя будет заблокирована после 3 неудачных попыток входа, совершенных за 5 минут. При этом если пользователь не достигнет лимита неудачных попыток и подождет более 5 минут после последней попытки, ему снова будут доступны 3 попытки.

- **Блокировать** — длительность блокировки учетной записи. Допустимые варианты: **Бессрочно** или **На время**.
- **Период блокировки** — время в минутах, на которое учетная запись будет заблокирована после превышения допустимого количества попыток входа. Допустимые значения: от 3 до 60 минут. Настройка доступна для редактирования если в поле **Блокировать** выбран вариант **На время**.

Блокирование неиспользуемой учетной записи

Чтобы включить блокирование неиспользуемых учетных записей:

1. Установите переключатель **Блокирование неиспользуемой учетной записи** в активное положение.
2. Выберите из выпадающего списка единицы времени: часы, дни, месяцы.
3. Задайте период неактивности.

Период неактивности позволяет указать промежуток времени с последнего входа в систему, по истечении которого неактивная учетная запись будет автоматически заблокирована.



Минимальный возможный период неактивности зависит от времени жизни токена обновления (refresh token) и может быть изменен с помощью переменной окружения `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME`.

Блокирование сеанса пользователя после установленного времени бездействия

Период неактивности устанавливает время в минутах, после которого неактивный сеанс пользователя будет автоматически завершен.



Минимальный возможный период неактивности зависит от времени жизни токена доступа (access token) и может быть изменен с помощью переменной окружения `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME`.

Настройка времени жизни для токенов доступа

В системе предусмотрены два вида токенов доступа:

- **Access token** — токен, выдаваемый пользователю при входе в систему. Имеет короткое время жизни.
- **Refresh token** — токен, предназначенный для периодического обновления пары **access/refresh token**. Имеет более продолжительное время жизни, чем **Access token**.

Чтобы изменить время жизни токенов доступа пользователей:

Через командную строку

1. Подключитесь к кластеру.
2. Измените время хранения токенов в ConfigMap `evo.core-access-env`, выполнив команду:

```
kubectl patch configmap -n <namespace> evo.core-access-env -p '{
  "data": {
    "EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME": "<access_lifetime>",
    "EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME":
"<refresh_lifetime>",
    "SINGLE_SESSION": <single_session_value>
  }
}'
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

- `<access_lifetime>` — временной интервал жизни токена **Access token**. Например: `30m`, `1h` или `2d`.
- `<refresh_lifetime>` — временной интервал жизни токена **Refresh token**. Например: `30m`, `1h` или `2d`.
- `<single_session_value>` — логическая переменная, определяющая количество одновременных сеансов:
  - Значение `true` — для одной учетной записи в системе разрешен только один одновременный сеанс.
  - Значение `false` — количество одновременных сеансов не ограничено.



- Значение `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME` должно быть меньше, чем период неактивности для блокирования сеанса пользователя из-за бездействия.
- Значение `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME` должно быть меньше, чем период неактивности для блокирования неиспользуемой учетной записи.
- Значение `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME` должно быть строго больше значения `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME`.

3. Перезапустите контроллер Deployment `evo.core.access`, выполнив команду:

```
kubectl -n <namespace> rollout restart deployment evo.core.access
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Через Lens

1. Подключитесь к кластеру.
2. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
3. Выберите ConfigMap `evo.core-access-env` в нужном пространстве имен. В правой части экрана отобразится панель настроек конфигурации.
4. В разделе **Data** панели настроек измените значения следующих переменных окружения:
   - `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME` — время жизни **Access token**.

     Период неактивности для блокирования сеанса пользователя из-за бездействия должен быть не меньше значения `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME`.

   - `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME` — время жизни **Refresh token**.

     Период неактивности для блокирования неиспользуемой учетной записи должен быть не меньше значения `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME`.

     Значение `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME` должно быть строго больше значения `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME`.

   - `SINGLE_SESSION` — логическая переменная:
     - если `SINGLE_SESSION` имеет значение `true`, то для одной учетной записи в системе разрешен только один одновременный сеанс;
     - если `SINGLE_SESSION` имеет значение `false`, то количество одновременных сеансов не ограничено.
5. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.
6. Выберите контроллер Deployment `evo.core.access` в таблице. В правой части экрана откроется панель свойств контроллера.
7. Чтобы применить настройки, перезагрузите контроллер, нажав на кнопку **Restart** на панели свойств.

LDAP-соединения

Данный раздел описывает процесс работы с LDAP-соединениями в системе. Работа осуществляется в разделе **Настройки** → **LDAP-соединения** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О LDAP-соединениях

LDAP-соединения используются для интеграции системы со службами каталогов, такими как Microsoft Active Directory, ALD Pro, FreeIPA, OpenLDAP. Использование LDAP-соединений позволяет управлять учетными записями и доступом пользователей к ресурсам системы.

LDAP (Lightweight Directory Access Protocol) позволяет централизованно управлять доступом пользователей и их правами в системе. В системе LDAP-соединения обеспечивают:

- аутентификацию пользователей с использованием внешних служб каталогов;
- синхронизацию данных пользователей, включая логин, ФИО, должность, статус, электронный адрес и номер телефона;
- управление ролями и разрешениями пользователей, импортированных из внешних каталогов.

Поддерживаются следующие типы каталогов:

- **Active Directory** — служба каталогов Microsoft Active Directory.
- **ALD Pro** — служба каталогов Astra Linux.



Для работы с ALD Pro требуется наличие сервисной учетной записи. Для ее настройки обратитесь к разделу Настройка сервисной учетной записи ALD Pro.

- **FreeIPA** — система управления идентификацией и аутентификацией для Linux.
- **OpenLDAP** — реализация протокола LDAP с открытым исходным кодом.



Система поддерживает сбор событий из вышеприведенных служб каталогов с помощью точки входа Syslog.

Сбор событий из ALD Pro и FreeIPA настраивается по одному и тому же алгоритму и описан в разделе FreeIPA: настройка источника руководства по настройке источников.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Создать** (+) позволяет добавить новое LDAP-соединение.
- Кнопка **Удалить** () позволяет удалить выбранное LDAP-соединение.
- Поле **Поиск** предназначено для быстрого поиска LDAP-соединений в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся в системе LDAP-соединений, которые доступны пользователю. Таблица содержит следующие столбцы:

- **Название** — имя LDAP-соединения.
- **Статус** — текущий статус соединения: **Включено** или **Нет соединения**.
- **Домен** — домен, к которому подключено соединение.

- **Тип** — тип каталога (например, Active Directory).
- **Описание** — краткое описание соединения.
- **Дата создания** — дата и время создания соединения.
- **Дата изменения** — дата и время последнего изменения соединения.

При работе с таблицей доступны следующие операции:

- Поиск LDAP-соединений по полям **Название** и **Домен**.
- Сортировка LDAP-соединений по полям **Название**, **Домен**, **Дата создания** и **Дата изменения**.

При выборе конкретного соединения в правой части рабочей области отображается его карточка с детальной информацией.

Работа с LDAP-соединением

Доступные операции над LDAP-соединением:

- Создание LDAP-соединения
- Просмотр LDAP-соединения
- Синхронизация LDAP-соединения
- Изменение LDAP-соединения
- Удаление LDAP-соединения

Создание LDAP-соединения

Вы можете создать новое LDAP-соединение для интеграции со службами каталогов, такими как Microsoft Active Directory, ALD Pro, FreeIPA, OpenLDAP.

Чтобы создать новое LDAP-соединение:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения об имеющихся соединениях.
2. Нажмите на кнопку **Создать** (＋). Отобразится окно добавления LDAP-соединения.
3. Заполните следующие поля:
    - **Название** — введите название LDAP-соединения.
    - **Тип** — выберите тип каталога из выпадающего списка:
        - **Active Directory** — служба каталогов Microsoft Active Directory.
        - **ALD Pro** — служба каталогов Astra Linux.



Для работы с ALD Pro требуется наличие сервисной учетной записи. Для ее настройки обратитесь к разделу Настройка сервисной учетной записи ALD Pro.

        - **FreeIPA** — система управления идентификацией и аутентификацией для Linux.
        - **OpenLDAP** — реализация протокола LDAP с открытым исходным кодом.

- **Домен** — введите домен LDAP-соединения. Например: `example.com`.
- **Адрес LDAP-сервера** — введите IP-адрес или полное доменное имя (FQDN) LDAP-сервера. Например: ldap.example.com или 192.168.1.1.
- **Порт** — введите номер порта. По умолчанию используются 389 или 636 (при использовании SSL).
- **Использовать SSL** — установите этот флажок, если необходимо использовать SSL для защищенного соединения.
- **Имя пользователя** — введите имя пользователя для подключения к LDAP-серверу. Формат имени пользователя зависит от используемой службы каталогов:
  - **Active Directory**: имя пользователя в формате `login@domain`. Например: `admin@example.com`.
  - **ALD Pro**, **OpenLDAP**, **FreeIPA**: имя пользователя в формате отличительного имени (Distinguished Name). Например:

```
UID=admin,CN=users,CN=accounts,DC=ipa,DC=company,DC=security
```



Примеры атрибутов отличительного имени приведены в разделе Атрибуты Distinguished Name.

- **Пароль** — введите пароль пользователя.
- **Базовый DN** — введите базовое отличительное имя (Distinguished Name) для поиска пользователей и групп. Например:

```
OU=users,DC=mycompany,DC=com
```



Примеры атрибутов отличительного имени приведены в разделе Атрибуты Distinguished Name.

- **Описание** (опционально) — введите описание соединения.

▼ *Пример настроек LDAP-соединения*

Название	Пример настройки
Тип	Active Directory
Домен	example.com
Адрес LDAP сервера	192.168.1.1
Порт	636
Использовать SSL	[✓]
Имя пользователя	admin@example.com
Пароль	password123
Базовый DN	dc=example,dc=com
Описание	Соединение для тестового домена

4. Нажмите на кнопку **Проверить подключение** (🔄). Система проверит настройки подключения к выбранной службе каталогов и отобразит статус подключения: **Подключение успешно** или **Ошибка подключения**.
5. Нажмите на кнопку **Создать**. Система создаст LDAP-соединение и отобразит соответствующее уведомление. Новое соединение отобразится в списке раздела.

Просмотр LDAP-соединения

Чтобы просмотреть LDAP-соединение:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения об имеющихся соединениях.
2. Нажмите на строку соединения в списке. Система отобразит в правой части экрана карточку этого соединения с подробной информацией о нем.

В верхней части карточки отображается название LDAP-соединения. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранного соединения.

Карточка содержит основные сведения о LDAP-соединении:

- **Название** — имя LDAP-соединения.
- **Статус** — текущий статус соединения: **Включен** или **Нет соединения**.
- **Тип** — тип каталога (например, Active Directory).
- **Домен** — домен, к которому подключено соединение.
- **Адрес LDAP-сервера** — адрес сервера LDAP.
- **Порт** — номер порта для подключения к серверу LDAP.
- **SSL** — информация о том, используется ли SSL для соединения.
- **Имя пользователя** — имя пользователя для подключения к серверу LDAP.
- **Базовый DN** — базовый DN для поиска пользователей и групп.
- **Описание** — краткое описание соединения.
- **Дата создания** — дата и время создания соединения.
- **Создал** — пользователь, инициировавший создание соединения.
- **Дата изменения** — дата и время последнего изменения соединения.
- **Изменил** — пользователь, выполнивший последнее изменение соединения.
- **Синхронизировано** — информация о последней синхронизации данных с LDAP-сервером.

В нижней части карточки также расположена кнопка **Синхронизировать** (🔄), позволяющая обновить данные пользователей из LDAP-сервера.

Синхронизация LDAP-соединения

Синхронизация обновляет данные пользователей из LDAP-сервера в системе.

Чтобы синхронизировать данные:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения об имеющихся соединениях.
2. Нажмите на строку соединения в списке. Система отобразит в правой части экрана карточку этого соединения с подробной информацией о нем.
3. Нажмите на кнопку **Синхронизировать** (🔄) в нижней части карточки. Система выполнит синхронизацию данных с сервером LDAP и отобразит уведомление о статусе синхронизации.

Изменение LDAP-соединения

Вы можете изменить LDAP-соединение через меню действий в его карточке.

Удаление LDAP-соединения

Вы можете удалить LDAP-соединение следующими способами:

- через меню действий в его карточке;
- через массовое удаление LDAP-соединений.



При удалении LDAP-соединения все учетные записи пользователей, связанные с этим доменом, также будут удалены.

Настройка сервисной учетной записи ALD Pro

Аутентификация и авторизация в ALD Pro выполняются методом LDAP Bind, для чего необходимо создать сервисную учетную запись, которая:

- не является POSIX-пользователем;
- не имеет прав на вход в домен;
- не отображается в портале управления;
- используется только для чтения LDAP.

Наличие такой сервисной учетной записи необходимо для возможности подключения по LDAP-протоколу к службе каталогов ALD Pro.



В данной инструкции приведена настройка сервисной учетной записи для ALD Pro версий 2.4 и 3.0.

Для создания сервисной учетной записи подключитесь по SSH к контроллеру домена ALD Pro и выполните следующие шаги:

1. Создайте файл `ldap-bind.update` со следующим содержимым:

```
dn: uid=ldap-bind,cn=sysaccounts,<cn_attributes>,<dc_attributes>
add:objectclass: account
add:objectclass: simplesecurityobject
add:uid: ldap-bind
add:userPassword: <password>
add:passwordExpirationTime: 20380119031407Z
```

```
add:nsIdleTimeout: 0
```

Здесь:

- `<cn_attributes>` — дополнительные атрибуты общего имени (Common Name). Например: `cn=etc`.
- `<dc_attrbitues>` — атрибуты компонента доменного имени (Domain Component). Например: `dc=ald,dc=company,dc=lan`.
- `<password>` — пароль для учетной записи.



Более подробная информация об атрибутах приведена в разделе Атрибуты Distinguished Name.

▼ *Описание параметров*

- `dn` — уникальный идентификатор записи пользователя в LDAP.
- `add:objectclass: account` — базовый класс для учетной записи.
- `add:objectclass: simplesecurityobject` — класс для хранения пароля и других атрибутов безопасности.
- `add:uid: ldap-bind` — уникальный идентификатор пользователя.
- `add:userPassword` — пароль для учетной записи.
- `add:passwordExpirationTime` — время истечения пароля. При значении `203801190314072` срок действия пароля неограничен.
- `add:nsIdleTimeout: 0` — отключение таймаута простоя для учетной записи.

2. Добавьте пользователя, выполнив следующую команду:

```
kinit admin && ipa-ldap-updater ldap-bind.update
```

Атрибуты Distinguished Name

В данном разделе представлены примеры атрибутов отличительного имени (Distinguished Name), которые можно использовать при создании LDAP-соединения.

- Атрибуты в LDAP
- Атрибуты в Active Directory

Атрибуты в LDAP

Атрибут	Англоязычное название	Русскоязычное название	Пример
`uid`	User ID	Уникальный идентификатор пользователя	`ldap-bind`
`cn`	Common Name	Общее имя	`cn=sysaccounts`

Атрибут	Англоязычное название	Русскоязычное название	Пример
dc	Domain Component	Компонент доменного имени	company
ou	Organizational Unit	Подразделение	corp
o	Organization	Организация	org
c	Country	Страна расположения организации	Russia
l	Locality	Населенный пункт расположения организации	Moscow

Атрибуты в Active Directory

OU (Organizational Unit) — Подразделение

Атрибут	Англоязычное название	Русскоязычное название	Пример
distinguishedName	Distinguished Name	Отличительное (уникальное) имя	OU=Компания, DC=domain, DC=com
name			Компания

Group — Группа

Атрибут	Англоязычное название	Русскоязычное название	Пример
distinguishedName	Distinguished Name	Отличительное (уникальное) имя	CN=Группа, OU=Компания, DC=domain, DC=com
name			Группа
member	Members	Члены группы (какие пользователи входят в данную группу)	CN=Сергей Петрович Иванов, OU=Компания, DC=domain, DC=com

User — Пользователь

Атрибут	Англоязычное название	Русскоязычное название	Пример
DN	Distinguished Name	Отличительное (уникальное) имя	CN=NW_Admins, OU=Corp, DC=domain, DC=com
DC	Domain Component	Компонент (класс) доменного имени	DC=domain, DC=com
OU	Organizational Unit	Подразделение	Corp
O	Organization	Организация	Org
CN	Common Name	Общее имя	NW_Admins
givenName	First Name	Имя	Сергей Петрович

Атрибут	Англоязычное название	Русскоязычное название	Пример
sn (SurName)	Last Name	Фамилия	Иванов
name	Full Name	Полное имя	Сергей Петрович Иванов
displayName	Display Name	Отображаемое имя	Сергей Петрович Иванов
mail	Email	Электронная почта	sivanov@domain.com
sAMAccountName	User Logon Name (pre-Windows 2000)	Имя входа пользователя (до Windows 2000)	IvanovSP
userPrincipalName	User Logon Name	Имя входа пользователя	IvanovSP@domain.com
memberOf	Member Of	Членство в группах	CN=Группа, OU=Компания, DC=domain, DC=com
AccountPassword	Password	Пароль	

Политики управления доступом к событиям

Данный раздел описывает процесс настройки политик доступа пользователей к событиям в хранилищах на основе их ролей. Работа осуществляется в разделе **Инструменты** → **Политики управления доступом к событиям** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О политиках управления доступом к событиям

Политика управления доступом к событиям — это правило, на основе которого пользователи, имеющие заданные роли, будут иметь доступ только к определенным событиям в хранилище. Правило формулируется как условие фильтрации на языке RQL, которое будет неявно добавляться в секцию WHERE поисковых запросов пользователей.

Составление условия фильтрации из правил

Политики, распространяющиеся на роли текущего пользователя, добавляются к запросу согласно следующей логике:

- Правила внутри одной политики соединяются оператором OR.
- Политики соединяются между собой оператором AND.
- Все политики соединяются с запросом оператором AND.

Рассмотрим следующий пример:

- Пользователь ввел следующий запрос в разделе **Поиск**:

```
dvendor = 'Linux'
```

- Первая политика задает для этого пользователя следующие правила доступа:

```
dproduct = 'R-Vision'
at = 'Endpoint'
```

- Вторая политика задает следующее правило доступа:

```
app = 'HTTPS'
```

Запрос, отображаемый в строке поиска, и запрос, который будет в действительности выполнен, соответствуют друг другу следующим образом:

Запрос в строке поиска	Запрос, который будет выполнен
`dvendor = 'Linux'`	`dvendor = 'Linux' AND ((dproduct = 'R-Vision' OR at = 'Endpoint') AND app = 'HTTPS')`

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую политику.
- Кнопка **Удалить** (🗑) позволяет удалить выбранные политики.
- Поле **Поиск** предназначено для быстрого поиска политик в отображаемом списке по названию.
- Кнопка **Фильтр** (▽) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка политик.

В рабочей области отображается таблица существующих политик. Таблица содержит следующие столбцы:

- **Название** — название политики, используемое для ее идентификации.
- **Роли** — список ролей, на которые распространяется политика.
- **Описание** — краткое описание политики.
- **Хранилища** — список хранилищ событий, на которые распространяется политика.
- **Дата создания** — дата и время создания политики.
- **Дата изменения** — дата и время последнего изменения политики.

При работе с таблицей политик доступны следующие операции:

- Поиск политик по полю **Название**.

- Фильтрация политик по полям **Роли** и **Хранилища**. Роли и хранилища выбираются из выпадающих списков.
- Сортировка политик по полям **Название**, **Описание**, **Дата создания**, **Дата изменения**.
- Настройка отображения таблицы политик.

При выборе конкретной политики в правой части рабочей области отображается ее карточка с детальной информацией.

Работа с политикой управления доступом к событиям

Доступные операции над политикой:

- Создание политики управления доступом к событиям
- Просмотр политики управления доступом к событиям
- Изменение политики управления доступом к событиям
- Удаление политики управления доступом к событиям

Создание политики управления доступом к событиям

Чтобы создать политику:

1. Перейдите в раздел **Инструменты** → **Политики управления доступом к событиям**. Система отобразит сведения об имеющихся политиках.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания политики.
3. Введите название политики.
4. При необходимости укажите описание политики.
5. В поле **Роль** выберите роли, на которые будет распространяться политика.
6. Добавьте правила доступа к хранилищам событий.

   Чтобы добавить правило:

   a. Нажмите на кнопку **Добавить правило**. Отобразится панель настроек нового правила.

   По умолчанию в окне уже отображается панель настройки для первого правила.

   b. В поле **Хранилище** выберите хранилища событий, на которые будет распространяться правило.

   c. В поле **Правило** введите условие на языке RQL, которое будет неявно добавляться к поисковым запросам пользователей.

   Например, чтобы разрешить пользователям просматривать только события, поступившие с устройств на ОС Linux из определенной подсети, вы можете задать следующее правило:

```
dvendor = 'Linux' AND sourceIP IN '203.0.113.0/24'
```

Чтобы удалить заданное правило, нажмите на кнопку в правом верхнем углу его панели настроек.

7. Нажмите на кнопку **Создать**. Система создаст политику и отобразит уведомление об успешном создании. Новая политика появится в списке раздела **Инструменты → Политики управления доступом к событиям**.

Просмотр политики управления доступом к событиям

Чтобы просмотреть политику:

1. Перейдите в раздел **Инструменты → Политики управления доступом к событиям**. Система отобразит сведения об имеющихся политиках.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.

В верхней части карточки отображается название политики. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранной политики.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка отображает следующую информацию о политике:

- **ID** — уникальный идентификационный код политики.
- **Название** — системное имя политики, используемое для ее идентификации.
- **Описание** — краткое описание политики.
- **ID тенанта** — идентификатор тенанта, в котором была создана политика.
- **Роли** — список ролей, на которые распространяется политика.
- **Дата создания** — дата и время создания политики.
- **Создал** — пользователь, инициировавший создание политики.
- **Дата изменения** — дата и время последнего изменения политики.
- **Изменил** — пользователь, выполнивший последнее изменение политики.
- **Правила доступа** — список правил, созданных в рамках политики. Для каждого правила указываются хранилища событий, на которые оно распространяется, и условие фильтрации на языке RQL.

Изменение политики управления доступом к событиям

Вы можете изменить политику управления доступом к событиям следующими способами:

- через меню действий в карточке политики;
- через изменение полей в карточке политики.

Удаление политики управления доступом к событиям

Вы можете удалить политику управления доступом к событиям следующими способами:

- через меню действий в карточке политики;
- через массовое удаление политик.

Настройка агентов

Агент обеспечивает мониторинг узла, на котором он установлен, и передает данные в режиме реального времени менеджеру через зашифрованный канал.

Основная информация о настройке агентов через веб-интерфейс системы приведена в разделе Настройки менеджера агентов.

Сведения о собираемой агентами информации на хостах приведена в следующих разделах:

- Сбор инвентарной информации
- Сбор событий ИБ на агентах

Настройки менеджера агентов

Данный раздел описывает настройки, применяемые ко всем агентам в системе. Эти настройки позволяют управлять параметрами работы агентов.

Настройки агентов задаются в разделе **Настройки** → **Агенты** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Настройки сгруппированы по следующим секциям:

- Параметры работы агентов
- Белый список IP-адресов с доступом к самоизолированным агентам

Параметры работы агентов

В данной секции вы можете настроить следующие параметры работы для агентов:

- **Максимальное время отсутствия активности от агента** — время в секундах, по истечении которого неактивный агент отключается и переходит в статус **Офлайн**.

  Значение по умолчанию — 30 секунд.

- **Максимальное время жизни команды/сообщения для агента** — количество дней, по истечении которого отправленная команда, не получившая ответа от агента, удаляется из очереди.

  Значение по умолчанию — 7 дней.

Белый список IP-адресов с доступом к самоизолированным агентам

В данной секции вы можете добавить в белый список IP-адреса, с которых разрешается доступ к агенту при выполнении команды самоизоляции.

Чтобы добавить запись в белый список:

1. Нажмите на кнопку **Добавить IP-адрес** (+). Система отобразит окно добавления IP-адреса в белый список.
2. Укажите в строке название узла и его IP-адрес.
3. Нажмите на кнопку **Добавить**. Запись будет добавлена в список.

Чтобы удалить запись из белого списка:

1. Наведите курсор на запись. Справа от записи отобразится кнопка .
2. Нажмите на кнопку . Отобразится окно подтверждения удаления записи.
3. Нажмите на кнопку **Удалить**. Строка с записью будет удалена.

Сбор инвентарной информации

В данном разделе представлены сведения о собираемой агентами инвентарной информации и структуре соответствующих сообщений, передаваемых в менеджер.

Об инвентарной информации

Агент позволяет собирать следующие инвентарные сведения о хосте:

- Аппаратная конфигурация: материнская плата, процессоры, память, диски, дисплеи, видеокарты.
- Процессы: состояние и идентификаторы процессов, использование ресурсов.
- Пользователи и группы.
- Сетевые интерфейсы и открытые порты.
- Общие папки и точки монтирования.
- Установленные пакеты и приложения.
- Docker-контейнеры.

Сбор инвентарной информации необходим для:

- отображения конфигурации хоста в веб-интерфейсе системы;
- использования конфигурации хоста для автоматического назначения групп агентам;
- регистрации агента в менеджере.

Сбор инвентарной информации осуществляется следующим образом:

- При запуске или перезапуске агента.
- При запросе от менеджера агентов.
- По расписанию, заданному в секции `inventory` файла конфигурации `rpoint.conf`.

Структура сообщения об инвентарной информации

Ниже представлена структура сообщения с инвентарными сведениями, которое агент отправляет в менеджер. Данную структуру необходимо учитывать при настройке автоназначения групп.

Параметр	Тип данных	Описание
os_info	объект OSInfo	Информация об ОС. Необязательный параметр.
hardware	объект Hardware	Информация об аппаратном обеспечении. Необязательный параметр.
network_interfaces	объект NetworkInterfaces	Информация о сетевых интерфейсах. Необязательный параметр.
processes	объект Processes	Информация о процессах. Необязательный параметр.
users_and_groups	объект UsersAndGroups	Информация о пользователях и группах. Необязательный параметр.
mounts	объект Mounts	Информация о точках монтирования. Необязательный параметр.
docker_containers	объект DockerContainers	Информация о контейнерах. Необязательный параметр.
packages	объект Packages	Информация о пакетах. Необязательный параметр.
shared_folders	объект SharedFolders	Информация об общих папках. Необязательный параметр.
open_ports	объект OpenPorts	Информация об открытых портах. Необязательный параметр.
created_at	int64	Время формирования сообщения в миллисекундах, прошедших с момента 00:00:00 UTC 1 января 1970.

OSInfo

Информация об операционной системе.

Параметр	Тип данных	Описание
os_name	string	Название ОС.

Параметр	Тип данных	Описание
os_platform	string	Платформа ОС.
os_version	string	Версия ОС.
os_major	int32	Мажорная версия ОС.
os_minor	int32	Минорная версия ОС.
os_patch	int32	Номер патча ОС.
os_build	string	Номер сборки ОС.
os_type	объект OsType	Тип ОС: • OS_TYPE_UNKNOWN — неизвестная ОС. • OS_TYPE_LINUX — Linux. • OS_TYPE_WINDOWS — Windows. • OS_TYPE_MACOS — macOS.
os_codename	string	Кодовое имя ОС.
sysname	string	Системное имя.
hostname	string	Имя хоста.
version	string	Версия ядра ОС.
architecture	string	Архитектура ОС.
release	string	Номер релиза ОС.
machine_id	string	Уникальный идентификатор устройства.
os_release	string	Подробная информация о релизе ОС.
os_display_version	string	Отображаемая версия ОС.

Hardware

Информация об аппаратном обеспечении.

Параметр	Тип данных	Описание
motherboard	объект Motherboard	Информация о материнской плате.
cpu	массив объектов Cpu	Список процессоров. Необязательный параметр.
memory	объект Memory	Информация об оперативной памяти.
displays	массив объектов Display	Список дисплеев. Необязательный параметр.

Параметр	Тип данных	Описание
`disks`	массив объектов Disk	Список дисков. Необязательный параметр.
`videos`	массив объектов Video	Список видеокарт. Необязательный параметр.

Motherboard

Информация о материнской плате.

Параметр	Тип данных	Описание
`serial_number`	string	Серийный номер материнской платы.
`manufacturer`	string	Производитель материнской платы.
`product`	string	Модель материнской платы.

Cpu

Информация о процессоре.

Параметр	Тип данных	Описание
`name`	string	Имя процессора.
`num_cores`	uint32	Количество ядер в процессоре.
`mhz`	double	Тактовая частота процессора в МГц.

Memory

Информация об оперативной памяти.

Параметр	Тип данных	Описание
`total_bytes`	uint64	Общий объем оперативной памяти в байтах.
`free_bytes`	uint64	Свободный объем оперативной памяти в байтах.
`modules`	массив объектов Module	Список модулей оперативной памяти. Необязательный параметр.

Module

Информация о модуле оперативной памяти.

Параметр	Тип данных	Описание
`serial_number`	string	Серийный номер модуля оперативной памяти.
`manufacturer`	string	Производитель модуля оперативной памяти.

Параметр	Тип данных	Описание
`size_bytes`	uint64	Размер модуля оперативной памяти в байтах.

Display

Информация о дисплее.

Параметр	Тип данных	Описание
`serial_number`	string	Серийный номер дисплея.
`model`	string	Модель дисплея.

Disk

Информация о диске.

Параметр	Тип данных	Описание
`serial_number`	string	Серийный номер диска.
`model`	string	Модель диска.
`size_bytes`	uint64	Размер диска в байтах.

Video

Информация о видеокарте.

Параметр	Тип данных	Описание
`serial_number`	string	Серийный номер видеокарты.
`manufacturer`	string	Производитель видеокарты.
`model`	string	Модель видеокарты.

NetworkInterfaces

Информация о сетевых интерфейсах.

Параметр	Тип данных	Описание
`ipv4`	массив объектов Ip	Список IPv4-адресов. Необязательный параметр.
`ipv6`	массив объектов Ip	Список IPv6-адресов. Необязательный параметр.
`adapter`	string	Название сетевого адаптера. Необязательный параметр.

Параметр	Тип данных	Описание
`type`	объект NetworkInterfaceType	Тип сетевого интерфейса: • `NIT_UNSPECIFIED` — тип не указан. • `NIT_ETHERNET` — Ethernet. • `NIT_TOKEN_RING` — Token Ring. • `NIT_POINT_TO_POINT` — Point-to-Point. • `NIT_ATM` — ATM. • `NIT_FIREWIRE` — Firewire. • `NIT_TUNNEL` — Tunnel. • `NIT_FIBRECHANNEL` — FibreChannel. • `NIT_WIRELESS` — Wireless. • `NIT_NONE` — интерфейс отсутствует.
`state`	объект NetworkInterfaceState	Состояние сетевого интерфейса: • `NIS_UNSPECIFIED` — состояние не указано. • `NIS_UP` — интерфейс активен. • `NIS_DOWN` — интерфейс выключен. • `NIS_TESTING` — интерфейс находится в состоянии тестирования. • `NIS_DORMANT` — интерфейс находится в спящем режиме. • `NIS_NOT_PRESENT` — интерфейс отсутствует. • `NIS_LOWER_LAYER_DOWN` — нижний уровень интерфейса не активен.
`mac`	string	MAC-адрес. Необязательный параметр.
`tx_packets`	int64	Количество отправленных пакетов. Необязательный параметр.
`rx_packets`	int64	Количество полученных пакетов. Необязательный параметр.
`tx_bytes`	int64	Количество отправленных байтов. Необязательный параметр.
`rx_bytes`	int64	Количество полученных байтов. Необязательный параметр.

Параметр	Тип данных	Описание
`tx_errors`	int64	Количество ошибок передачи, обнаруженных драйвером устройства. Необязательный параметр.
`rx_errors`	int64	Количество ошибок приема, обнаруженных драйвером устройства. Необязательный параметр.
`tx_dropped`	int64	Количество сброшенных пакетов при передаче. Необязательный параметр.
`rx_dropped`	int64	Количество сброшенных пакетов при приеме. Необязательный параметр.
`mtu`	uint32	Максимальный допустимый размер передаваемого пакета (MTU) в байтах. Необязательный параметр.
`gateway`	string	Информация о шлюзе. Необязательный параметр.

Ip

Информация об IP-адресе.

Параметр	Тип данных	Описание
`address`	string	IP-адрес. Примеры: `192.168.1.1`, `::ffff:192.168.1.1`
`netmask`	string	Маска. Примеры: `255.255.0.0`, `ffff:ffff:ffff:ffff::`
`broadcast`	string	Broadcast-адрес. Пример: `192.168.1.255`
`metric`	int32	Значение метрики маршрута. Необязательный параметр.
`dhcp`	объект NetworkInterfaceDhcp	Протокол DHCP: • `NID_UNSPECIFIED` — не указан. • `NID_ENABLED` — включен. • `NID_DISABLED` — выключен. • `NID_BOOTP` — Bootstrap Protocol.

Processes

Информация о процессах.

Параметр	Тип данных	Описание
`values`	массив объектов Process	Список процессов. Необязательный параметр.

Process

Информация о процессе.

Параметр	Тип данных	Описание
`process_id`	int64	Идентификатор процесса.
`name`	string	Имя процесса. Пример: `system`, `conhost.exe`
`state`	объект ProcessState	Состояние процесса: • `PS_UNSPECIFIED` — состояние не указано. • `PS_RUNNING` — процесс выполняется. • `PS_UNINTERRUPTIBLE_SLEEP` — процесс находится в непрерываемом сне. • `PS_INTERRUPTABLE_SLEEP` — процесс находится в ожидании события. • `PS_STOPPED` — процесс остановлен. • `PS_ZOMBIE` — процесс завершил выполнение, но его родительский процесс еще не считал статус завершения (процесс-зомби). • `PS_IDLE` — процесс находится в состоянии простоя (Idle). • `PS_PARKED` — процесс остановлен и ожидает возобновления или завершения. • `PS_TRACING_STOP` — процесс остановлен отладчиком. • `PS_DEAD` — процесс завершен. Параметр обязателен только в ОС Linux.
`parent_process_id`	int64	Идентификатор родительского процесса.
`user_mode_cpu_time_ms`	int64	Время выполнения процесса в пользовательском режиме (user mode) в миллисекундах. Необязательный параметр.

Параметр	Тип данных	Описание
kernel_mode_cpu_time_ms	int64	Время выполнения процесса в режиме ядра (kernel mode) в миллисекундах. Необязательный параметр.
command	string	Команда запуска процесса.
command_arguments	массив string	Аргументы команды запуска процесса. Необязательный параметр.
effective_user_name	string	Имя пользователя, права которого используются процессом (эффективный пользователь). Необязательный параметр.
real_user_name	string	Имя пользователя, запустившего процесс (реальный пользователь). Необязательный параметр.
saved_user_name	string	Имя пользователя для восстановления исходных прав доступа. Необязательный параметр.
effective_group_name	string	Группа пользователей, права которой используются процессом (эффективная группа). Необязательный параметр.
real_group_name	string	Группа пользователей, запустившая процесс (реальная группа). Необязательный параметр.
saved_group_name	string	Группа пользователей для восстановления исходных прав доступа. Необязательный параметр.
filesystem_group_name	string	Файловая группа, к которой обращается процесс. Необязательный параметр.
priority	int32	Системный приоритет процесса.
nice	int32	Пользовательский приоритет процесса. Необязательный параметр.
total_pages	int64	Объем выделенной памяти в страницах.
virtual_memory_size_kb	int64	Размер виртуальной памяти в КБ.
resident_pages	int64	Количество страниц, загруженных в память. Необязательный параметр.

Параметр	Тип данных	Описание
shared_pages	int64	Количество страниц с общим доступом. Необязательный параметр.
start_time_ms	int64	Время запуска процесса в миллисекундах, прошедших с момента 00:00:00 UTC 1 января 1970.
process_group_id	int64	Идентификатор группы процессов. Необязательный параметр.
session_id	int64	Идентификатор сессии.
task_group_id	int64	Идентификатор группы, к которой принадлежит процесс. Необязательный параметр.
tty_device_number	int32	Номер терминального устройства процесса. Необязательный параметр.
current_cpu_num	int32	Номер ядра процессора, в котором выполняется процесс. Необязательный параметр.
threads_number	int32	Количество потоков процесса. Необязательный параметр.
container_id	string	Идентификатор контейнера, в котором запущен процесс. Необязательный параметр.
container_name	string	Имя контейнера, в котором запущен процесс. Необязательный параметр.

UsersAndGroups

Информация о пользователях и группах.

Параметр	Тип данных	Описание
users	объект Users	Информация о пользователях.
groups	объект Groups	Информация о группах пользователей.

Users

Информация о пользователях.

Параметр	Тип данных	Описание
values	массив объектов User	Список пользователей. Необязательный параметр.

User

Информация о пользователе.

Параметр	Тип данных	Описание
user_id	int64	Идентификатор пользователя.
group_id	int64	Идентификатор группы, в которой состоит пользователь. Необязательный параметр.
user_name	string	Имя пользователя.
home_dir	string	Домашний каталог пользователя.
description	string	Описание пользователя. Необязательный параметр.
type	объект UserType	Тип пользователя: • USER_TYPE_UNSPECIFIED — не указан. • USER_TYPE_LOCAL — локальный пользователь. • USER_TYPE_ROAMING — пользователь, чьи данные хранятся на сервере.
group_ids	массив int64	Список групп, в которых состоит пользователь. Необязательный параметр.
is_local_admin	bool	Признак, является ли пользователь локальным администратором. Необязательный параметр.
is_disabled	bool	Признак, отключен ли пользователь. Необязательный параметр.
full_name	string	Полное имя пользователя. Необязательный параметр.
last_seen_ms	int64	Время последнего входа пользователя в систему в миллисекундах, прошедших с момента 00:00:00 UTC 1 января 1970. Необязательный параметр.

Параметр	Тип данных	Описание
`is_online`	bool	Признак того, находится ли сейчас пользователь онлайн. Необязательный параметр.
`has_sudo_rights`	bool	Признак наличия у пользователя прав sudo. Необязательный параметр.
`sid`	string	Идентификатор безопасности (SID) пользователя. Необязательный параметр.
`domain_name`	string	Доменное имя пользователя. Необязательный параметр.

Groups

Информация о группах пользователей.

Параметр	Тип данных	Описание
`values`	массив объектов Group	Список групп пользователей. Необязательный параметр.

Group

Информация о группе пользователей.

Параметр	Тип данных	Описание
`group_id`	int64	Идентификатор группы.
`group_name`	string	Имя группы.
`sid`	string	Идентификатор безопасности (SID) группы. Необязательный параметр.
`description`	string	Описание группы. Необязательный параметр.

Mounts

Информация о точках монтирования.

Параметр	Тип данных	Описание
`values`	массив объектов Mount	Список точек монтирования. Необязательный параметр.

Mount

Информация о точке монтирования.

Параметр	Тип данных	Описание
`device`	string	Устройство, которое монтируется.
`mount_point`	string	Путь монтирования.
`filesystem`	string	Тип файловой системы. Примеры: `tmpfs`, `ext4`, `ntfs`
`free_space_bytes`	uint64	Свободное место в файловой системе в байтах. Необязательный параметр.
`size_bytes`	uint64	Общий размер файловой системы в байтах. Необязательный параметр.

DockerContainers

Информация о Docker-контейнерах.

Параметр	Тип данных	Описание
`containers`	массив объектов DockerContainer	Список Docker-контейнеров. Необязательный параметр.

DockerContainer

Информация о Docker-контейнере.

Параметр	Тип данных	Описание
`container_id`	string	Идентификатор контейнера.
`names`	массив string	Список имен контейнера. Необязательный параметр.
`image`	string	Образ контейнера.
`command`	string	Команда запуска контейнера.
`created_ms`	int64	Время создания контейнера в миллисекундах, прошедших с момента 00:00:00 UTC 1 января 1970.
`path`	string	Путь запуска контейнера.

Параметр	Тип данных	Описание
`status`	объект DockerContainerStatus	Статус контейнера: • `DCS_UNSPECIFIED` — статус не указан. • `DCS_CREATED` — контейнер создан. • `DCS_RESTARTING` — контейнер перезапускается. • `DCS_RUNNING` — контейнер работает. • `DCS_REMOVING` — контейнер удаляется. • `DCS_PAUSED` — контейнер приостановлен. • `DCS_EXITED` — контейнер завершен. • `DCS_DEAD` — контейнер завершен некорректно или не может быть удален.
`ports`	массив объектов DockerPort	Список портов контейнера. Необязательный параметр.

DockerPort

Информация о порте Docker-контейнера.

Параметр	Тип данных	Описание
`hosts`	массив объектов DockerPortHost	Список привязок порта к хостам. Необязательный параметр.
`private_port`	int64	Приватный порт контейнера.
`protocol`	объект DockerPortProtocol	Протокол соединения: • `DPP_UNSPECIFIED` — не указан. • `DPP_TCP` — TCP. • `DPP_UDP` — UDP. • `DPP_SCTP` — SCTP.

DockerPortHost

Информация о привязке порта Docker-контейнера к хосту.

Параметр	Тип данных	Описание
`host`	string	Адрес хоста.
`port`	int32	Порт подключения.

Packages

Информация о пакетах.

Параметр	Тип данных	Описание
`values`	массив объектов Package	Список пакетов. Необязательный параметр.

Package

Информация о пакете.

Параметр	Тип данных	Описание
`architecture`	string	Архитектура пакета. Примеры: `x86_64`, `noarch`, `amd64`, `all`, `i686`
`size_bytes`	uint64	Размер пакета в байтах. Необязательный параметр.
`format`	объект PackageFormat	Формат пакета: • `PACKAGE_FORMAT_UNSPECIFIED` — не указан. • `PACKAGE_FORMAT_DEB` — DEB. • `PACKAGE_FORMAT_RPM` — RPM. • `PACKAGE_FORMAT_WIN` — Windows. • `PACKAGE_FORMAT_APPX` — APPX. • `PACKAGE_FORMAT_APK` — APK. • `PACKAGE_FORMAT_PACMAN` — Pacman. • `PACKAGE_FORMAT_SNAP` — Snap. • `PACKAGE_FORMAT_CHROME_EXT` — Chrome extension. • `PACKAGE_FORMAT_FIREFOX_EXT` — Firefox extension. • `PACKAGE_FORMAT_BREW_PKG` — Homebrew. • `PACKAGE_FORMAT_MACPORTS` — MacPorts. • `PACKAGE_FORMAT_NPM` — NPM. • `PACKAGE_FORMAT_PYPI` — PyPI. • `PACKAGE_FORMAT_PKG` — PKG.
`vendor`	string	Производитель пакета.
`install_time_ms`	int64	Время установки пакета в миллисекундах, прошедших с момента 00:00:00 UTC 1 января 1970. Необязательный параметр.

Параметр	Тип данных	Описание
`location`	string	Путь к пакету.
`groups`	массив string	Список групп пакета. Примеры: `["System Environment/Base"]`, `["utils"]` Необязательный параметр.
`priority`	string	Приоритет пакета. Примеры: `standard`, `important`
`multi_arch`	string	Информация, поддерживает ли пакет различные архитектуры.
`source`	string	Источник пакета.
`identifier`	string	Идентификатор пакета.
`name`	string	Название пакета.
`version`	string	Версия пакета.
`description`	string	Описание пакета.

SharedFolders

Информация об общих папках.

Параметр	Тип данных	Описание
`values`	массив объектов SharedFolder	Список общих папок. Необязательный параметр.

SharedFolder

Информация об общей папке.

Параметр	Тип данных	Описание
`type`	объект ShareType	Тип ресурса: • `SHARE_TYPE_UNSPECIFIED` — не указан. • `SHARE_TYPE_SMB` — SMB. • `SHARE_TYPE_NFS` — NFS.
`name`	string	Имя общей папки. Необязательный параметр.
`path`	string	Путь к общей папке.
`description`	string	Описание общей папки. Необязательный параметр.

OpenPorts

Информация об открытых портах.

Параметр	Тип данных	Описание
`values`	массив объектов NetworkPort	Список открытых портов. Необязательный параметр.

NetworkPort

Информация об открытом сетевом порте.

Параметр	Тип данных	Описание
`protocol`	объект NetworkPortProtocol	Протокол порта: • `NPP_UNSPECIFIED` — не указан. • `NPP_UDP` — UDP. • `NPP_UDP6` — UDP IPv6. • `NPP_TCP` — TCP. • `NPP_TCP6` — TCP IPv6.
`local_ip`	string	Локальный IP-адрес.
`local_port`	int32	Локальный порт.
`remote_ip`	string	Удаленный IP-адрес.
`remote_port`	int32	Удаленный порт.
`tx_queue`	uint64	Очередь передачи в байтах. Необязательный параметр.
`rx_queue`	uint64	Очередь приема в байтах. Необязательный параметр.
`inode`	uint64	Идентификатор inode. Необязательный параметр.

Параметр	Тип данных	Описание
`state`	объект NetworkPortState	Состояние порта: • `NPS_UNSPECIFIED` — состояние не указано. • `NPS_TCP_ESTABLISHED` — соединение установлено. • `NPS_TCP_SYN_SENT` — отправлен запрос на установку соединения (SYN). • `NPS_TCP_SYN_RECV` — получен запрос на установку соединения (SYN). • `NPS_TCP_FIN_WAIT1` — отправлен запрос на закрытие соединения. • `NPS_TCP_FIN_WAIT2` — запрос на закрытие соединения подтвержден, ожидается финальное подтверждение. • `NPS_TCP_TIME_WAIT` — ожидается закрытие соединения. • `NPS_TCP_CLOSE` — соединение закрыто. • `NPS_TCP_CLOSE_WAIT` — одна сторона готова закрыть соединение. • `NPS_TCP_LAST_ACK` — ожидается финальное подтверждение закрытия соединения. • `NPS_TCP_LISTEN` — ожидаются входящие соединения. • `NPS_TCP_CLOSING` — происходит закрытие соединения. • `NPS_TCP_DELETE_TCB` — происходит удаление блока управления передачей (TCB). Состояние доступно только в ОС Windows.
`process_id`	int64	Идентификатор процесса, использующего порт. Необязательный параметр.
`process_name`	string	Имя процесса, использующего порт. Необязательный параметр.
`container_id`	string	Идентификатор контейнера, в котором размещен порт. Необязательный параметр.
`container_name`	string	Имя контейнера, в котором размещен порт. Необязательный параметр.

Сбор событий ИБ на агентах

При разворачивании агента на устройстве, выполняется сбор следующих событий ИБ:

- входы и выходы пользователей;

- создание процессов;
- добавление или изменение сервисов;
- управление пользователями;
- действия с объектами.



Настройки сбора событий задаются в конфигурации группы агентов.



Настройка конвейера для сбора и обработки событий с агентов приведена в разделе Сбор событий с агентов.

Для сбора событий ИБ используются следующие механизмы:

- в ОС Windows — **Windows Event Log** и чтение событий журнала модулем **Чтение файлов**;
- в macOS — **ESF**;
- в ОС Linux — **eBPF**.



Для сбора событий по технологии eBPF:

- Необходима версия ядра Linux не ниже 5.10. Чтобы проверить текущую версию ядра, используйте следующую команду:

```
uname -r
```

- Ядро Linux должно поддерживать BPF. Чтобы это проверить, используйте следующую команду:

```
ls /sys/kernel/btf/vmlinux
```

Если ядро поддерживает технологию BPF, команда выведет информацию о файле `/sys/kernel/btf/vmlinux`. В противном случае команда вернет сообщение следующего вида:

```
ls: cannot access '/sys/kernel/btf/vmlinux': No such file or
directory
```

Структура событий на агентах

Поле	Тип данных	Описание
`agent`	Object	Основная информация об агенте, на хосте которого произошло событие. Подробности приведены в описании секции `agent`.

Поле	Тип данных	Описание
`data`	Object	Основная информация о событии. Структура и набор полей в данной секции зависят от используемой операционной системы на агенте и от типа события.
`full_log`	String	JSON-строка, содержащая сведения о событии из логов агента. Поле может отсутствовать в зависимости от типа события.
`id`	String	Уникальный идентификатор события в формате UUID.
`location`	String	Полный путь до файла логов, в котором хранится информация о событии.
`timestamp`	String	Временная метка возникновения события.

Поля секции `agent`

В секции `agent` хранится основная информация об агенте, на хосте которого произошло событие.

Поле	Тип данных	Описание
`hostname`	String	Имя хоста агента.
`id`	String	Уникальный идентификатор агента в формате UUID.
`ip`	String	IP-адрес агента.
`name`	String	Название агента.
`os_name`	String	Название операционной системы на компьютере агента.
`os_version`	String	Версия операционной системы на компьютере агента.

Пример события


```
{
  "agent": {
    "hostname": "rpoint-agent-debian.rvision.local",
    "id": "0ccaa170-25a7-5a0e-a748-f52d118833b1",
    "ip": "198.51.100.0",
    "name": "rpoint-agent-debian.rvision.local",
    "os_name": "Debian GNU/Linux",
    "os_version": "12"
  },
  "data": {
    "audit": {
      "_ev": 204,
      "_os": 100,
      "exe": "/usr/bin/bash",
      "id": 5,
      "is_container": false,
```

```
      "login_type": "account_switch",
      "prevuid": 1000,
      "prevuname": "user",
      "session": 1833,
      "success": "yes",
      "terminal": "pipe:[33703]",
      "time": 1750077602349,
      "type": "LOGIN",
      "uid": 0,
      "uname": "root"
    }
  },
  "full_log": "type=LOGIN msg=audit(1750077602.349:5): uid=0 type=account_switch
terminal="pipe:[33703]" exe="/usr/bin/bash" success=yes session=1833",
  "id": "b24d3967-0c4b-48f5-9562-3c5c839ded47",
  "location": "logs/login-logoff.json",
  "timestamp": "2025-06-16T15:40:02"
}
```


Аудит

В данном разделе представлена информация о процедурах аудита, используемых для мониторинга системных событий и проверки состояния источников событий в системе.

Содержание раздела:

- Аудит источников событий
  - Политики аудита источников событий
  - Настройка конвейера для аудита коллектора
- Аудит системных событий
  - Модель события аудита
  - Создание оповещений о критических событиях
  - Отслеживание действий суперадминистраторов
  - Отслеживание действий над объектами

Аудит источников событий

В данном разделе представлена информация о способах мониторинга и проверки состояния источников событий системы.

Содержание раздела:

- Политики аудита источников событий
- Настройка конвейера для аудита коллектора

Политики аудита источников событий

Данный раздел описывает процесс работы с источниками событий в системе. Работа осуществляется в разделе **Инструменты** → **Аудит источников** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об аудите источников

В системе можно контролировать состояние источников, из которых поступают события в коллекторы.

Состояние источников оценивается на основе данных о частоте и количестве поступающих событий. Это позволяет обнаружить отклонения, такие как слишком слабые или слишком интенсивные потоки событий либо полное их отсутствие. Для отслеживания подобных ситуаций используются настраиваемые политики аудита источников.

Настройки и логика работы политик аудита зависит от их типов. Система позволяет задавать следующие политики аудита источников:

- **По маске** — для политик данного типа источники событий определяются по указанной маске полей на нормализаторе. Политика по маске применяется ко всем источникам, полученным с нормализатора.
- **По точке входа** — для политик данного типа источники событий определяются по точкам входа. Политику по точке входа можно применить к нескольким точкам, но каждой точке может быть назначена только одна такая политика.

Политики аудита источников позволяют автоматически отслеживать изменения в потоке событий и отправлять уведомления при достижении заданных пороговых значений через настроенные в системе интеграции.



В связи с техническими ограничениями Prometheus рекомендуется использовать политики аудита типа **По маске** с методом расчета **По среднему EPS**, только если требуется обрабатывать не более 3000 источников событий.

При попытке обработать более 3000 источников стабильная работа аудита не гарантируется.

Интерфейс раздела

Интерфейс раздела зависит от открытой вкладки. Доступные вкладки:

- **Источники** — вкладка для работы с источниками данных, определенными маской.
- **Точки входа** — вкладка для работы с точками входа, для которых заданы политики аудита.
- **Политики аудита источников** — вкладка для работы с политиками аудита.

Интерфейс вкладки Источники

В верхней части вкладки расположена панель инструментов, которая включает следующие компоненты:

- Поле **Поиск** предназначено для быстрого поиска источников событий.
- Кнопка **Фильтр** ( ) позволяет применять фильтры для отображения источников событий.

В рабочей области отображается таблица источников, определенных значениями полей в масках. Если один и тот же источник получен с нескольких конвейеров, он отображается в списке только один раз.

Таблица содержит следующие столбцы:

- **Статус** — текущий статус источника событий:
  - **Активен** — заданная политика аудита включена, события поступают на источник.
  - **Нарушение политики** — заданная политика аудита включена, на источник поступает событий меньше или больше, чем определено политикой аудита.
  - **Не определяется** — заданная политика аудита выключена.
- **Название** — название источника событий. Название формируется из полей, определенных в маске.

  Например, в маске указаны поля `dvendor`, `dhost`, `user`, `tenantId`. Название источника может выглядеть так: `Windows|admin.domain.local|UserAdmin|Main`.
- **Время последнего обновления** — дата и время последнего обновления информации об источнике.
- **Политика аудита источников** — название политики аудита, примененной к источнику.

При работе с таблицей на вкладке **Источники** доступны следующие операции:

- Поиск источников событий по полю **Название**.
- Фильтрация источников событий по полям **Статус** и **Политика аудита источников**. Значения выбираются из выпадающего списка.

  В поле **Политика аудита источников** возможен выбор нескольких пунктов.
- Сортировка источников событий по полю **Название**.
- Настройка отображения таблицы.
- Обновление таблицы.

Интерфейс вкладки Точки входа

В верхней части вкладки расположена панель инструментов, которая включает следующие компоненты:

- Поле **Поиск** предназначено для быстрого поиска точек входа.

- Кнопка **Фильтр** ( ) позволяет применять фильтры для отображения точек входа по различным критериям.

В рабочей области вкладки **Точки входа** отображается таблица имеющихся в системе точек входа, для которых включен аудит. Таблица содержит следующие столбцы:

- **Статус** — текущий статус точки входа:
  - **Включен** — коллектор и конвейер, содержащие точку входа, включены, заданная политика аудита также включена и на точку поступают события.
  - **Выключен** — коллектор или конвейер, содержащий точку входа, отключен.
  - **Нарушение политики** — коллектор и конвейер, содержащие точку входа, включены, заданная политика аудита также включена, но на точку входа поступает событий меньше или больше, чем определено политикой аудита.
  - **Не определяется** — заданная политика аудита выключена.
- **ID элемента** — уникальный идентификатор точки входа.
- **Название** — имя точки входа.
- **Тип точки входа** — тип точки входа, например, Database.
- **Адрес подключения** — IP-адрес или доменное имя источника событий.
- **Порт подключения** — номер порта для подключения к источнику событий
- **Конвейер** — название и идентификатор конвейера событий.
- **Коллектор** — название и идентификатор коллектора.
- **Политика аудита источников** — название политики аудита, примененной к точке входа.

При работе с таблицей на вкладке **Точки входа** доступны следующие операции:

- Поиск точек входа по полям **Название** и **ID элемента**.
- Фильтрация точек входа по полям **Тип точки входа**, **Статус**, **Коллектор**, **Конвейер** и **Политика аудита источников**. Значения выбираются из выпадающего списка.



В полях **Коллектор**, **Конвейер** и **Политика аудита источников** возможен выбор нескольких пунктов.

- Сортировка точек входа по полям **Название**, **Адрес подключения** и **Порт подключения**.
- Настройка отображения таблицы.
- Обновление таблицы.

Интерфейс вкладки Политики аудита источников

В верхней части вкладки расположена панель инструментов, которая включает следующие компоненты:

- Кнопка **Создать** ( ) позволяет добавить новую политику аудита источников.
- Поле **Поиск** предназначено для быстрого поиска политик аудита источников.
- Кнопка **Фильтр** ( ) позволяет применять фильтры для отображения политик аудита источников

по различным критериям.

В рабочей области вкладки **Политики аудита источников** отображается таблица существующих политик аудита источников. Таблица содержит следующие столбцы:

- **Название** — имя политики.
- **Статус** — текущий статус политики: **Включена** или **Выключена**.
- **Тип политики** — способ определения источника: **По точке входа** или **По маске**.
- **Метод расчета** — метод расчета для политик **По маске**: **По количеству событий** или **По среднему EPS**.
- **Нижний порог** — нижний порог, по которому фиксируется нарушение политики:
  - Для политик **По точке входа** и политик **По маске** с методом расчета **По количеству событий**: минимальное количество событий, которое можно получить от источника без нарушения политики.
  - Для политик **По маске** с методом расчета **По среднему EPS**: минимальное допустимое отклонение от среднего значения EPS в процентах, которое можно получить от источника без нарушения политики.
- **Верхний порог** — верхний порог, по которому фиксируется нарушение политики:
  - Для политик **По точке входа** и политик **По маске** с методом расчета **По количеству событий**: максимальное количество событий, которое можно получить от источника без нарушения политики.
  - Для политик **По маске** с методом расчета **По среднему EPS**: максимальное допустимое отклонение от среднего значения EPS в процентах, которое можно получить от источника без нарушения политики.
- **Период** — период, за который проверяются поступаемые события. Используется для политик **По маске** с методом расчета **По среднему EPS**.
- **Интервал** — интервал времени, логика работы которого зависит от политики аудита и метода расчета:
  - Для политик **По точке входа** и политик **По маске** с методом расчета **По количеству событий**: интервал, в течение которого проверяется политика.
  - Для политик **По маске** с методом расчета **По среднему EPS**: интервал, в течение которого ежеминутно проверяется отклонение количества событий от нормы.
- **Дата создания** — дата и время создания политики.
- **Дата изменения** — дата и время последнего изменения политики.

При работе с таблицей на вкладке **Политики аудита источников** доступны следующие операции:

- Поиск политик аудита источников по полям **Название** и **ID элемента**.
- Фильтрация политик аудита источников по полям **Статус**, **Тип политики** и **Метод расчета**. Значения выбираются из выпадающих списков.
- Сортировка политик аудита источников по полям **Название**, **Дата создания**, **Дата изменения**, **Нижний порог** и **Верхний порог**.
- Настройка отображения таблицы.

При выборе отдельного источника событий или политики аудита в правой части рабочей области отображается карточка с детальной информацией.

Просмотр источников событий по маске

Чтобы просмотреть карточку источника событий по маске:

1. Перейдите на вкладку **Источники событий** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных источниках.
2. Нажмите на строку источника в списке. Система отобразит в правой части экрана карточку с информацией об источнике.

В верхней части карточки отображается название источника событий.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка источника событий включает в себя следующие поля:

- **Название** — название источника событий. Название формируется из полей, определенных в маске.



Например, в маске указаны поля `dvendor`, `dhost`, `user`, `tenantId`. Название источника может выглядеть так: `Windows|admin.domain.local|UserAdmin|Main`.

- **Статус** — текущий статус источника событий:
  - **Активен** — заданная политика аудита включена, события поступают на источник.
  - **Нарушение политики** — заданная политика аудита включена, на источник поступает событий меньше или больше, чем определено политикой аудита.
  - **Не определяется** — заданная политика аудита выключена.
- **Политика аудита источников** — название политики аудита, примененной к источнику.
- **Время последнего обновления** — дата и время последнего обновления информации об источнике.
- **Маска** — список полей маски и их значений.

Работа с вкладкой Точки входа

Доступные операции на вкладке **Точки входа**:

- Просмотр карточки точки входа
- Переход к конвейеру источника событий
- Переход к коллектору источника событий
- Переход к секрету источника событий
- Выбор политики аудита источников

Просмотр карточки точки входа

Чтобы просмотреть карточку точки входа:

1. Перейдите на вкладку **Точки входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных точках входа.
2. Нажмите на строку точки входа в списке. Система отобразит в правой части экрана карточку этой точки с подробной информацией о ней.

В верхней части карточки отображается название точки входа.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка точки входа включает в себя следующие поля:

- **ID** — уникальный идентификатор точки входа.
- **Название** — название точки входа.
- **ID тенанта** — идентификатор тенанта, к которому относится точка входа.
- **Тип точки входа** — тип точки входа, например, Database.
- **Статус** — текущий статус точки входа:
  - **Включен** — коллектор и конвейер, содержащие точку входа, включены, заданная политика аудита также включена и на точку поступают события.
  - **Выключен** — коллектор или конвейер, содержащий точку входа, отключен.
  - **Нарушение политики** — коллектор и конвейер, содержащие точку входа, включены, заданная политика аудита также включена, но на точку входа поступает событий меньше или больше, чем определено ее политикой аудита.
  - **Не определяется** — заданная политика аудита выключена.
- **Адрес подключения** — IP-адрес или доменное имя источника событий.

  Для точек входа AMQP и Database в поле указывается название секрета, который используется для задания адреса подключения.
- **Порт точки входа** — номер порта для подключения к источнику событий.
- **Конвейер** — название конвейера обработки событий, к которому относится точка входа.
- **Коллектор** — название коллектора, к которому относится точка входа.
- **Политика аудита источников** — название политики аудита, примененной к точке входа.

Переход к конвейеру источника событий

Чтобы перейти к конфигурации конвейера, в котором настроен источник событий:

1. Перейдите на вкладку **Точки входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных источниках событий.
2. Нажмите на строку точки входа в списке. Система отобразит в правой части экрана карточку этой точки с подробной информацией о ней.

3. Нажмите на значение поля **Конвейер** в карточке точки входа. Система откроет страницу с конфигурацией конвейера, в котором находится выбранная точка.

Переход к коллектору источника событий

Чтобы перейти к коллектору, к которому относится источник событий:

1. Перейдите на вкладку **Точки входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных точках входа.
2. Нажмите на строку точки входа в списке. Система отобразит в правой части экрана карточку этой точки с подробной информацией о ней.
3. Нажмите на значение поля **Коллектор** в карточке точки входа. Система откроет страницу раздела **Коллекторы** и отобразит карточку коллектора, в котором содержится конвейер с выбранной точкой.

Переход к секрету источника событий

Для точек входа AMQP и Database адрес подключения задается с помощью секрета типа **Строка подключения**.

Чтобы просмотреть информацию о секрете, использующемся в точке входа:

1. Перейдите на вкладку **Точки входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных точках входа.
2. Нажмите на строку точки входа типа AMQP или Database в списке. Система отобразит в правой части экрана карточку этой точки с подробной информацией о ней.
3. Нажмите на значение поля **Адрес подключения** в карточке точки входа. Система откроет страницу раздела **Секреты** и отобразит карточку секрета, который используется в выбранной точке.

Выбор политики аудита источников

Чтобы изменить выбранную политику аудита для точки входа:

1. Перейдите на вкладку **Точки входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных точках входа.
2. Нажмите на строку точки входа в списке. Система отобразит в правой части экрана карточку этой точки с подробной информацией о ней.
3. Нажмите на кнопку в строке поля **Политика аудита источников** в карточке точки входа и выберите политику из выпадающего списка. На выбор доступны только включенные политики аудита источников.
4. Нажмите на кнопку **Сохранить**. Выбранная политика применится к точке входа.



Вы также можете выбирать политику аудита для точки входа:

- при добавлении или изменении точки входа;
- при добавлении или изменении политики аудита источников.

Работа с политиками аудита источников

Доступные операции на вкладке **Политики аудита источников**:

- Добавление политики аудита источников
- Включение политики аудита источников
- Отключение политики аудита источников
- Просмотр политики аудита источников
- Изменение политики аудита источников
- Удаление политики аудита источников

Добавление политики аудита источников

Система позволяет добавлять политики аудита следующих типов:

- **По маске** — для политики данного типа источники событий определяются по заданной маске.
- **По точке входа** — для политики данного типа источники событий определяются по точкам входа.

Добавление политики По маске



В связи с техническими ограничениями Prometheus рекомендуется использовать политики аудита типа **По маске** с методом расчета **По среднему EPS**, только если требуется обрабатывать не более 3000 источников событий.

При попытке обработать более 3000 источников стабильная работа аудита не гарантируется.

Чтобы добавить политику аудита источников типа **По маске**:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на кнопку **Создать** (┼). Отобразится форма создания политики аудита источников.
3. Введите название политики аудита источников.
4. При необходимости введите описание политики аудита источников.
5. Выберите из выпадающего списка тип политики **По маске**.
6. Задайте маску. Для этого укажите от одного до десяти полей событий для определения источников.

   Чтобы добавить поле в маску:

   a. Нажмите на кнопку **Добавить**. Отобразится строка добавления поля.

   b. Выберите поле универсальной модели события или универсальной модели 2.0 либо введите пользовательское значение.

   c. Если поле не обязательное, установите переключатель **Опциональное** в активное положение.



Хотя бы одно поле маски должно быть обязательным.

Чтобы удалить поле из маски, нажмите на кнопку в строке этого поля.

7. Выберите из выпадающего списка метод расчета политики.

По количеству событий

При данном методе расчета условие нарушения политики определяется количеством событий, поступивших за определенный интервал времени.

Задайте параметры метода:

a. Задайте пороговые значения количества событий:
   - **Нижний порог** — минимальное количество событий, которое можно получить от источника без нарушения политики.
   - **Верхний порог** (опционально) — максимальное количество событий, которое можно получить от источника без нарушения политики.

b. Задайте интервал времени между проверками источников в минутах, часах или днях. По умолчанию установлен интервал в 30 минут.

   Если за указанный интервал событий поступает меньше, чем указано в нижнем пороге, или больше, чем указано в верхнем пороге, то будет зафиксировано нарушение политики.

   По умолчанию установлен интервал в 30 минут.



Подробности о логике работы метода приведены в разделе Метод расчета по количеству событий.

По среднему EPS

При данном методе расчета условие нарушения политики определяется отклонением от среднего количества событий в секунду (EPS), поступивших за определенный период времени.

Задайте параметры метода:

a. Задайте пороговые значения количества событий:
   - **Нижний порог** — максимальное допустимое отклонение вниз от среднего значения EPS в процентах, которое можно получить от источника без нарушения политики.
   - **Верхний порог** (опционально) — максимальное допустимое отклонение вверх от среднего значения EPS в процентах, которое можно получить от источника без нарушения политики.

b. Задайте общий период времени в минутах, часах или днях, за который должны проверяться события из источника. Данный период используется для расчета среднего показателя EPS.

   По умолчанию установлен период в 7 дней. Максимальный допустимый период: 30 дней.

c. Задайте интервал между проверками источников в минутах. В течение данного интервала будет происходить поминутная проверка количества событий, поступившего от источника. Если в течение всего интервала при каждой проверке количество событий отклоняется от среднего показателя EPS сильнее, чем указано в пороговых значениях, то будет зафиксировано нарушение политики.

   По умолчанию установлен интервал в 1 минуту. Максимальный допустимый интервал: 1440 минут (24 часа).



Подробности о логике работы метода приведены в разделе Метод расчета по среднему EPS.



При выборе метода расчета **По среднему EPS** системе необходимо собрать минимальное количество данных по источнику для корректного расчета среднего значения EPS. После включения политики система ожидает 24 часа, накапливая данные. В этот период нарушения политики по отклонениям EPS не фиксируются.

8. При необходимости выберите из дерева источников элементы, к которым необходимо применить политику аудита. Чтобы выбрать элемент, установите флажок слева от его названия в списке.

   Для выбора доступны нормализаторы всех установленных конвейеров во всех включенных коллекторах.

   Нормализаторы в дереве сгруппированы по конвейерам и коллекторам. Чтобы просмотреть список конвейеров нужного коллектора или список элементов нужного конвейера, нажмите на кнопку ⌄ справа от названия соответствующего родительского элемента в дереве.

   Чтобы скрыть отображенный список дочерних элементов, нажмите на кнопку ⌃ справа от названия родительского элемента в дереве.

9. При необходимости выберите интеграции для отправки уведомлений о нарушении политики аудита источников.
10. Нажмите на кнопку **Создать**. Система создаст новую политику аудита источников и отобразит ее в списке политик.

Добавление политики По точке входа

Чтобы добавить политику аудита источников типа **По точке входа**:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на кнопку **Создать** (＋). Отобразится форма создания политики аудита источников.
3. Введите название политики аудита источников.
4. При необходимости введите описание политики аудита источников.
5. Выберите из выпадающего списка тип политики **По точке входа**.

6. Задайте пороговые значения количества событий:
   - **Нижний порог** — минимальное количество событий, которое можно получить от источника без нарушения политики.
   - **Верхний порог** (опционально) — максимальное количество событий, которое можно получить от источника без нарушения политики.
7. Задайте интервал времени в минутах, часах или днях, за который будут проверяться поступаемые события.

   Если за указанный интервал событий поступает меньше, чем указано в нижнем пороге, или больше, чем указано в верхнем пороге, то будет зафиксировано нарушение политики.

   По умолчанию установлен интервал в 30 минут.

8. При необходимости выберите из дерева источников элементы, к которым необходимо применить политику аудита. Чтобы выбрать элемент, установите флажок слева от его названия в списке.

   Для выбора доступны точки входа всех установленных конвейеров во всех включенных коллекторах.

   Точки входа в дереве сгруппированы по конвейерам и коллекторам. Чтобы просмотреть список конвейеров нужного коллектора или список элементов нужного конвейера, нажмите на кнопку ⌄ справа от названия соответствующего родительского элемента в дереве.

   Чтобы скрыть отображенный список дочерних элементов, нажмите на кнопку ⌃ справа от названия родительского элемента в дереве.

9. При необходимости выберите интеграции для отправки уведомлений о нарушении политики аудита источников.
10. Нажмите на кнопку **Создать**. Система создаст новую политику аудита источников и отобразит ее в списке политик.

Включение политики аудита источников

После включения политика аудита источников активно применяется к привязанным точкам входа или нормализаторам и доступна для выбора при их создании.

Чтобы включить политику аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты → Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Установите переключатель состояния элемента в верхней части карточки в активное положение. Отобразится окно подтверждения включения политики аудита источников.
4. Нажмите на кнопку **Включить**. Система обновит информацию о статусе политики аудита источников, и она станет активной.

Отключение политики аудита источников

После отключения политика аудита источников перестает применяться к привязанным точкам входа или нормализаторам и недоступна для выбора при их создании. При этом источники, отслеживаемые данной политикой, переходят в статус **Не определяется**.

Чтобы отключить политику аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Установите переключатель состояния элемента в верхней части карточки в неактивное положение. Отобразится окно подтверждения отключения политики аудита источников.
4. Нажмите на кнопку **Отключить**. Система обновит информацию о статусе политики аудита источников, и она станет неактивной.

Просмотр политики аудита источников

Чтобы просмотреть карточку политики аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.

В верхней части карточки отображается название политики аудита источников. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранной политики.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка политики аудита источников содержит следующие поля:

- **ID** — уникальный идентификатор политики аудита источников.
- **Название** — имя политики аудита источников.
- **Описание** — описание политики аудита источников.
- **ID тенанта** — идентификатор тенанта.
- **Статус** — текущее состояние политики аудита источников: **Включена** или **Выключена**.
- **Тип политики** — способ определения источника: **По маске** или **По точке входа**.
- **Метод расчета** — метод расчета для политик **По маске**: **По количеству событий** или **По среднему EPS**.
- **Нижний порог** — нижний порог, по которому фиксируется нарушение политики:
  - Для политик **По точке входа** и политик **По маске** с методом расчета **По количеству событий**: минимальное количество событий, которое можно получить от источника без нарушения политики.

- Для политик **По маске** с методом расчета **По среднему EPS**: минимальное допустимое отклонение от среднего значения EPS в процентах, которое можно получить от источника без нарушения политики.

- **Верхний порог** — верхний порог, по которому фиксируется нарушение политики:
  - Для политик **По точке входа** и политик **По маске** с методом расчета **По количеству событий**: максимальное количество событий, которое можно получить от источника без нарушения политики.
  - Для политик **По маске** с методом расчета **По среднему EPS**: максимальное допустимое отклонение от среднего значения EPS в процентах, которое можно получить от источника без нарушения политики.
- **Период** — период, за который проверяются поступаемые события. Используется для политик **По маске** с методом расчета **По среднему EPS**.
- **Интервал** — интервал времени, логика работы которого зависит от политики аудита и метода расчета:
  - Для политик **По точке входа** и политик **По маске** с методом расчета **По количеству событий**: интервал, в течение которого проверяется политика.
  - Для политик **По маске** с методом расчета **По среднему EPS**: интервал, в течение которого ежеминутно проверяется отклонение количества событий от нормы.
- **Дата создания** — дата и время создания политики аудита источников.
- **Дата изменения** — дата и время последнего изменения политики аудита источников.
- **Создал** — имя пользователя, создавшего политику аудита источников.
- **Изменил** — имя пользователя, изменившего политику аудита источников.
- **Интеграции** — список интеграций для отправки уведомлений. Список представлен в виде таблицы со столбцами **Название** и **Статус**.

  Список можно сортировать по названию.

  Поле отображается, если для политики задана хотя бы одна интеграция.

- **Точки входа** — список точек входа, к которым применяется политика аудита. Список представлен в виде таблицы со столбцами **Название** и **Статус**.

  Список можно сортировать по названию.

  Поле отображается для политик **По точке входа**.

- **Источники** — список определенных по маске источников, к которым применяется политика аудита. Список представлен в виде таблицы со столбцами **Название** и **Статус**.

  Список можно сортировать по названию.

  Поле отображается для политик **По маске**.

Также в карточке находится переключатель статуса политики аудита источников.

Изменение политики аудита источников

Вы можете изменить политику аудита источников следующими способами:

- через меню действий в карточке политики;
- через изменение полей в карточке политики.



- Тип политики изменить невозможно.
- Название и описание системных политик изменить невозможно.
- Если при редактировании **системной политики аудита** в дереве источников были удалены связи с элементами конвейера, то для этих элементов аудит будет отключен.

  Если связи были удалены при редактировании **пользовательской политики**, то после нажатия на кнопку **Сохранить** отобразится окно, в котором необходимо выбрать действие над элементами с удаленными связями:

  - **Использовать политику по умолчанию** — для исключенных элементов конвейера будет назначена системная политика аудита:
    - для точек входа — **Политика аудита источников по точке входа**;
    - для нормализаторов — **Политика аудита источников по маске**.
  - **Выключить аудит** — для исключенных элементов конвейера аудит будет отключен.



Если был изменен набор полей в маске, все источники, которые были созданы по этой маске, удаляются. Новые источники будут определяться по мере поступления событий на нормализатор.

Удаление политики аудита источников

Вы можете удалить политику аудита источников через меню действий в ее карточке.



Невозможно удалить предустановленные политики аудита источников, а также политики, назначенные источникам событий.

Настройка параметров политики аудита источников

- Настройка маски
- Метод расчета по среднему EPS
- Метод расчета по количеству событий
- Системные политики аудита

Настройка маски

Маска — это набор полей события, которые используются для идентификации источника. Источник определяется как уникальный набор значений полей, указанных в маске. При изменении набора полей источники, сформированные по маске, удаляются.

Особенности маски:

- В маску можно добавить от одного до десяти полей.
- Поля в маске могут быть обязательными или опциональными. В маске должно быть хотя бы одно обязательное поле.
- Если любое из обязательных полей содержит пустое значение, источник не формируется.



Пустыми значениями считаются `null`, а также `0`.

Система не позволяет создавать несколько политик аудита с одинаковым набором полей в маске, даже если в них отличается указанный порядок или список опциональных полей.

Пример работы маски

В политике задана маска со следующими полями:

- `dvendor` (обязательное);
- `dhost` (обязательное);
- `user` (опциональное).

После обработки следующих трех событий формируются два источника:

dvendor	dhost	user	Результат
`Windows`	`admin.example.local`	`Admin`	Windows|admin.example.local|Admin
`WINDOWS`	`admin.example.local`	`Admin`	WINDOWS|admin.example.local|Admin
`null`	`admin.example.local`	`User32`	Источник не формируется.

Метод расчета по количеству событий

При использовании метода расчета **По количеству событий** условие нарушения политики определяется количеством событий, поступивших за определенный интервал времени.

Логика работы метода:

1. Проверяется количество событий, поступивших от источника за указанный интервал времени.
2. Если за рассматриваемый интервал событий пришло меньше, чем указано в нижнем пороге, или больше, чем указано в верхнем пороге, будет зафиксировано нарушение политики.

Пример расчета по количеству событий

Пусть задана политика с расчетом по количеству событий со следующими параметрами:

- **Нижний порог**: 10.
- **Верхний порог**: 100.

- **Интервал**: 30 минут.

Для источника с такой политикой каждые 30 минут будет запускаться проверка поступивших событий. Если за прошедший интервал времени (30 минут) событий пришло меньше 10 или больше 100, будет выявлено нарушение политики.

Метод расчета по среднему EPS



В связи с техническими ограничениями Prometheus рекомендуется использовать политики аудита типа **По маске** с методом расчета **По среднему EPS**, только если требуется обрабатывать не более 3000 источников событий.

При попытке обработать более 3000 источников стабильная работа аудита не гарантируется.

При использовании метода расчета **По среднему EPS** условие нарушения политики определяется отклонением от среднего количества событий в секунду (EPS), поступивших за определенный период времени.

Логика работы метода:

1. Высчитывается среднее значение EPS за указанный период. Данное значение принимается за норму.
2. Каждую минуту проверяется отклонение поступивших событий от нормы. Если в течение указанного интервала в каждую минуту поток событий отклоняется вниз сильнее, чем указано в нижнем пороге, или вверх сильнее, чем указано в верхнем пороге, будет зафиксировано нарушение политики.



При выборе метода расчета **По среднему EPS** системе необходимо собрать минимальное количество данных по источнику для корректного расчета среднего значения EPS. После включения политики система ожидает 24 часа, накапливая данные. В этот период нарушения политики по отклонениям EPS не фиксируются.

Пример расчета по среднему EPS

Пусть задана политика с расчетом по среднему значению EPS со следующими параметрами:

- **Нижний порог**: 5%.
- **Верхний порог**: 10%.
- **Период**: 10 дней.
- **Интервал**: 30 минут.

Для источника с такой политикой будет рассчитан средний показатель EPS за 10 дней, который будет принят за норму. Далее будут происходить ежеминутные проверки поступающего потока событий. Если в течение указанного интервала времени (30 минут) поток событий каждую минуту отклоняется от нормы более чем на 5% вниз или более чем на 10% вверх, то будет зафиксировано нарушение политики.

Системные политики аудита

В системе имеются следующие политики, применяемые по умолчанию:

- **Политика аудита источников по маске**;
- **Политика аудита источников по точке входа**.

Параметры системных политик можно редактировать. По умолчанию установлены следующие значения:

По маске

- Поля маски:
  - `dproduct` (обязательное);
  - `dvchost` (обязательное);
  - `dvc` (опциональное);
  - `deviceProcessName` (опциональное).
- Метод расчета: **По количеству событий**.
- Нижний порог: 1.
- Верхний порог: не задан.
- Интервал: 30 минут.

По точке входа

- Нижний порог: 1.
- Верхний порог: не задан.
- Интервал: 30 минут.

Настройка конвейера для аудита коллектора

Данное руководство описывает процесс настройки конвейера для обнаружения отсутствия ожидаемых событий от точек входа конвейеров выбранного коллектора.

О настройке аудита точек входа коллектора

Чтобы настроить конвейер для мониторинга состояния источников, первоначально требуется создать и настроить конвейер в рамках коллектора, который содержит источники событий с необходимыми метриками для мониторинга. Данные, проходящие через такой конвейер, должны быть отфильтрованы и проанализированы на предмет отсутствия активности от заданных источников.

Для создания конвейера корреляции на отсутствие события, понадобятся следующие элементы:

1. **Сбор метрик**: обеспечивает непрерывный сбор метрик активности от выбранных элементов коллектора.
2. **Нормализатор**: приводит собранные метрики к единому формату для упрощения последующего анализа.

3. **Коррелятор**: проверяет поступление событий в заданные временные интервалы (настраиваемый параметр `ttl`). Если от элемента события не поступают, генерируется корреляционное событие об их отсутствии. Настройка коррелятора включает определение оптимальных значений для `ttl` и `count` с учетом частоты и природы поступающих данных.
4. **Конечная точка**: отправляет обработанные и коррелированные события в хранилища данных и внешние системы.
5. **Сервис оповещений**: позволяет настраивать связи между правилами корреляции и правилами сегментации для эффективного распределения корреляционных событий по оповещениям и их отправки через почтовые интеграции.

Таким образом, настройка конвейера для мониторинга источников событий включает следующие шаги:

1. Добавление конвейера
2. Настройка конфигурации конвейера
    a. Сбор метрик
    b. Нормализация данных
    c. Корреляция данных
    d. Конечная точка
    e. Сервис оповещений
3. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.



Добавление и настройка конвейера обработки событий осуществляются в рамках коллектора, который содержит элементы с необходимыми метриками для мониторинга.

Чтобы добавить конвейер:

1. Перейдите на вкладку **Конвейеры** в карточке выбранного коллектора. Система отобразит список конвейеров коллектора.
2. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
3. Введите название конвейера.
4. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

5. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке

коллектора.

Настройка конфигурации конвейера



Перед настройкой конфигурации конвейера определите, метрики каких источников на конвейерах коллектора требуется отслеживать.

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера с помощью кнопки ⌄ в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Сбор метрик

Сбор метрик — это элемент конвейера, который собирает метрики по отдельным элементам конвейеров коллектора для их дальнейшей обработки и анализа.

С помощью сбора метрик можно отслеживать поступление событий от источника на конвейер и выявлять ситуации, когда точка входа перестает получать события.

При создании сбора метрик можно задать интервал между сборами метрик в секундах в поле **Период сбора метрик**.

Добавьте на конвейер сбор метрик. При создании выберите элементы для отслеживания и определите интервал между сборами метрик в поле **Период сбора метрик**. Для выбора доступны элементы всех конвейеров в рамках коллектора.

Нормализация данных

На этом этапе метрики, поступающие от различных элементов конвейера, приводятся к единому формату.

Нормализатор преобразовывает события с помощью установленных на нем правил нормализации. Этот элемент конвейера принимает на вход события, проводит их нормализацию и на выходе предоставляет нормализованные события.

Чтобы добавить нормализацию:

1. Добавьте на конвейер нормализатор. При создании элемента, используйте системное правило нормализации "Правило нормализации Мониторинг источников".

   Правило нормализации "Мониторинг источников", схема RObject

   ```
   id: "786d83cb-48e0-4d5a-b050-ebeea34455cd"
   name: "Правило нормализации Мониторинг источников"
   type: normalization_rule
   description: "Правило нормализации для событий сбора метрик"
   ```

```
version: 0.0.5
author: R-Vision
tags:
  - metrics

filter: !vrl |
  true

normalizer: !vrl |
  .dproduct = "metrics"
  . |= compact({
    "dvendor": to_string(.raw.namespace) ?? "",
    "name": to_string(.raw.tags.component_id) ?? "",
    "cnt": to_string(.raw.counter.value) ?? "",
    "cat" : to_string(.raw.name) ?? "", })

tests:
  # Наименование теста. Должен быть уникальным для каждого теста в рамках правила
  - name: "testing"
    # Список тестовых событий
    events:
      - raw:
{"counter":{"value":1751.0},"kind":"absolute","name":"component_received_event_byte
s_total","namespace":"vector","tags":{"component_id":"56c8067a-7f1f-4e4c-8095-
f8261ed8227d_WXAm_yVi0VM90-
btxewDl","component_kind":"transform","component_type":"remap","host":"collector-
bf157c1b-107d-4090-b137-65c89fa14933-795dbc6cff-mhckd"},"timestamp":"2024-05-
29T10:26:42.688089484Z"}
    # Проверка для тестовых событий
    assertion: !vrl |
      assert_eq!(.[0].cat, "component_received_event_bytes_total")
      assert_eq!(.[0].cnt, "1751")
      assert_eq!(.[0].dvendor, "vector")
      assert_eq!(.[0].name, "56c8067a-7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-
btxewDl")
```

▼ *Интерпретация*

В блоке `normalizer` правила применяется VRL-функция `compact` для создания упрощенной структуры данных. Она включает следующие поля:

- `dvendor`: присваивает данному полю строковое представление значения поля `namespace` из исходного события. Если поле `namespace` отсутствует, присваивается пустая строка.
- `name`: присваивает данному полю строковое представление значения `component_id`, содержащегося в тегах исходного события. Если значение `component_id` отсутствует, возвращается пустая строка.
- `cnt`: задается строковое представление значения счетчика `counter.value` из исходного события. Если значение счетчика отсутствует, возвращается пустая строка.
- `cat`: строковое представление поля name из исходного события. Если поле name не определено, результатом будет пустая строка.



Функция `compact` автоматически удаляет из результата любые ключи, которым присвоены пустые строки или значения `null`, тем самым создавая более простую для обработки структуру данных.

2. Выход элемента **Сбор метрик** соедините со входом элемента **Нормализатор**.

Корреляция данных

Чтобы события, поступающие на конвейер, коррелировались, необходимо добавить на конвейер коррелятор.

Коррелятор позволяет связывать различные события и создавать комплексные правила на их основе. На входе применяет правила корреляции к потоку событий. На выходе выдает корреляционные события.

Чтобы добавить коррелятор, необходимо установить в нем правило корреляции. Правила корреляции определяют, как коррелятор будет обрабатывать входящие данные и формировать на их основе корреляционные события.

1. Добавьте на конвейер коррелятор. При создании элемента, используйте системное правило корреляции "Правило корреляции Мониторинг источников".

   Правило корреляции "Мониторинг источников", схема RObject

```
id: "6cd640a1-c065-40fc-9341-78067ccdefe6"
name: "Правило корреляции Мониторинг источников"
type: correlation_rule
description: "Правило корреляции ориентировано на события с элемента Сбор метрик.
При отсутствии событий с точки входа в течении заданного периода, будет создано
корреляционное событие"
author: R-Vision
version: 0.0.16
known_false_positives:
  - "события приходят блоком данных с определенной периодичностью, которая выше
ttl"
severity: high
tags:
  - metrics

filter: !vrl |
  .dvendor == "vector"

aliases:
  event_1:
    filter: !vrl |
      .cat == "component_received_events_total" && .cnt != "0"

select:
  alias: event_1

group:
```

```
  - alias: event_1
    by:
      - cnt
      - name
    count: 3

ttl: 60

trigger_alert: true

on_correlate: !vrl |
  .dvendor = "vector"
  cs = [ encode_json(%event_1[0].raw), encode_json(%event_1[1].raw),
encode_json(%event_1[2].raw)]
  .cs1 = join(cs, ", ") ?? ""
  .cs1Label = "Базовые события"
  .cnt = to_string(%event_1[0].cnt) ?? "-"
  name = to_string(%event_1[0].name) ?? "-"
  .name = name
  .message = "Перестали поступать события с источника {{name}}"

tests:
# Наименование теста. Должен быть уникальным для каждого теста в рамках правила
  - name: Test event
# Список тестовых событий
    events:
     - {"id":"2f81df63-c2ea-429e-b16d-
cdd1cc1d7a63","sourceIp":"","tenantId":"00000000-0000-0000-0000-
000000000000","art":"2024-06-
04T14:02:18.000+03:00","raw":"{\"counter\":{\"value\":13.0},\"kind\":\"absolute\",\
"name\":\"component_received_events_total\",\"namespace\":\"vector\",\"tags\":{\"co
mponent_id\":\"56c8067a-7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-
btxewDl\",\"component_kind\":\"transform\",\"component_type\":\"remap\",\"host\":\"
collector-bf157c1b-107d-4090-b137-65c89fa14933-85c5cb5c6f-
qnh97\"},\"timestamp\":\"2024-06-
04T11:02:18.582264635Z\"}","collectorId":"bf157c1b-107d-4090-b137-
65c89fa14933","timestamp":"2024-06-
04T14:02:18.000+03:00","cnt":13,"type":"нормализованное
событие","dvendor":"vector","dproduct":"metrics","name":"56c8067a-7f1f-4e4c-8095-
f8261ed8227d_WXAm_yVi0VM90-btxewDl","cat":"component_received_events_total"}
     - {"id":"2f81df63-c2ea-429e-b16d-
cdd1cc1d7a63","sourceIp":"","tenantId":"00000000-0000-0000-0000-
000000000000","art":"2024-06-
04T14:02:18.000+03:00","raw":"{\"counter\":{\"value\":13.0},\"kind\":\"absolute\",\
"name\":\"component_received_events_total\",\"namespace\":\"vector\",\"tags\":{\"co
mponent_id\":\"56c8067a-7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-
btxewDl\",\"component_kind\":\"transform\",\"component_type\":\"remap\",\"host\":\"
collector-bf157c1b-107d-4090-b137-65c89fa14933-85c5cb5c6f-
qnh97\"},\"timestamp\":\"2024-06-
04T11:02:18.582264635Z\"}","collectorId":"bf157c1b-107d-4090-b137-
65c89fa14933","timestamp":"2024-06-
```

```
04T14:02:18.000+03:00","cnt":13,"type":"нормализованное
событие","dvendor":"vector","dproduct":"metrics","name":"56c8067a-7f1f-4e4c-8095-
f8261ed8227d_WXAm_yVi0VM90-btxewDl","cat":"component_received_events_total"}
     - {"id":"22daee10-9148-4284-be42-
3c0e9c2b9f33","sourceIp":"","tenantId":"00000000-0000-0000-0000-
000000000000","art":"2024-06-
04T14:01:48.000+03:00","raw":"{\"counter\":{\"value\":13.0},\"kind\":\"absolute\",\
"name\":\"component_received_events_total\",\"namespace\":\"vector\",\"tags\":{\"co
mponent_id\":\"56c8067a-7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-
btxewDl\",\"component_kind\":\"transform\",\"component_type\":\"remap\",\"host\":\"
collector-bf157c1b-107d-4090-b137-65c89fa14933-85c5cb5c6f-
qnh97\"},\"timestamp\":\"2024-06-
04T11:01:48.582435089Z\"}","collectorId":"bf157c1b-107d-4090-b137-
65c89fa14933","timestamp":"2024-06-
04T14:01:48.000+03:00","cnt":13,"type":"нормализованное
событие","dvendor":"vector","dproduct":"metrics","name":"56c8067a-7f1f-4e4c-8095-
f8261ed8227d_WXAm_yVi0VM90-btxewDl","cat":"component_received_events_total"}

    assertion: !vrl |
      assert_eq!(.[0].message, "Перестали поступать события с источника 56c8067a-
7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-btxewDl")
      assert_eq!(.[0].cnt, "13")
      assert_eq!(.[0].dvendor, "vector")
      assert_eq!(.[0].severity, "high")
```

▼ *Интерпретация*

Правило корреляции "Мониторинг источников" предназначено для отслеживания метрик конвейеров коллектора. Если события с точки входа не поступают в течение 1 минуты, коррелятор генерирует корреляционное событие.

Логика фильтрации (filter)

- Основной фильтр — `.dvendor == "vector"`, выбирающий события, у которых значение `dvendor` равно "vector".

Логика фильтрации (aliases)

- Псевдоним `event_1`: фильтр для событий с метрикой, где `cat` равно "component_received_events_total" и `cnt` не равно "0".

Соединение потоков событий (select и group)

- `select`: выбор псевдонима для дальнейшей обработки. В данном случае используется `event_1`.
- `group`: группировка событий по полям `cnt` и `name`. Группировка применяется к псевдониму `event_1` и формируется при накоплении 3 событий.

Срок хранения промежуточных данных корреляции (ttl)

- `ttl` на уровне правила задает общее максимальное время жизни промежуточных корреляционных данных — 60 секунд. Это время определяет, как долго промежуточные

результаты будут храниться в памяти до их автоматического удаления.

Пост-корреляционная логика (on_correlate)

При срабатывании корреляции создается сообщение об отсутствии событий с указанного источника.

- Формирование сообщения:
  - `message`: формируется сообщение "Перестали поступать события с источника " + значение `name`, которое указывает на идентификатор элемента, из которого ожидались данные.
  - `dvendor`: поле устанавливается в значение "vector", что соответствует источнику данных.
  - `cnt`: выводится значение счетчика последнего события, что помогает определить объем данных до прекращения поступления.
  - `name`: содержит идентификаторы конвейера и элемента, которые породили событие.
  - `component_id`: отображает идентификатор элемента, предоставляющий конкретный источник данных.

Оповещение

- `trigger_alert`: указывает на то, что по результатам корреляции будет создано оповещение.

2. Выход элемента **Нормализатор** соедините со входом элемента **Коррелятор**.

Конечная точка

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему. Имеет один вход и не имеет выходов.

1. Добавьте на конвейер конечную точку, в которой будут сохраняться события.
2. Выход элемента **Коррелятор** соедините со входом элемента **Конечная точка**.

Сервис оповещений

Сервис оповещений позволяет настраивать связи между правилами корреляции и правилами сегментации для распределения корреляционных событий по оповещениям.

1. Добавьте на конвейер сервис оповещений, чтобы на полученные корреляционные события в системе формировалось оповещение. При создании элемента, используйте системное "Правило сегментации Мониторинг источников (группировка по id)".



Можно создать **Сервис оповещений** без указания правил сегментации, но рекомендуется добавить правило сегментации, чтобы корреляционные события по одному компоненту собирались в отдельное оповещение.

Пример правила сегментации, схема RObject

```
id: 55b22053-0000-420a-0000-2fa5b176b4f5
name: "Правило сегментации Мониторинг источников (группировка по id)"
```

```
type: segmentation_rule
version: 0.0.1
tags:
  - metrics

group_by:
  - correlationRuleName
  - name

alert_name: !vrl |
  "Перестали поступать события из элемента " + to_string!(.name)
```

▼ *Интерпретация*

Данное правило сегментации позволяет формировать оповещения, группируя корреляционные события по правилам и именам элементов. Например, если события перестали поступать из определенного элемента, система сгенерирует оповещение с указанием названия этого элемента.

- `group_by`: определяет поля, по которым будет происходить группировка корреляционных событий. В данном примере используются поля `correlationRuleName` и `name`.
  - `correlationRuleName`: имя правила корреляции.
  - `name`: название элемента.
- `alert_name`: задает шаблон для формирования имени оповещения. В данном примере используется VRL для создания строки, включающей значение поля `name`.
  - `"Перестали поступать события из элемента "`: текстовая часть сообщения.
  - `to_string!(.name)`: динамическая часть сообщения, которая добавляет значение поля `name` в строку.

2. Выход элемента **Коррелятор** соедините со входом элемента **Сервис оповещений**.

Включение коллектора и установка конфигурации конвейера



Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы метрики начали поступать на конвейер и обрабатываться:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

Рекомендации по настройке

Параметры `ttl`, `count` и период сбора метрик влияют на чувствительность и время реакции системы. Важно настроить их в соответствии с конкретными требованиями, чтобы обеспечить баланс между чувствительностью системы и устойчивостью к ложным срабатываниям.

Параметр `ttl`

`ttl` (time-to-live) определяет максимальное время, в течение которого промежуточные результаты корреляционных операций сохраняются в памяти. Значение `ttl` должно соответствовать максимальному времени между ожидаемыми событиями, с учетом допустимой погрешности для обработки возможных задержек.

Чем выше значение `ttl`, тем более длительный период времени охватывает корреляционное окно, что может привести к более позднему обнаружению проблем. Напротив, более низкое значение `ttl` может привести к частым ложным срабатываниям, особенно в периоды низкой активности или вариативности данных.

Конфигурация `ttl` должна соответствовать максимальному времени между ожидаемыми событиями с учетом некоторой погрешности для обработки возможных задержек.

Формула для настройки `ttl`:

```
ttl = максимальное время между событиями + допустимая погрешность при реагировании
```

Параметр `count`

`count` определяет количество событий, необходимых для формирования группы в правилах корреляции. Значение `count` должно быть выбрано на основе анализа частоты и объема поступающих данных.

Чем выше значение `count`, тем больше событий требуется для срабатывания корреляции, что может уменьшить количество ложных срабатываний, но при этом привести к пропуску важных корреляций. Напротив, более низкое значение `count` увеличивает чувствительность системы, но также может привести к частым ложным срабатываниям.

Период сбора метрик

При создании элемента сбора метрик можно задать интервал между сборами в секундах, что позволяет контролировать частоту мониторинга активности источников. Этот интервал настраивается в поле **Период сбора метрик** при добавлении сбора метрик на конвейер. Его необходимо адаптировать в соответствии с динамикой изменений в отслеживаемой среде.

При выборе интервала важно учитывать не только частоту изменений в данных, но и влияние этого интервала на производительность системы и ресурсы серверов.

Пример настройки параметров `ttl` и `count`

- Для событий, поступающих с периодичностью раз в час, значение `ttl` следует установить на 2 часа для некритичных событий и 1 час 2 минуты для критичных событий.
- Для событий, поступающих каждую секунду, значение `ttl` можно установить на 1 минуту.

Выбор значения параметра `count` зависит от частоты событий:

- Для редких событий значение `count` можно установить на 1, чтобы система реагировала на каждое событие.

- Для частых событий значение `count` можно установить на 5 или выше, чтобы уменьшить количество ложных срабатываний.

Аудит системных событий

О сервисе аудита событий

Сервис аудита предназначен для сбора, анализа и хранения данных о действиях пользователей и событиях в системе. Он обеспечивает контроль за безопасностью, позволяя администраторам выявлять, хранить и отслеживать все действия пользователей в системе для предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности. С помощью сервиса аудита можно фиксировать такие действия, как входы в систему, создание, удаление и изменение ресурсов системы и конфигурации ее отдельных компонентов.



При большом количестве событий аудита в системе вы можете увеличить ресурсы для их обработки. Подробности приведены в разделе Изменение ресурсов для обработки событий аудита.

Сервис аудита работает таким образом, что каждое действие пользователя в системе идентифицируется как отдельное событие аудита. Это событие затем регистрируется и сохраняется в системное хранилище аудита. Основой для этого хранилища служит системная база данных событий аудита, структура которой определяется системной моделью события аудита.

События аудита систематически записываются в базу данных, обеспечивая возможность их постоянного хранения, систематизации и последующего ретроспективного анализа. События аудита доступны для просмотра в разделе **Поиск**. Доступ к этим данным позволяет просматривать историю действий пользователей, анализировать поведенческие паттерны и, при необходимости, реагировать на подозрительные или несанкционированные действия.

События сервиса аудита можно анализировать и обрабатывать на конвейере с помощью точки входа Audit. Например, вы можете настроить конфигурацию конвейера на выполнение следующих операций:

- Создание оповещений о критических событиях
- Отслеживание действий суперадминистраторов
- Отслеживание действий над объектами

Просмотр списка событий аудита

Чтобы просмотреть события сервиса аудита:

1. Перейдите в раздел **Поиск**.
2. Выберите из выпадающего списка в верхней части раздела **Хранилище событий аудита**.
3. Задайте критерии поиска событий:
    - Для поиска событий по ключевым словам или по RQL-запросам введите запрос в поле поиска.



При установке курсора в поле поиска отображается панель с историей

поиска. Система сохраняет здесь 5 последних поисковых запросов.

- Для поиска событий за период времени выберите из выпадающего списка способ задания периода:
  i. **Последний период** — задайте значение вручную или с помощью кнопок — и +, после чего выберите единицы времени из выпадающего списка. Поиск событий будет выполняться в указанном диапазоне вплоть до нынешнего момента.
  ii. **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск событий.
  iii. **Часто используемые** — выберите в разделе один из предлагаемых периодов.



В нижней части выпадающего списка параметров для определения периода расположена секция **История поиска**. Система сохраняет здесь 5 последних поисковых запросов.

4. Нажмите на кнопку **Поиск**. Список событий аудита, соответствующий заданным критериям, отобразится в таблице. Также над таблицей отобразится количество найденных совпадений.



При необходимости вы можете остановить процесс поиска, нажав на кнопку **Стоп**. В разделе будут отображены только те события, что были найдены до прерывания поиска.

Для событий аудита доступны все стандартные операции над событиями.

Типы событий аудита

Тип события аудита зависит от действия пользователя, а также от раздела системы, в котором это действие было выполнено.

Система отслеживает следующие события аудита:

- Системные события аудита
- События аудита пространств
- События аудита работы с событиями ИБ
- События аудита менеджера агентов
- События аудита агентов

Системные события аудита

Группа событий	Тип события
Аутентификация	• Вход пользователя • Выход пользователя • Неуспешная попытка входа • Попытка входа заблокированного пользователя • Пользователь заблокирован по причине превышения попыток входа
Настройка средств аутентификации	• Смена пароля пользователя • Изменение настроек политики требований к паролю • Изменение настроек политики требований к входу
Пользователи	• Создание пользователя • Изменение пользователя • Блокировка пользователя • Разблокировка пользователя • Удаление пользователя
Группы пользователей	• Создание группы пользователей • Изменение группы пользователей • Удаление группы пользователей • Синхронизация группы пользователей
Тенанты	• Создание тенанта • Блокировка тенанта • Разблокировка тенанта • Изменение тенанта • Удаление тенанта
Роли	• Создание роли • Изменение роли • Удаление роли
LDAP-соединения	• Создание LDAP-соединения • Изменение LDAP-соединения • Удаление LDAP-соединения • Синхронизация пользователей домена

События аудита пространств

Группа событий	Тип события
Пространства	• Создание пространства • Изменение пространства • Удаление пространства
Сервисы	• Запуск сервиса • Остановка сервиса

События аудита работы с событиями ИБ

Группа событий	Тип события
Оповещения	• Закрытие всех открытых оповещений • Создание оповещения • Изменение оповещения • Массовое изменение оповещений
Комментарии	• Создание комментария к оповещению • Изменение комментария к оповещению • Удаление комментария к оповещению
Дашборды	• Создание дашборда • Изменение дашборда • Удаление дашборда • Импорт дашборда
Виджеты	• Создание виджета • Изменение виджета • Удаление виджета
Активные списки	• Создание активного списка • Изменение активного списка • Удаление активного списка
Записи активных списков	• Создание записи активного списка • Изменение записи активного списка • Удаление записи активного списка • Импорт записей активного списка

Группа событий	Тип события
Модели событий	• Создание модели события • Изменение модели события • Удаление модели события • Включение модели события • Выключение модели события • Импорт модели события
Базы данных событий	• Создание базы данных • Изменение базы данных • Удаление базы данных
Хранилища событий	• Создание хранилища событий • Изменение хранилища событий • Удаление хранилища событий
Драйверы БД	• Создание драйвера БД • Изменение драйвера БД • Удаление драйвера БД
Секреты	• Создание секрета • Изменение секрета • Удаление секрета
Интеграции	• Создание интеграции • Изменение интеграции • Удаление интеграции • Включение интеграции • Выключение интеграции
Отправка по интеграции	• Отправка оповещений по интеграции • Отправка отчетов по интеграции • Отправка политик мониторинга по интеграции
Шаблоны отчета	• Создание шаблона отчета • Изменение шаблона отчета • Удаление шаблона отчета
Отчеты	• Генерация отчета • Удаление отчета

Группа событий	Тип события
Элементы экспертизы	• Публикация и изменение элемента экспертизы • Создание черновика элемента экспертизы • Удаление элемента экспертизы • Включение элемента экспертизы • Выключение элемента экспертизы
Каталоги	• Создание каталога • Изменение каталога • Перенос элемента экспертизы
Коллекторы	• Создание коллектора • Изменение коллектора • Удаление коллектора • Включение коллектора • Выключение коллектора • Импорт коллектора
Шины	• Создание шины • Изменение шины • Удаление шины
Обогащения	• Добавление обогащения • Изменение обогащения • Удаление обогащения
Конвейеры	• Создание конвейера • Изменение конвейера • Удаление конвейера • Создание копии конвейера • Импорт конвейера • Сброс метрик конвейера
Конфигурации конвейеров	• Установка конфигурации • Отключение конфигурации • Изменение конфигурации
Политики мониторинга	• Создание политики мониторинга • Изменение политики мониторинга • Удаление политики мониторинга • Включение политики мониторинга • Выключение политики мониторинга

События аудита менеджера агентов

Раздел	Тип события
Агенты → Агенты	• Отправка команды на выполнение специального действия
Агенты → Группы агентов	• Отправка команды на выполнение специального действия • Изменение конфигурации группы агентов
Настройки → Агенты	• Изменение белого списка IP-адресов

События аудита агентов



События аудита агентов регистрируются на основе действий, выполняющихся на хостах агентов.

Модуль агента	Тип события
`launcher`	• Включение агента • Отключение агента • Получена команда на переустановку агента • Ошибка переустановки агента • Агент успешно переустановлен • Получена команда на удаление агента из менеджера • Ошибка удаления агента из менеджера • Получена команда на удаление агента с хоста • Ошибка удаления агента с хоста • Зафиксировано превышение лимита ОЗУ • Процесс остановлен из-за превышения лимита ОЗУ
`active-response`	• Успешное выполнение действия • Ошибка выполнения действия

▼ *Информация о модулях агента*

Под модулями агента подразумеваются процессы, реализующие ту или иную его функциональность:

- `launcher` — запуск и отключение агента.
- `active-response` — выполнение специальных действий для быстрого реагирования на инциденты или угрозы.

Модель события аудита

Модель события аудита предназначена для хранения событий сервиса аудита.

Сервис аудита позволяет собирать, анализировать и хранить данные о действиях пользователей и событиях в системе. Он обеспечивает контроль за безопасностью системы, позволяя обнаруживать

возможные угрозы или нарушения безопасности и реагировать на них.

С помощью сервиса аудита можно фиксировать такие действия, как входы в систему, создание, удаление и изменение ресурсов системы и конфигурации ее отдельных компонентов. Каждое действие фиксируется в виде события аудита и сохраняется в специальное системное хранилище. События аудита доступны для просмотра в разделе **Поиск**.

Модель события аудита является встроенной системной моделью и недоступна для изменения или удаления. Она отображается в разделе **Ресурсы → Модели событий**.

При необходимости вы можете создать собственную (пользовательскую) модель события на базе системной модели события аудита.

Поля модели события аудита

Система заполняет поля событий аудита автоматически. Часть полей заполняется только для определенных типов событий и может не содержать значений. Остальные поля универсальны и заполняются независимо от типа события.

Поля модели событий аудита

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`id`	UUID	Да	Уникальный идентификатор события аудита. Генерируется системой автоматически.
`sourceIp`	IPv6	Да	IP-адрес источника, от которого получено базовое событие.
`tenantId`	LCString	Да	Идентификатор тенанта.
`art`	DateTime	Да	Дата и время получения события системой-источником или агентом.
`raw`	String	Да	Текст базового (сырого) события.
`collectorId`	LCString	Да	Идентификатор коллектора, получившего событие.
`timestamp`	DateTime	Да	Дата и время получения события коллектором.
`type`	Enum	Да	Тип события: 4 — событие аудита.
`rt`	DateTime	Да	Дата и время получения устройством сведений о наблюдаемом событии.
`eventType`	LCString	Да	Тип события аудита.
`moduleKey`	LCString	Да	Раздел системы, в котором совершено действие.

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`eventRiskLevel`	Enum	Да	Уровень угрозы события: • 1 — отладочный (`DEBUG`); • 2 — низкий (`LOW`); • 3 — средний (`MEDIUM`); • 4 — высокий (`HIGH`); • 5 — критический (`CRITICAL`); • 6 — фатальный (`FATAL`); • 7 — аварийный (`ACCIDENT`).
`dvc`	IPv6	Нет	IPv6-адрес устройства, с которого получено событие.
`suser`	JSON-строка	Нет	Информация о пользователе, который совершил действие. Примеры сведений: • ID учетной записи. • ФИО учетной записи. • Наличие прав администратора. • Статус учетной записи.
`data`	JSON-строка	Да	Информация о сущности, над которой совершено действие. Примеры сведений: • ID сущности. • Тип сущности. • Название сущности. • Версия сущности. • Длительность блокировки. • Количество неуспешных попыток ввода. • Максимальное время действия пароля. • Минимальная длина пароля. • Способ аутентификации. • Тип события безопасности.

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`meta`	JSON-строка	Да	Информация о событии аудита. Примеры сведений: • Тип события. • Результат выполнения действия: успешно или неуспешно. • Тип действия: создание, изменение, удаление, запуск, включение, остановка, выключение.

Создание оповещений о критических событиях

Сервис аудита в системе позволяет собирать, анализировать и хранить данные о действиях пользователей и событиях в системе. Эти события можно доставлять на конвейер с помощью точки входа Audit и настраивать их обработку в соответствии с конфигурацией конвейера.

Например, можно настроить конфигурацию конвейера так, чтобы при обнаружении действий, критичных для организации, в системе генерировались соответствующие оповещения.

Данное руководство описывает процесс настройки конвейера для создания оповещений при обнаружении критических действий.

О создании оповещений о критических событиях

Для настройки конвейера с генерацией оповещений о критических событиях в системе понадобятся следующие элементы:

1. **Точка входа Аудит**: обеспечивает непрерывный сбор событий сервиса аудита с выбранными уровнями угрозы.
2. **Коррелятор**: генерирует корреляционные события при обнаружении опасных действий в системе.
3. **Конечная точка**: отправляет обработанные и коррелированные события в хранилища данных.
4. **Сервис оповещений**: позволяет настраивать связи между правилами корреляции и правилами сегментации для эффективного распределения корреляционных событий по оповещениям и их отправки через почтовые интеграции.

Таким образом, настройка конвейера для создания оповещений о событиях сервиса аудита включает следующие шаги:

1. Добавление конвейера
2. Настройка конфигурации конвейера
    a. Шаг 1. Точка входа Audit

b. Шаг 2. Корреляция данных

c. Шаг 3. Конечная точка

d. Шаг 4. Настройка отправки оповещений

e. Шаг 5. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, в котором будет выполняться настройка конвейера. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система добавит конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера с помощью кнопки ⌄ в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Шаг 1. Точка входа Audit

Точка входа Audit — это элемент конвейера, который собирает события сервиса аудита и доставляет их на конвейер для дальнейшей обработки и анализа. Точка входа имеет один выход и не имеет входов.

С помощью точки входа Audit можно отслеживать действия пользователей в системе для предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности.

Добавьте на конвейер точку входа типа Audit. При создании выберите необходимые уровни угрозы для фильтрации входящего потока событий аудита. Чтобы получать события с уровнем угрозы не ниже критического, выберите следующие варианты:

- **Критический** — уровень угрозы 5 (`CRITICAL`);
- **Фатальный** — уровень угрозы 6 (`FATAL`);
- **Аварийный** — уровень угрозы 7 (`ACCIDENT`).

Шаг 2. Корреляция данных

Чтобы события, поступающие на конвейер, коррелировались, необходимо добавить на конвейер коррелятор.

Чтобы добавить коррелятор, необходимо установить в нем правило корреляции. Правила корреляции определяют, как коррелятор будет обрабатывать входящие данные и формировать на их основе корреляционные события.

Создание правила корреляции

Правила корреляции в системе позволяют обнаруживать и анализировать взаимосвязанные события, соответствующие определенным правилам, написанным на языке описания преобразований и корреляций. Эти правила также могут инициировать действия после обнаружения, такие как создание корреляционных событий и оповещений.

Создайте правило корреляции с необходимыми настройками. Например:

Пример правила корреляции, схема RObject

```
id: "audit_alert_correlation_rule"
name: "Правило корреляции Оповещения о событиях аудита"
type: correlation_rule
version: 1.0.0

severity: critical

aliases:
  audit_events:
    filter: !vrl |
      true

select:
  alias: audit_events

on_correlate: !vrl |
  name = to_string!(%audit_events.meta.name)
  eventRiskLevel = to_string!(%audit_events.eventRiskLevel)
  .message = "Обнаружено действие [{{name}}] с уровнем угрозы {{eventRiskLevel}}"
```

```
trigger_alert: true
```

▼ *Интерпретация*

Данное правило корреляции позволяет отслеживать поступающие события аудита. Коррелятор генерирует корреляционное событие на каждое событие, доставленное на конвейер точкой входа.

Логика фильтрации (aliases)

- Псевдоним `audit_events`: охватывает все поступающие события.

Соединение потоков событий (select)

- `select`: выбор псевдонима для дальнейшей обработки. В данном случае используется `audit_events`.

Пост-корреляционная логика (on_correlate)

При срабатывании корреляции создается сообщение об обнаружении действия с обозначением его уровня угрозы.

- Формирование сообщения:
  - `name`: устанавливается значение поля `.meta.name`, в котором содержится название совершенного действия.
  - `eventRiskLevel`: устанавливается значение поля `.eventRiskLevel`, в котором содержится информация об уровне угрозы действия.
  - `message`: формируется сообщение вида `"Обнаружено действие [{{name}}] с уровнем угрозы {{eventRiskLevel}}"`.

Оповещение

- `trigger_alert`: указывает на то, что по результатам корреляции будет создано оповещение.

Добавление коррелятора

Коррелятор позволяет связывать различные события и создавать комплексные правила на их основе. На входе применяет правила корреляции к потоку событий. На выходе выдает корреляционные события.

1. Добавьте на конвейер коррелятор. При добавлении элемента используйте созданное правило корреляции.
2. Выход из элемента **Точка входа** соедините со входом элемента **Коррелятор**.

Шаг 3. Конечная точка

Конечная точка пересылает событие в хранилище событий. Имеет один вход и не имеет выходов.

1. Добавьте на конвейер конечную точку, в которой будут сохраняться события.
2. Соедините выходы из элементов **Точка входа** и **Коррелятор** со входом элемента **Конечная точка**. Это необходимо для сохранения исходных и корреляционных событий в одном хранилище.

Шаг 4. Настройка отправки оповещений

Чтобы корреляционные события, получаемые после срабатывания правил корреляции, генерировали оповещения, необходимо добавить на конвейер сервис оповещений.

Чтобы добавить сервис оповещений, необходимо создать и установить на нем правило сегментации, согласно которому корреляционные события будут распределяться по оповещениям.

Создание правила сегментации

Правила сегментации в системе позволяют управлять потоком корреляционных событий, обеспечивая их агрегацию, классификацию и преобразование в оповещения согласно заданным условиям.

Создайте правило сегментации с необходимыми настройками. Например:

Пример правила сегментации, схема RObject

```
id: "audit_alert_segmentation_rule"
name: "Правило сегментации Оповещения о событиях аудита"
type: segmentation_rule
version: 1.0.0
group_by:
  - message
alert_name: !vrl |
  to_string!(.message)
```

▼ *Интерпретация*

Данное правило сегментации позволяет формировать оповещения, группируя корреляционные события по пользователю, который совершил действие. Например, если подозрительные действия совершает конкретный пользователь, система сгенерирует оповещение с указанием информации об этом пользователе.

- `group_by`: определяет поля, по которым будет происходить группировка корреляционных событий. В данном примере используется поле `suser`.
  - `message`: сообщение корреляционного события.
- `alert_name`: задает шаблон для формирования имени оповещения. В данном примере в качестве названия оповещения используется сообщение корреляционного события из поля `message`.

Добавление сервиса оповещений

Сервис оповещений позволяет настраивать связи между правилами корреляции и правилами сегментации для распределения корреляционных событий по оповещениям.

1. Добавьте на конвейер сервис оповещений, чтобы на полученные корреляционные события в системе формировалось оповещение. При добавлении элемента используйте созданное правило сегментации.



Можно создать **Сервис оповещений** без указания правил сегментации, но

рекомендуется добавить правило сегментации, чтобы корреляционные события по одному компоненту собирались в отдельное оповещение.

2. Выход из элемента **Коррелятор** соедините со входом элемента **Сервис оповещений**.

Шаг 5. Включение коллектора и установка конфигурации конвейера



Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события аудита начали поступать на конвейер и обрабатываться:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

   При обнаружении действий критического, фатального и аварийного уровней угрозы, в системе будут генерироваться оповещения.

Отслеживание действий суперадминистраторов

Сервис аудита в системе позволяет собирать, анализировать и хранить данные о действиях пользователей и событиях в системе. Эти события можно доставлять на конвейер с помощью точки входа Audit и настраивать их обработку в соответствии с конфигурацией конвейера.

Например, можно настроить конфигурацию конвейера так, чтобы события пользователей с ролью **Суперадминистратор** направлялись в отдельное хранилище или внешнюю систему. Это позволит отслеживать действия пользователей с привилегированными правами для выявления потенциальных злоупотреблений или ошибок, которые могут привести к утечке данных.

Данное руководство описывает процесс настройки конвейера для отслеживания действий суперадминистраторов.

Об отслеживании действий суперадминистраторов

Для настройки конвейера с отслеживанием действий суперадминистраторов понадобятся следующие элементы:

1. **Точка входа Аудит**: обеспечивает непрерывный сбор событий сервиса аудита с выбранными уровнями угрозы.
2. **Фильтр**: отсеивает события, не удовлетворяющие условию фильтрации.
3. **Конечная точка**: отправляет обработанные и коррелированные события в хранилища данных и внешние системы.

Таким образом, настройка конвейера для отслеживания действий суперадминистратора включает следующие шаги:

1. Добавление конвейера
2. Настройка конфигурации конвейера
   a. Шаг 1. Точка входа Audit

b. Шаг 2. Фильтрация данных

c. Шаг 3. Конечная точка

d. Шаг 4. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, в котором будет выполняться настройка конвейера. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система добавит конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера с помощью кнопки ⌄ в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Шаг 1. Точка входа Audit

Точка входа Audit — это элемент конвейера, который собирает события сервиса аудита и доставляет их на конвейер для дальнейшей обработки и анализа. Точка входа имеет один выход и не имеет входов.

С помощью точки входа Audit можно отслеживать действия пользователей в системе для

предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности.

Добавьте на конвейер точку входа типа Audit. При создании выберите необходимые уровни угрозы для фильтрации входящего потока событий аудита.

Шаг 2. Фильтрация данных

Чтобы фильтровать входящие события по наличию роли **Суперадминистратор** у пользователя, выполнившего какое-либо действие в системе, необходимо добавить на конвейер фильтр.

Фильтр — это элемент конвейера, который позволяет сократить объем обрабатываемых событий путем их отсеивания на основе условий, заданных на языке VRL. Имеет один вход и один выход.

1. Добавьте на конвейер фильтр. При добавлении элемента задайте следующее условие фильтрации:

   ```
   .suser.is_super_admin == true
   ```

   Данное условие проверяет, является ли пользователь, совершивший действие, суперадминистратором.

2. Выход из элемента **Точка входа** соедините со входом элемента **Фильтр**.

Шаг 3. Конечная точка

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему. Имеет один вход и не имеет выходов.

1. Добавьте на конвейер конечную точку, в которой будут сохраняться события.
2. Выход из элемента **Фильтр** соедините со входом элемента **Конечная точка**.

Шаг 4. Включение коллектора и установка конфигурации конвейера



Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события аудита начали поступать на конвейер и обрабатываться:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

   События, порожденные действиями суперадминистраторов, будут направляться в заданное хранилище событий или внешнюю систему.

Отслеживание действий над объектами

Сервис аудита в системе позволяет собирать, анализировать и хранить данные о действиях пользователей и событиях в системе. Эти события можно доставлять на конвейер с помощью точки входа Audit и настраивать их обработку в соответствии с конфигурацией конвейера.

Например, можно настроить конфигурацию конвейера так, чтобы в разные хранилища направлялись события аудита разных типов. Это позволит отслеживать определенные действия над различными сущностями в системе для обнаружения несанкционированных действий и поддержания безопасности инфраструктуры организации. Так, например, можно обнаружить изменение или удаление коллектора, критически важного для организации, и своевременно принять необходимые меры.

Данное руководство описывает процесс настройки конвейера для отслеживания действий над объектами системы.



В представленном руководстве термин "объект" используется для обозначения сущностей системы, например, коллекторов, конвейеров, элементов экспертизы, активных списков.

Об отслеживании действий над объектами

Для отслеживания определенных действий над объектами, первоначально требуется создать хранилища событий, в каждом из которых будут храниться события аудита определенного типа. В представленной инструкции будет рассматриваться отслеживание действий с типами "Создание", "Изменение" и "Удаление". Таким образом, для представленного случая понадобятся четыре хранилища: по одному на каждый из трех типов действий и еще одно — для всех остальных.

Для настройки конвейера с отслеживанием действий над объектами понадобятся следующие элементы:

1. **Точка входа Аудит**: обеспечивает непрерывный сбор событий сервиса аудита с выбранными уровнями угрозы.
2. **Маршрутизатор**: направляет поток событий по разным маршрутам в зависимости от условий фильтрации на этих маршрутах.
3. **Конечные точки**: отправляют обработанные события в хранилища данных.

Таким образом, настройка конвейера для отслеживания действий над объектами включает следующие шаги:

1. Добавление конвейера
2. Настройка конфигурации конвейера
    a. Шаг 1. Точка входа Audit
    b. Шаг 2. Маршрутизация данных
    c. Шаг 3. Конечные точки
    d. Шаг 4. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, в котором будет выполняться настройка конвейера. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система добавит конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера с помощью кнопки ⌄ в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Шаг 1. Точка входа Audit

Точка входа Audit — это элемент конвейера, который собирает события сервиса аудита и доставляет их на конвейер для дальнейшей обработки и анализа. Точка входа имеет один выход и не имеет входов.

С помощью точки входа Audit можно отслеживать действия пользователей в системе для предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности.

Добавьте на конвейер точку входа типа Audit. При создании выберите необходимые уровни угрозы для фильтрации входящего потока событий аудита.

Шаг 2. Маршрутизация данных

Чтобы направлять поступающие данные по разным путям обработки, необходимо добавить на конвейер маршрутизатор.

Маршрутизатор — это элемент конвейера, который позволяет на основе заданных фильтров направлять события по разным маршрутам в зависимости от выполнения условий фильтрации.

Имеет один вход и несколько выходов: по одному выходу на каждый настроенный маршрут.

1. Добавьте на конвейер маршрутизатор. При добавлении элемента настройте необходимые маршруты. Например:

   Условие маршрута 1

   ```
   .meta.action_type == "Создание"
   ```

   Условие маршрута 2

   ```
   .meta.action_type == "Изменение"
   ```

   Условие маршрута 3

   ```
   .meta.action_type == "Удаление"
   ```

   Данные условия проверяют поле `action_type`, в котором хранится тип действия, породившего событие, и на основе его значения направляют событие по одному из настроенных маршрутов. В представленном примере учитываются действия типа "Создание", "Изменение" и "Удаление".

   В настройках маршрутизатора установите флажок **Добавить маршрут по умолчанию**, чтобы направлять события, не удовлетворяющие ни одному из заданных условий, в маршрут **По умолчанию**.

2. Выход из элемента **Точка входа** соедините со входом элемента **Маршрутизатор**.

Шаг 3. Конечные точки

Конечная точка пересылает событие в хранилище событий. Имеет один вход и не имеет выходов.

1. Добавьте на конвейер конечные точки, в которые будут сохраняться события. При добавлении элементов выберите необходимые хранилища, куда будут направляться события требуемого типа.

   Количество добавленных конечных точек должно совпадать с количеством маршрутов элемента **Маршрутизатор**.

2. Выходы из маршрутов элемента **Маршрутизатор** соедините со входами элементов **Конечная точка**.

Шаг 4. Включение коллектора и установка конфигурации конвейера



Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события аудита начали поступать на конвейер и обрабатываться:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

События, порождаемые действиями в системе, будут направляться в разные хранилища в зависимости от типа действия.

Мониторинг RQL-запросов и управление ресурсами

Сбор и хранение данных об RQL-запросах осуществляется в разделе **Инструменты** → **Диспетчер запросов**. В этом разделе отображается информация о запросах из разделов **Поиск** и **Дашборды**, времени их выполнения и объеме памяти, выделенной для выполнения. Здесь также можно управлять лимитами памяти, выделяемой для запросов.

По умолчанию в диспетчере хранится информация о запросах, сделанных в течение 7 последних дней. Изменить срок хранения можно с помощью переменной окружения.

Интерфейс раздела

Вкладка Журнал запросов

В верхней части вкладки расположена панель инструментов, которая включает следующие компоненты:

- Кнопка ↑ открывает окно для экспорта истории запросов.
- Кнопка **Фильтр** (▽) позволяет задать критерии, по которым будут отобраны запросы.

В рабочей области отображается таблица найденных вхождений. Таблица содержит следующие столбцы:

- **Время запуска** — дата и время начала выполнения.
- **Статус** — состояние обработки запроса: **Запущен**, **Выполнен** или **Ошибка**.
- **Запустил** — имя учетной записи пользователя, который сделал запрос. При наведении отображается окно с именем и ролью пользователя.
- **Тип запроса** — раздел, где был сделан запрос: **Поиск**, **Дашборды**, **Общая статистика**, **Статистика по полю**, **Оповещения**.
- **Время выполнения** — время выполнения в миллисекундах.
- **Выделенная память** — память, выделенная для выполнения запроса, в мегабайтах с точностью до второго знака после десятичной точки.
- **Время окончания** — дата и время окончания (заполнено, если запрос завершен). Поле скрыто по умолчанию.
- **Теги** — пользовательские теги, указываются в карточке запроса.
- **Запрос RQL** — первые 20 символов запроса на RQL. При наведении курсора мыши отображается окно с полным запросом и кнопка для его копирования.
- **Запрос SQL** — первые 20 символов итогового запроса на SQL с учетом примененных фильтров, ограничений по времени и пагинации. При наведении курсора мыши отображается окно с полным запросом и кнопка для его копирования. Поле скрыто по умолчанию.
- **Ошибка** — описание ошибки (заполнено, если при выполнении запроса произошла ошибка). Поле скрыто по умолчанию.

При работе с таблицей доступны следующие операции:

- Фильтрация списка запросов. В параметрах фильтрации доступны следующие опции:
  - **Время запуска** — временной диапазон, в котором был сделан запрос. Начало и конец диапазона можно ввести вручную или задать с помощью панели выбора даты и времени.
  - **Статус** — состояние выполнения, выбирается из выпадающего списка.
  - **Запустил** — пользователь, который запустил запрос, выбирается из выпадающего списка.
  - **Тип запроса** — раздел, где сделан запрос, выбирается из выпадающего списка.
  - **Минимальное время выполнения** — отображает запросы с длительностью выполнения, равной или большей заданному времени, указывается вручную в миллисекундах.
  - **Выделенная память** — отображает запросы, для которых выделено количество памяти, равное или большее заданного, указывается вручную в мегабайтах.
  - **Теги** — пользовательские теги. Вводятся вручную и применяются с помощью клавиши ENTER.
- Сортировка по дате и времени начала, времени выполнения запроса и выделенной памяти.

  По умолчанию таблица упорядочена по времени начала выполнения от самых новых запросов к более старым.
- Обновление таблицы.
- Настройка отображения таблицы.

При выборе отдельного запроса в правой части рабочей области отображается карточка с детальной информацией о нем.

Вкладка Управление ресурсами

В верхней части вкладки расположена панель инструментов, которая включает следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый лимит памяти.
- Поле поиска позволяет искать лимиты по названию.
- Кнопка **Фильтр** ( ) позволяет задать критерии, по которым будут отобраны лимиты.

В рабочей области отображается таблица найденных вхождений. Таблица содержит следующие столбцы:

- **Название** — имя, используемое для идентификации лимита в системе.
- **Статус** — состояние лимита: **Включен** или **Выключен**.
- **Роль** — список ролей, для которых установлен лимит.
- **Лимит** — лимит выделяемой памяти в мегабайтах.
- **Дата создания** — дата и время создания лимита.
- **Дата изменения** — дата и время изменения лимита.

При работе с таблицей доступны следующие операции:

- Фильтрация списка лимитов по ролям, для которых установлены лимиты.
- Сортировка таблицы по названию, объему, датам создания и изменения лимита.
- Настройка отображения таблицы.

При выборе отдельного лимита в правой части рабочей области отображается карточка с детальной информацией о нем.

Просмотр запроса

Чтобы просмотреть сведения о запросе, нажмите на строку запроса в списке. В правой части экрана отобразится карточка с подробной информацией.

В верхней части карточки отображается идентификатор запроса.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит следующие поля:

- **Время запуска** — дата и время начала выполнения.
- **Статус** — состояние обработки запроса: **Запущен**, **Выполнен** или **Ошибка**.
- **Ошибка** — сообщение об ошибке (поле заполнено, если запрос завершился с ошибкой).
- **Запустил** — имя и роль учетной записи пользователя, который сделал запрос.
- **Время выполнения** — время выполнения с точностью до миллисекунд.
- **Выделенная память** — память, выделенная для выполнения запроса, в мегабайтах.
- **Тип запроса** — раздел, где был сделан запрос: **Поиск**, **Дашборды**, **Общая статистика**, **Статистика по полю**, **Оповещения**.
- **Время окончания** — дата и время окончания (поле заполнено, если запрос завершен).
- **Теги** — теги запроса, указываются вручную.
- **Запрос RQL** — запрос на языке RQL и кнопка для его копирования.
- **Запрос SQL** — запрос на SQL и кнопка для его копирования.

Экспорт истории запросов

Чтобы экспортировать историю запросов:

1. Нажмите на кнопку **Экспорт** ( ) на панели инструментов. Отобразится окно экспорта истории.
2. Нажмите на кнопку **Экспортировать**. Система загрузит историю запросов на устройство пользователя в формате CSV и отобразит соответствующее уведомление.

Работа с лимитами

Лимиты регулируют объем потребления оперативной памяти для определенных ролей пользователей. Лимит можно установить для нескольких ролей, но для каждой роли допускается только один лимит.

Лимиты применяются в следующем порядке:

- При выполнении запроса система проверяет лимит роли пользователя.
- Если у пользователя есть несколько ролей, для которых установлены лимиты, используется наибольший из лимитов.
- Если для ролей пользователя нет включенных лимитов, система использует ограничение на максимальное потребление памяти из настроек ClickHouse.

Доступные операции над лимитами:

- Создание лимита
- Просмотр лимита
- Включение и выключение лимита
- Изменение лимита
- Удаление лимита

Создание лимита



Создавать лимиты могут администраторы, инженеры, пользователи, которые имеют соответствующее разрешение в составе своих ролей, а также пользователи со статусом **Суперадминистратор**.

Чтобы создать лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания лимита.
3. Введите название лимита.
4. Выберите из выпадающего списка роли, для которых устанавливается лимит.
5. Введите значение лимита в мегабайтах (по умолчанию — 1024 Мб).
6. Нажмите на кнопку **Создать**. Система создаст лимит и отобразит соответствующее уведомление. Новый лимит появится в разделе **Инструменты** → **Диспетчер запросов** на вкладке **Управление ресурсами**.



Если для каких-либо из указанных ролей уже заданы лимиты, система отобразит уведомление об ошибке.

Просмотр лимита

Чтобы просмотреть лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.
2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией о лимите.

В верхней части карточки отображается название лимита.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит следующие элементы:

- Переключатель статуса лимита.
- **ID** — идентификатор лимита.
- **Название** — имя, используемое для идентификации лимита в системе.
- **ID тенанта** — идентификатор тенанта.
- **Статус** — состояние лимита: **Включен** или **Выключен**.
- **Роль** — роли, для которых установлен лимит.
- **Лимит** — лимит выделяемой памяти в мегабайтах.
- **Дата создания** — дата и время создания лимита.
- **Создал** — имя учетной записи пользователя, который создал лимит.
- **Дата изменения** — дата и время изменения лимита.
- **Изменил** — имя учетной записи пользователя, который изменил лимит.

Включение и выключение лимита

Чтобы лимит учитывался при выполнении RQL-запросов, его нужно включить.

Чтобы включить лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.
2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией об этом лимите.
3. Установите переключатель состояния лимита в верхней части карточки в активное положение. Отобразится окно подтверждения.
4. Нажмите на кнопку **Включить**. Система включит лимит и отобразит соответствующее уведомление. Лимит станет действовать при выполнении запросов.

Чтобы выключить действующий лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.
2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией об этом лимите.
3. Установите переключатель в верхней части карточки в неактивное положение. Отобразится окно подтверждения.
4. Нажмите на кнопку **Выключить**. Система выключит лимит и отобразит соответствующее уведомление. Лимит перестанет действовать при выполнении запросов.

Изменение лимита

Вы можете изменить лимит следующими способами:

- через меню действий в карточке лимита;
- через изменение полей в карточке лимита.

Удаление лимита

Вы можете удалить лимит через меню действий в его карточке.

Авторизация

Перед входом в систему запросите у системного администратора:

- адрес для входа в веб-интерфейс продукта;
- логин и пароль вашей учетной записи пользователя.

По умолчанию для первого входа в систему используются учетные данные **admin** /**admin**.

При переходе по адресу на экране отобразится страница авторизации, которая содержит поля для ввода логина и пароля учетной записи пользователя, а также выпадающий список для выбора языка интерфейса в системе.

Вход в систему

Чтобы выполнить вход в систему:

1. В адресной строке браузера введите ссылку для входа в веб-интерфейс системы. На экране отобразится страница авторизации.
2. Укажите логин и пароль учетной записи пользователя и нажмите на кнопку **Войти**.

Если в систему добавлены пользователи с таким же логином из других доменов, укажите домен в поле логина. В противном случае домен можно не указывать.

Если срок действия пароля истекает менее, чем через семь дней или пароль не соответствует текущим требованиям, система предоставит ссылку для его смены.

Если вход в систему выполнен успешно, в окне браузера отобразится стартовый раздел системы. В правом верхнем углу экрана отобразится текущая учетная запись.

Выход из системы

Чтобы выйти из системы:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Выйти из системы** (). Система завершит сеанс, и на экране отобразится страница авторизации.

Интерфейс

После успешной авторизации в браузере отобразится окно стартового раздела веб-интерфейса системы.

Основные элементы интерфейса

В верхней правой части интерфейса отображаются следующие кнопки:

- 🔍 — позволяет осуществить глобальный поиск по объектам системы в рамках текущего тенанта.

  ▼ *Сущности глобального поиска*

  Поиск охватывает следующие сущности:

  - Сущности аналитики данных:
    - дашборды;
    - сформированные отчеты;
    - шаблоны отчетов.
  - Ресурсы:
    - активные списки;
    - базы данных;
    - интеграции;
    - коллекторы;
    - конвейеры;
    - модели событий;
    - секреты;
    - хранилища событий.
  - Экспертиза:
    - глобальные функции;
    - правила агрегации;
    - правила корреляции;
    - правила нормализации;
    - правила сегментации;
    - схемы активных списков;
    - схемы дашбордов;
    - схемы шаблонов отчетов;
    - таблицы обогащения.
  - Инструменты:

- политики аудита источников;
- политики управления доступом;
- сниппеты.



Кнопка отображается, только если открыт один из разделов главного меню. При переходе в разделы настроек системы кнопка скрывается.

- — открывает меню со следующими настройками:
  - **Настройки сервера** () — открывает окно настроек для подключения к другим продуктам R-Vision.
  - **Документация** () — открывает руководство по эксплуатации системы.
  - **Настройки профиля** () — открывает страницу настроек профиля текущего пользователя.
  - **Тема оформления** () — переключает тему интерфейса: темная, светлая или синяя.
  - **Выйти из системы** () — прекращает текущий сеанс пользователя и осуществляет его выход из системы.

В левой части окна расположено **главное меню** с основными разделами системы. В верхней части главного меню отображается название продукта, его текущая версия и меню выбора тенанта. В нижней части главного меню расположена кнопка для перехода в **настройки системы**, а также кнопка со стрелкой для его сворачивания и разворачивания.

При нажатии на название раздела система откроет этот раздел и отобразит его содержимое в **рабочей области**, расположенной в центральной части окна.

В верхней части рабочей области расположена **панель инструментов**. Вид панели инструментов зависит от открытого раздела. На панели инструментов могут отображаться:

- название раздела или элемента системы;
- выпадающие списки, кнопки и поля для управления разделом;
- информация о разделе.

Разделы могут отображать таблицу элементов. Набор возможностей для работы с таблицей зависит от открытого раздела.

Разделы главного меню

В системе доступны следующие разделы:

- **Дашборды** — раздел аналитики по зарегистрированным событиям.
- **Оповещения** — раздел для работы с оповещениями о событиях в системе.
- **Отчеты** — раздел для работы с отчетами в системе.
- **Поиск** — раздел для работы с событиями, зарегистрированными в системе, и записями активных списков.
- Группа разделов **Ресурсы**:

- **Активные списки** — раздел создания и настройки активных списков.
- **Коллекторы** — раздел создания и настройки коллекторов.
- **Модели событий** — раздел создания и настройки модели данных.
- **Базы данных событий** — раздел регистрации и настройки баз данных.
- **Хранилища событий** — раздел создания и настройки хранилищ событий.
- **Интеграции** — раздел управления интеграциями с другими системами.
- **Секреты** — раздел создания и настройки секретов для безопасного хранения чувствительных данных.
- **Драйверы БД** — раздел создания и настройки драйверов для подключения внешних баз данных.

- **Экспертиза** — раздел создания и настройки правил обработки событий безопасности.
- Группа разделов **Инструменты**:
  - **VRL-песочница** — раздел написания и проверки VRL-программ.
  - **Сниппеты** — раздел создания и настройки сниппетов для хранения шаблонов VRL-кода.
  - **Аудит источников** — раздел для контроля состояния источников событий с помощью политик аудита источников.
  - **Мониторинг** — раздел наблюдения за состоянием кластера Kubernetes.
  - **Диспетчер запросов** — раздел для сбора и хранения данных об RQL-запросах.
  - **Политики управления доступом к событиям** — раздел настройки политик доступа пользователей к событиям в хранилищах на основе их ролей.
  - **Покрытие MITRE ATT&CK** — раздел для оценки покрытия техник и сабтехник MITRE ATT&CK правилами корреляции.

  Если в настройках сервера заданы адреса R-Vision SOAR и/или R-Vision UEBA, в списке также отображаются разделы **SOAR** и/или **UEBA** соответственно.

- Группа разделов **Агенты**:
  - **Агенты** — раздел управления агентами.
  - **Группы агентов** — раздел для работы с группами агентов.
  - **Шлюзы** — раздел для работы со шлюзами, с помощью которых агенты могут подключаться к кластерам.

Разделы настроек системы

Настройки системы включают следующие разделы:

- Группа разделов **Управление доступом**:
  - **Группы пользователей** — раздел для управления группами пользователей.
  - **Пользователи** — раздел для создания и управления пользователями.

    - **Роли** — раздел для управления ролями пользователей.
    - **Тенанты** — раздел для управления тенантами.
    - **Политики входа и пароля** — раздел для управления аспектами безопасности пароля и правилами доступа к системе.
- Группа разделов **Пространства и сервисы**:
    - **Пространства** — раздел для управления подключениями к пространствам.
    - **Сервисы** — раздел для управления сервисами в пространствах.
- **LDAP-соединения** — раздел для управления LDAP-соединениями в системе.
- **Лицензия** — раздел для управления лицензией в системе.
- **Агенты** — раздел для настройки работы агентов.

Документация

Чтобы просмотреть руководство по эксплуатации из веб-интерфейса системы:

1. Откройте меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Документация** (). Система отобразит страницу руководства по эксплуатации системы на новой вкладке браузера.

Настройки профиля

Чтобы открыть настройки профиля:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.

Настройки профиля сгруппированы по следующим секциям:

- **Личные данные** — отображает основные сведения о пользователе: логин, ФИО, должность и описание.
- **Общие настройки интерфейса** — предлагает опции для выбора языка интерфейса системы и формата отображения дат.
- **Смена пароля** — предназначена для изменения текущего пароля пользователя.
- **Доступные разрешения** — отображает таблицу всех разрешений пользователя. Таблица содержит следующие столбцы:
    - **Разрешение** — название разрешения.
    - **Раздел** — раздел системы, который затрагивается разрешением. Путь до раздела указывается через /.
    - **Модуль** — функциональный модуль платформы, к которому относится разрешение.
    - **Тенанты и организации** — список тенантов, в рамках которых выдано разрешение.

  При работе с таблицей разрешений доступны следующие операции:

- Поиск разрешений по всем полям.
- Сортировка разрешений по полям **Разрешение** и **Модуль**.



Для учетных записей со статусом **Суперадминистратор** список разрешений не приводится, поскольку они обладают всеми доступными в системе разрешениями. Для таких пользователей в секции отображается только статус **Суперадминистратор**.

Настройки интерфейса

В системе доступны следующие настройки интерфейса:

- Выбор языка
- Выбор формата даты
- Выбор формата даты и времени событий
- Выбор часового пояса

Выбор языка

Система поддерживает два языка интерфейса: **русский** и **английский**.

Чтобы изменить язык интерфейса:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В поле **Язык** секции **Общие настройки интерфейса** выберите из один из доступных языков: **Русский** или **Английский**.
4. Нажмите на кнопку **Сохранить**. Система автоматически сменит язык интерфейса на выбранный.



Выбор языка также доступен на странице авторизации системы.

Выбор формата даты

В системе преднастроены два формата дат и времени:

- `YYYY-MM-DDThh:mm:ss+/-hh:mm`;
- `DD.MM.YYYY hh:mm:ss`.



При использовании формата `DD.MM.YYYY hh:mm:ss` время указывается относительно часового пояса пользователя.

Здесь:

- `YYYY-MM-DD` — дата в формате "год-месяц-день".
- `DD.MM.YYYY` — дата в формате "день.месяц.год".

- `T` — начало обозначения времени.
- `hh:mm:ss` — время в формате "час:минута:секунда".
- `+hh:mm` или `-hh:mm` — разница между часовым поясом пользователя и часовым поясом UTC+0.

По умолчанию в системе используется формат `DD.MM.YYYY hh:mm:ss`.

Чтобы изменить формат даты:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В поле **Формат даты и времени** секции **Общие настройки интерфейса** выберите из один из доступных форматов.
4. Нажмите на кнопку **Сохранить**. Система автоматически сменит формат отображения дат и времени на выбранный.

Выбор формата даты и времени событий

В системе преднастроены следующие форматы даты и времени событий в разделе **Поиск** и на дашбордах.

- `YYYY-MM-DDThh:mm:ss+/-hh:mm`;
- `DD.MM.YYYY hh:mm:ss.sss`.
- `DD.MM.YYYY hh:mm:ss`.



- При использовании формата `DD.MM.YYYY hh:mm:ss` или `DD.MM.YYYY hh:mm:ss.sss` время указывается относительно часового пояса пользователя.
- Если при использовании формата `DD.MM.YYYY hh:mm:ss.sss` в хранилище событий присутствуют поля с типом `Datetime` или `DateTime64` без поддержки миллисекунд, то значения этих полей будут отображаться с нулевыми миллисекундами, например: `15.05.2024 06:28:15.000`.

Здесь:

- `YYYY-MM-DD` — дата в формате "год-месяц-день".
- `DD.MM.YYYY` — дата в формате "день.месяц.год".
- `T` — начало обозначения времени.
- `hh:mm:ss` — время в формате "час:минута:секунда".
- `sss` — миллисекунды.
- `+hh:mm` или `-hh:mm` — разница между часовым поясом пользователя и часовым поясом UTC+0.



По умолчанию в системе используется формат `DD.MM.YYYY hh:mm:ss.sss`.

Чтобы изменить формат даты:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.

2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В поле **Формат даты и времени для событий** секции **Общие настройки интерфейса** выберите из один из доступных форматов.
4. Нажмите на кнопку **Сохранить**. Система автоматически сменит формат отображения дат и времени на выбранный.

Выбор часового пояса

Время в системе определяется на основе часового пояса, выбранного пользователем.

Чтобы изменить часовой пояс:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В поле **Часовой пояс** секции **Общие настройки интерфейса** выберите необходимый часовой пояс.
4. Нажмите на кнопку **Сохранить**. Система автоматически пересчитает временные метки относительно выбранного часового пояса.



После изменения часового пояса фильтры по времени, настроенные в разделе **Поиск**, перестроятся с учетом нового часового пояса.

Например, если изначально был выбран часовой пояс **GMT+3 Moscow** и настроена фильтрация по времени `00:00:00`, то после изменения часового пояса на **GMT+5 Yekaterinburg** фильтрация данных начнет выполняться по времени `02:00:00`. В таком случае, если требуется, чтобы фильтрация происходила по времени `00:00:00` в текущем часовом поясе, необходимо задать фильтр заново.

Смена пароля



Смена пароля недоступна пользователям из домена.

Чтобы изменить текущий пароль вашей учетной записи:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В секции **Смена пароля** укажите текущий пароль, затем введите новый пароль и повторите его для подтверждения.
4. Нажмите на кнопку **Изменить пароль**. Если новый пароль удовлетворяет заданным требованиям, система сохранит его и отобразит соответствующее уведомление.

Работа с таблицами

Списки объектов системы, как правило, организованы в виде таблиц, сгруппированных по различным системным разделам.

Система предлагает следующие возможности для работы с таблицей:

- Пагинация таблицы
- Изменение ширины столбцов таблицы
- Просмотр элементов таблицы
- Выделение элементов таблицы
- Сортировка элементов таблицы
- Фильтрация элементов таблицы
- Поиск элементов таблицы
- Обновление таблицы
- Настройка отображения таблицы

Набор доступных возможностей зависит от раздела системы.

Пагинация таблицы

Пагинация позволяет оптимизировать работу с таблицей путем структурирования и группировки ее элементов по страницам.

Панель пагинации располагается в нижней части таблицы и содержит следующие компоненты для управления страницами:

- Кнопки ‹ и › позволяют перейти к предыдущей и следующей странице соответственно.
- Кнопки « и » позволяют перейти к первой и последней странице соответственно.
- Поле ввода, отображающее номер открытой страницы, позволяет ввести номер нужной страницы для быстрого перехода к ней.
- Выпадающий список позволяет настроить количество отображаемых на странице элементов: 50, 100, 200 или 500. По умолчанию установлен лимит в 50 элементов.



При изменении количества отображаемых элементов изменяется и общее количество страниц в таблице.

Выбранное количество элементов сохраняется до конца сеанса пользователя.

Изменение ширины столбцов таблицы

По умолчанию все столбцы в таблице имеют одинаковую ширину. Стандартная ширина столбцов не всегда достаточна для корректного отображения их содержимого. Для лучшего отображения данных ширину столбцов можно изменить.

Чтобы изменить ширину столбца таблицы:

1. Наведите курсор на правую границу заголовка нужного столбца. Курсор примет вид двунаправленной стрелки.
2. Перетащите правую границу заголовка в нужном направлении.

Двойным нажатием по правой границе заголовка можно автоматически подогнать ширину столбца под его содержимое.

Просмотр элементов таблицы

Просмотр элемента таблицы позволяет получить детальную информацию об этом элементе: например, его название, идентификатор, тенант, даты создания и изменения.

Просмотр сведений об элементе осуществляется с помощью его карточки. Чтобы открыть карточку, нажмите на нужный элемент в таблице. Способ отображения карточки и ее содержание зависят от раздела системы.

Выделение элементов таблицы

Выделение позволяет выполнять массовые операции над элементами таблицы: например, экспортировать или удалять их.

Есть два способа выделения:

- Чтобы выделить один или несколько элементов, установите флажки напротив этих элементов в левом столбце таблицы.
- Чтобы выделить все элементы на странице, установите флажок в заголовке первого столбца таблицы. При этом будут выделены только элементы текущей страницы. Чтобы выделить большее число элементов, перейдите на следующую страницу и повторите действие.

Выделение сохраняется при переходе между страницами и изменении количества отображаемых элементов.

Общее количество выделенных элементов и доступные операции над выделенными элементами отображаются на панели инструментов.

Сортировка элементов таблицы

Сортировка позволяет упорядочить элементы таблицы по какому-либо столбцу в выбранном направлении.

Чтобы отсортировать таблицу, нажмите на кнопку действий ⋮ в заголовке нужного столбца и выберите направление сортировки:

- **По умолчанию** ( ) — сортировка отсутствует.
- **Возрастание** ( ).
- **Убывание** ( ).

Применить сортировку также можно с помощью аналогичных кнопок в заголовке столбца: последовательное нажатие кнопки меняет направление сортировки.

Фильтрация элементов таблицы

Фильтрация позволяет отобразить в таблице только те элементы, которые удовлетворяют

заданным условиям.

В системе доступны следующие способы фильтрации:

- через боковую панель;
- через выпадающий список.

Способ фильтрации зависит от раздела системы.



Если задан хотя бы один фильтр, система отобразит текущие настройки фильтрации под панелью инструментов. Справа от настроек отобразится кнопка **Сбросить все** (✕) для удаления всех имеющихся фильтров.

Фильтрация через боковую панель

Чтобы отфильтровать таблицу через боковую панель:

1. Нажмите на кнопку **Фильтр** ( ), расположенную над таблицей. В правой части рабочей области откроется панель настройки фильтрации.
2. Задайте настройки фильтрации на панели. В таблице отобразятся только элементы, удовлетворяющие заданным условиям.

Чтобы сбросить условие фильтрации, очистите соответствующее поле с помощью кнопки ✕ . Чтобы сбросить сразу все условия, нажмите на кнопку **Сбросить** в нижней части панели.

Фильтрация через выпадающий список

Чтобы отфильтровать таблицу через выпадающий список, выберите нужный пункт в списке, расположенном над таблицей. В таблице отобразятся только элементы с выбранным критерием.

Чтобы сбросить условие фильтрации, снимите выбор в выпадающем списке с помощью кнопки ✕ .

Поиск элементов таблицы

Поиск позволяет быстро находить в таблице нужные элементы по содержимому их полей.

Чтобы выполнить поиск, введите запрос в строку поиска, расположенную над таблицей, и нажмите клавишу ENTER. В таблице отобразятся элементы, в соответствующих полях которых содержится введенный запрос.



Особенности поиска:

- Поиск осуществляется по вхождению указанных символов в полях элементов.
- При поиске учитывается строгая последовательность введенных символов, вхождение которой проверяется в любом месте строки.
- При поиске не учитывается регистр символов.

Чтобы сбросить поиск, очистите поле поиска вручную или с помощью кнопки ✕ .

Обновление таблицы

Во время работы с таблицей перечень ее элементов может обновиться. При обновлении таблицы изменения не отображаются автоматически в интерфейсе программы.

Чтобы отобразить изменения в таблице, ее требуется обновить. Вы можете обновить таблицу вручную или настроить автоматическое обновление.

Обновление таблицы вручную

Чтобы обновить таблицу вручную, нажмите на кнопку **Обновить** (🔄) в нижней части таблицы. Перечень элементов таблицы будет обновлен.

Автоматическое обновление таблицы

Чтобы настроить регулярное автоматическое обновление таблицы:

1. Нажмите на кнопку **Автообновление** в нижней части таблицы. Отобразится панель настроек автоматического обновления.
2. Задайте настройки автоматического обновления:
   a. Укажите количество единиц времени, по истечении которых будет происходить обновление. Вы можете сделать это вручную или с помощью кнопок — и +.
   b. Выберите из выпадающего списка единицы времени: секунды, минуты, часы.
   c. Установите переключатель **Включить автообновление** в активное положение.

   Перечень элементов таблицы будет автоматически обновляться по истечении указанного периода времени.

Настройка отображения таблицы



Настройки отображения таблицы сохраняются индивидуально для текущего пользователя системы.

Настройки задаются отдельно для каждого раздела. В разделе **Поиск** отображение и порядок полей настраиваются отдельно для каждого хранилища событий и активного списка. Если выполняется поиск по нескольким хранилищам, настройки сохраняются для выбранной комбинации хранилищ.

При необходимости вы можете задать настройки отображения таблицы для повышения удобства работы с ней.

Чтобы настроить внешний вид таблицы:

1. Нажмите на кнопку ⊞ в нижней части таблицы. Система отобразит в правой части экрана панель настроек таблицы.
2. Настройте отображение таблицы:
   a. Выберите высоту строки: стандартную или компактную.
   b. Установите флажки рядом с названиями полей, которые должны отображаться в таблице.

c. Разместите поля в желаемом порядке, перетаскивая их с помощью мыши, чтобы определить порядок их отображения в таблице. Для этого наведите курсор на иконку ═ слева от флажка и перетащите поле в нужную позицию.

Чтобы применить к таблице настройки отображения по умолчанию, нажмите на кнопку **Восстановить по умолчанию**.

Настроить внешний вид таблицы также можно через меню действий в заголовках столбцов:

- Перемещение столбцов таблицы через меню действий
- Скрытие столбцов таблицы через меню действий

Перемещение столбцов таблицы через меню действий

Чтобы переместить столбец, нажмите на кнопку действий ⋮ в его заголовке и выберите направление перемещения:

- **Переместить влево** (←) — меняет местами выбранный столбец и столбец слева.

  Данная опция недоступна для первого столбца в таблице.

- **Переместить вправо** (→) — меняет местами выбранный столбец и столбец справа.

  Данная опция недоступна для последнего столбца в таблице.

Скрытие столбцов таблицы через меню действий

Чтобы скрыть столбец, нажмите на кнопку действий ⋮ в его заголовке и выберите опцию **Скрыть колонку** (📎).

Работа с объектами

Система предлагает следующие способы работы с объектами:

- Изменение объектов
- Удаление объектов

Добавление и просмотр объектов со свойственными им параметрами приведены в описании соответствующих разделов системы.

Изменение объектов

В системе можно изменять параметры объекта следующими способами:

- Изменение через меню действий в карточке объекта
- Изменение полей объекта через кнопку редактирования в карточке
- Изменение объекта через кнопку в карточке
- Изменение объекта через панель инструментов в полноэкранной карточке

- Изменение объекта через панель инструментов
- Массовое изменение объектов

Изменение через меню действий в карточке объекта

Чтобы изменить объект:

1. Откройте карточку объекта. Способ открытия карточки зависит от объекта и раздела системы, в котором он расположен.

   ▼ *Пример открытия карточки*

   a. Перейдите в раздел веб-интерфейса системы, где находится объект.

   b. Нажмите на строку с объектом, который вы хотите изменить. Система отобразит карточку этого объекта с подробной информацией о нем.

2. Выберите опцию **Изменить** в выпадающем меню кнопки действий ⋮ в верхней части карточки. Отобразится окно редактирования объекта.

   В некоторых разделах системы опция может иметь название **Редактировать**.

3. Внесите изменения в требуемые поля объекта.
4. Нажмите на кнопку **Сохранить**. Система сохранит измененные настройки объекта.

Изменение полей объекта через кнопку редактирования в карточке

Чтобы изменить поля объекта в его карточке:

1. Откройте карточку объекта. Способ открытия карточки зависит от объекта и раздела системы, в котором он расположен.

   ▼ *Пример открытия карточки*

   a. Перейдите в раздел веб-интерфейса системы, где находится объект.

   b. Нажмите на строку с объектом, который вы хотите изменить. Система отобразит карточку этого объекта с подробной информацией о нем.

2. Нажмите на кнопку ✎ в строке необходимого поля для его редактирования.

   Кнопка может быть доступна не для всех полей в карточке объекта.

3. Внесите изменения в поле объекта.
4. Нажмите на кнопку **Сохранить** (✓). Система сохранит измененные настройки.

Изменение объекта через кнопку в карточке

Чтобы изменить объект:

1. Откройте карточку объекта. Способ открытия карточки зависит от объекта и раздела системы, в котором он расположен.

▼ *Пример открытия карточки*

a. Перейдите в раздел веб-интерфейса системы, где находится объект.

b. Нажмите на строку с объектом, который вы хотите изменить. Система отобразит карточку этого объекта с подробной информацией о нем.

2. Нажмите на кнопку **Изменить** в верхней части карточки. Отобразится окно редактирования объекта.



В некоторых разделах системы кнопка может иметь название **Редактировать** или отображаться в виде иконки .

3. Внесите изменения в требуемые поля объекта.
4. Нажмите на кнопку **Сохранить**. Система сохранит измененные настройки объекта.

Изменение объекта через панель инструментов в полноэкранной карточке

Чтобы изменить объект:

1. Откройте полноэкранную карточку объекта. Способ открытия зависит от объекта и раздела системы, в котором он расположен.

▼ *Пример открытия полноэкранной карточки*

a. Перейдите в раздел веб-интерфейса системы, где находится объект.

b. Нажмите на строку с объектом, который вы хотите изменить. Система отобразит карточку этого объекта с подробной информацией о нем.

c. Нажмите на кнопку **Подробнее** в нижней части карточки. Система отобразит на экране подробную карточку этого объекта.

2. Нажмите на кнопку на панели инструментов. Отобразится окно редактирования объекта.
3. Внесите изменения в требуемые поля объекта.
4. Нажмите на кнопку **Сохранить**. Система сохранит измененные настройки объекта.

Изменение объекта через панель инструментов

Чтобы изменить объект:

1. Перейдите в раздел веб-интерфейса системы, где находится объект.
2. Нажмите на кнопку на панели инструментов. Отобразится окно редактирования объекта.
3. Внесите изменения в требуемые поля объекта.
4. Нажмите на кнопку **Сохранить**. Система сохранит измененные настройки объекта.

Массовое изменение объектов

Чтобы изменить объекты:

1. Перейдите в раздел веб-интерфейса системы, где находятся объекты.
2. Установите флажки напротив тех объектов, которые вы хотите изменить, в левом столбце

таблицы.

3. Нажмите на кнопку на панели инструментов. Отобразится окно изменения выбранных объектов.
4. Внесите изменения в требуемые поля объектов.
5. Нажмите на кнопку **Сохранить**. Система сохранит измененные настройки объектов.

Удаление объектов

В системе можно удалять объекты следующими способами:

- Удаление объекта через меню действий
- Массовое удаление объектов
- Удаление объекта через кнопку в карточке
- Удаление объекта через панель инструментов в полноэкранной карточке
- Удаление объекта через панель инструментов

Удаление объекта через меню действий

Чтобы удалить объект:

1. Откройте карточку объекта. Способ открытия карточки зависит от объекта и раздела системы, в котором он расположен.

   ▼ *Пример открытия карточки*

   a. Перейдите в раздел веб-интерфейса системы, где находится объект.

   b. Нажмите на строку с объектом, который вы хотите удалить. Система отобразит карточку этого объекта с подробной информацией о нем.

2. Выберите опцию **Удалить** в выпадающем меню кнопки действий ⋮ в верхней части карточки.
3. Если система запросит подтверждение, подтвердите удаление.

   Система удалит выбранный объект.

Массовое удаление объектов

Чтобы удалить сразу несколько объектов:

1. Перейдите в раздел веб-интерфейса системы, где находятся объекты.
2. В левом столбце таблицы установите флажки напротив тех объектов, которые вы хотите удалить.
3. Нажмите на кнопку **Удалить выбранные** на панели инструментов.

   В некоторых разделах системы кнопка может иметь название **Удалить** или отображаться в виде иконки .

4. Если система запросит подтверждение, подтвердите удаление.

Система удалит выбранные объекты.

Удаление объекта через кнопку в карточке

Чтобы удалить объект:

1. Откройте карточку объекта. Способ открытия карточки зависит от объекта и раздела системы, в котором он расположен.

   ▼ *Пример открытия карточки*

   a. Перейдите в раздел веб-интерфейса системы, где находится объект.

   b. Нажмите на строку с объектом, который вы хотите удалить. Система отобразит карточку этого объекта с подробной информацией о нем.

2. Нажмите на кнопку **Удалить** в верхней части карточки.

   В некоторых разделах системы кнопка может иметь название **Удалить** или отображаться в виде иконки .

3. Если система запросит подтверждение, подтвердите удаление.

   Система удалит выбранный объект.

Удаление объекта через панель инструментов в полноэкранной карточке

Чтобы удалить объект:

1. Откройте полноэкранную карточку объекта или экран его настроек. Способ открытия зависит от объекта и раздела системы, в котором он расположен.

   ▼ *Пример открытия полноэкранной карточки*

   a. Перейдите в раздел веб-интерфейса системы, где находится объект.

   b. Нажмите на строку с объектом, который вы хотите изменить. Система отобразит карточку этого объекта с подробной информацией о нем.

   c. Нажмите на кнопку **Подробнее** в нижней части карточки. Система отобразит на экране подробную карточку этого объекта.

2. Нажмите на кнопку на панели инструментов.
3. Если система запросит подтверждение, подтвердите удаление.

   Система удалит выбранный объект.

Удаление объекта через панель инструментов

Чтобы удалить объект:

1. Перейдите в раздел веб-интерфейса системы, где находится объект.
2. Нажмите на кнопку на панели инструментов.
3. Если система запросит подтверждение, подтвердите удаление.

Система удалит выбранный объект.

Клавиши и комбинации клавиш

Клавиши и их комбинации для работы в интерфейсе системы.

Общие

Функция	Клавиши и их комбинации
Перейти на предыдущую строку и выделить ее	SHIFT+↑
Перейти на следующую строку и выделить ее	SHIFT+↓
Перейти на первую строку в списке (на текущей странице) и выделить ее	CTRL+SHIFT+↑
Перейти на последнюю строку в списке (на текущей странице) и выделить ее	CTRL+SHIFT+↓
Перейти на следующую страницу	SHIFT+→
Перейти на предыдущую страницу	SHIFT+←
Перейти на последнюю страницу	CTRL+SHIFT+→
Перейти на первую страницу	CTRL+SHIFT+←
Перейти к следующему полю формы	TAB
Перейти к предыдущему полю формы	SHIFT+TAB
Закрыть с отменой (отправки формы)	ESC
Закрыть с подтверждением (отправки формы)	ENTER
Развернуть карточку выбранного элемента	ENTER
Свернуть карточку выбранного элемента	ESC

Поиск по событиям

Функция	Клавиши и их комбинации
Перейти в строку поиска	SHIFT+R
Запустить поиск	SHIFT+ENTER
Остановить текущий поиск и запустить новый (перезапустить)	CTRL+SHIFT+ENTER
Открыть форму сохранения поискового запроса как шаблона	CTRL+S
Открыть форму добавления фильтра	SHIFT+F

Конфигурация конвейера

Функция	Клавиши и их комбинации
Удалить выделенный элемент конвейера или связь	DELETE (в Windows также BACKSPACE)

VRL-песочница

Функция	Клавиши и их комбинации	Область применения
Отмена последнего изменения	CTRL+Z	Панели **Программа**, **Событие** и **Мета**
Повтор последнего отмененного действия	CTRL+Y	Панели **Программа**, **Событие** и **Мета**
Открытие панели со списком встроенных функций	CTRL+SPACE	Панели **Программа**, **Событие** и **Мета**
Открытие панели детального описания ошибок	ALT+F8	Панели **Программа**, **Событие** и **Мета**, где есть ошибки
Запуск выполнения программы	SHIFT+ENTER	Любое место песочницы

Редактор кода

Функция	Клавиши и их комбинации
Сместить табуляцию на шаг вправо	TAB
Сместить табуляцию на шаг влево	SHIFT+TAB
Вызвать подсказки кода Чтобы вставить выбранную подсказку, нажмите TAB или ENTER.	CTRL+SPACE
Искать по коду	CTRL+F
Включить или выключить поиск с учетом регистра	ALT+C
Включить или выключить поиск слова целиком	ALT+W
Включить или выключить поиск с регулярными выражениями	ALT+R
Выделить текущее слово.	CTRL+D Чтобы выделить следующее идентичное значение и добавить курсор после него, удерживая CTRL, нажмите D.
Добавить курсор на следующие строки	CTRL+ALT+↓
Добавить курсор на предыдущие строки	CTRL+ALT+↑
Выделить всю строку	CTRL+L
Выделить следующую строку	SHIFT+↓
Выделить предыдущую строку	SHIFT+↑

Функция	**Клавиши и их комбинации**
Выделить весь код	CTRL+A
Копировать выделенное в буфер обмена	CTRL+C
Вырезать выделенное	CTRL+X
Вставить из буфера обмена	CTRL+V
Переместить текущую строку вниз	ALT+↓
Переместить текущую строку вверх	ALT+↑
Дублировать текущую строку и переместить курсор вниз	SHIFT+ALT+↓
Дублировать текущую строку и оставить курсор на текущей строке	SHIFT+ALT+↑
Отменить последнее действие	CTRL+Z
Повторить отмененное действие	CTRL+Y

Начало работы с системой

Для начала работы необходимо настроить обработку событий ИБ в системе. Настройка включает следующие этапы:

1. Регистрация баз данных событий.
2. Создание моделей событий.
3. Создание хранилищ событий с указанием используемой базы данных и модели события.
4. Настройка элементов экспертизы.
5. Настройка активных списков на основе схем.
6. Создание секретов.
7. Настройка драйверов внешних баз данных.
8. Настройка коллекторов.
9. Настройка интеграций.

Предустановленные примеры сущностей системы

Чтобы ознакомиться с функционалом системы, вы можете воспользоваться предустановленными примерами сущностей. Система содержит примеры следующих сущностей:

- База данных событий.
- Хранилище событий.
- Элементы экспертизы:
  - Правила нормализации.
  - Правила корреляции.
  - Правило агрегации.
  - Глобальная функция.
  - Шаблоны поиска.
  - Схемы шаблонов отчетов.
  - Схемы дашбордов.
  - Таблицы обогащения.
  - Схемы активных списков.
- Активные списки.
- Секреты.
- Коллектор.
- Дашборды.
- Шаблон отчета.

Названия сущностей-примеров имеют префикс **Demo**.



Сущности-примеры можно редактировать и удалять.

Подключение источников

Для получения и обработки событий необходимо подключить нужные источники, откуда эти события будут поступать в систему. Процесс подключения источников состоит из следующих этапов:

1. Создание коллектора.
2. Создание конвейера в коллекторе.
3. Создание точки входа на конвейере для получения событий.
4. Настройка обработки и хранения событий с помощью прочих элементов конвейера.



- Если вы хотите отслеживать состояние подключенных источников, обратитесь к разделу Аудит источников событий.
- Если при подключении источника наблюдаются проблемы (например, источник подключен, но события не приходят), обратитесь к разделу Устранение проблем при эксплуатации системы.

Справочник по конфигурации источников

С инструкциями по настройке и подключению конкретных источников можно ознакомиться в руководстве по настройке источников на справочном портале R-Vision.

Список поддерживаемых системой источников событий можно посмотреть в разделе Поддерживаемые источники событий руководства по настройке источников.

Основные сценарии подключения источников

Для ознакомления с основными этапами подключения источника и настройки конвейера обратитесь к тестовому сценарию Обработка событий на одном конвейере.

Если вы планируете работать с небольшим количеством источников данных (например, до пяти штук), вы можете настроить их обработку на нескольких конвейерах в рамках одного коллектора. Подробности приведены в разделе Обработка событий на нескольких конвейерах.

Если вы планируете работать с большим количеством источников, рекомендуется настраивать отдельный коллектор для каждого из источников. Подробности приведены в разделе Обработка событий на нескольких коллекторах.

Подключение источников для решения конкретных задач

Если вы хотите настроить источник Syslog или Socket и данные приходят с нескольких IP-адресов, обратитесь к разделу Обработка событий с нескольких IP-адресов.

Если вы работаете в распределенной системе и вам необходимо получать события из сателлита, обратитесь к разделу Сбор событий из вынесенного коллектора.

Если вам необходимо получать информацию с удаленных хостов, на которых установлены агенты, обратитесь к разделу Сбор событий с агентов.

Обработка событий на одном конвейере

Данное руководство описывает процесс создания коллектора и конвейера для сбора и обработки событий.

Об обработке событий в коллекторе

Для работы с событиями нужно создать **коллектор** — компонент, который обеспечивает сбор событий из источников, их обработку и дальнейшую отправку на хранение и/или во внешние системы. События поступают в коллектор через установленные на нем конвейеры, посредством которых они могут быть обработаны.

Конвейер — это упорядоченная последовательность взаимосвязанных этапов обработки, представленных такими элементами, как коррелятор, нормализатор, агрегатор, VRL-трансформация и фильтр. Эти элементы конфигурируются на основе правил и условий, сформулированных на VRL — языке описания трансформаций и корреляций.

С помощью маршрутизатора и связей между элементами конвейера можно управлять порядком обработки событий. После обработки события можно отправлять в связанные с конвейером хранилища, а также во внешние системы. Хранилище находится в базе данных в виде таблицы, формат которой определяется моделью событий хранилища.

В данном разделе рассмотрен процесс настройки коллектора, в котором все события собираются и обрабатываются в рамках единственного конвейера.

В полном цикле настройки системы для приема и обработки событий можно выделить следующие этапы:

1. Создание модели и хранилища событий
    a. Создание модели события
    b. Регистрация базы данных событий
    c. Создание хранилища событий
2. Создание коллектора и конвейера
    a. Создание коллектора
    b. Создание конвейера
3. Добавление элементов конвейера
    a. Добавление точки входа
    b. Добавление VRL-трансформации
    c. Добавление маршрутизатора
    d. Добавление фильтра
    e. Настройка агрегации событий
    f. Настройка нормализации событий

g. Настройка корреляции событий
h. Настройка оповещений
i. Добавление конечной точки
j. Объединение элементов конвейера

4. Включение коллектора и установка конфигурации конвейера



Следуйте шагам настройки в указанной последовательности. Добавить элементы конвейера можно в любом порядке. Однако соответствующие элементы экспертизы нужно создавать перед добавлением элементов конвейера.

Примеры конкретных конфигураций приведены после общего описания действий.

Создание модели и хранилища событий

Создание модели события

При создании хранилища нужно обязательно указать используемую в нем **модель события**. Модель определяет формат хранилища событий: имена, типы полей и данных в полях.

Чтобы создать модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания модели события.
3. Введите название модели события.
4. Введите описание модели события.
5. В нижней части окна находится раздел **Схема**, содержащий список полей, которые будут использоваться в этой модели события.

   По умолчанию система отображает набор **служебных** полей. Список можно расширить, добавив вручную **регулярные** поля.

   Для обеспечения оптимальной производительности системы рекомендуется создавать модели с количеством полей не более 1000.

6. Нажмите на кнопку **Создать**. Система создаст модель событий и отобразит соответствующее уведомление. Новая модель появится в списке раздела **Ресурсы** → **Модели событий**.

Регистрация базы данных событий

Базы данных (БД) событий используются для хранения событий в системе. Единицей хранения событий в БД является хранилище событий. На уровне БД хранилище представляет собой таблицу. Столбцы и типы данных хранилища задаются моделью события.

Чтобы зарегистрировать БД событий:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об

имеющихся БД событий.

2. Нажмите на кнопку **Зарегистрировать** (+). Отобразится окно регистрации БД событий.
3. Введите название БД событий.
4. Введите идентификатор сервиса ClickHouse, в котором необходимо зарегистрировать БД событий, в NATS Service Discovery.



Чтобы получить идентификатор сервиса ClickHouse в NATS Service Discovery, обратитесь к администратору системы или воспользуйтесь разделом Получение идентификатора сервиса в NATS Service Discovery.

5. При необходимости введите описание БД событий.
6. Нажмите на кнопку **Зарегистрировать**. Система создаст БД событий и отобразит соответствующее уведомление. Новая база отобразится в списке раздела **Ресурсы** → **Базы данных событий**.

Создание хранилища событий

Хранилище событий используется для сохранения событий, поступивших в систему и обработанных коллектором. Оно представляет собой таблицу в БД событий. Формат хранилища определяется его моделью событий.

Чтобы создать хранилище событий:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания хранилища событий.
3. Выберите из выпадающего списка БД событий, в которую требуется поместить хранилище. Система отобразит в нижней части поля для настройки томов хранения этой БД.



В системных базах данных создание хранилищ событий не поддерживается.

4. Выберите из выпадающего списка модель события, которая будет использована для создаваемого хранилища.
5. Введите название хранилища событий.
6. При необходимости введите описание хранилища событий.
7. Задайте настройки хранения данных:

   a. Выберите том хранения данных:

      - **hot_volume** — том для быстрого доступа к часто используемым данным (горячее хранение).
      - **cold_volume** — том для хранения редко используемых данных (холодное хранение).

   b. Укажите срок хранения данных в днях.



Если поле для срока хранения данных оставить пустым, то срок хранения данных в этом хранилище будет неограниченным.

c. Выберите из выпадающего списка действие над данными в томе по окончании срока хранения:

   - **Переместить** — данные будут перемещены в другой том. Введите в появившейся строке настройки нового тома, в который требуется переместить данные.
   - **Удалить** — данные в томе будут удалены.

   При необходимости можно задать настройки хранения данных на основе настроек по умолчанию в выбранной БД. Для этого нажмите на кнопку **Применить настройки хранения по умолчанию БД**. Если настройки по умолчанию не заданы, система отобразит соответствующее сообщение.

8. Нажмите на кнопку **Создать**. Система создаст хранилище событий и отобразит соответствующее уведомление. Новое хранилище появится в списке раздела **Ресурсы** → **Хранилища событий**.

Создание коллектора и конвейера

Создание коллектора

Перед тем как система начнет принимать события от источника данных, необходимо создать соответствующий коллектор.

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором будет создан коллектор. По умолчанию выбрано подключение к центральному пространству.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.

   - При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов на каждом узле кластера Kubernetes.
   - Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
   - Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. Для работы в режиме распределенной корреляции установите переключатель **Распределенный коррелятор** в активное положение.

   Изменение этой настройки потребует перезапуска коллектора.



Если система работает на базе кластера Kubernetes с одним узлом, включение режима распределенной корреляции недоступно.

8. Укажите количество ядер процессора (**CPU**) и объем оперативной памяти (**RAM**), выделяемый для работы распределенного коррелятора. Это можно сделать непосредственно в поле или с помощью кнопок + и —.



Операции полей `join` и `group` декларативного правила вычисляются на коррелляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. В составе распределенного коррелятора может быть только один коррелятор, но несколько предпроцессоров.

   a. Укажите ресурсы для коррелятора:

      i. Выберите узел для размещения коррелятора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы коррелятора. Количество доступных на узле ресурсов отображается под полями ввода.

   b. Укажите ресурсы для предпроцессора:

      i. Выберите узел для размещения предпроцессора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы предпроцессора. Количество доступных на узле ресурсов отображается под полями ввода.

9. Нажмите на кнопку **Добавить предпроцессор**, если необходимо добавить дополнительный предпроцессор.

10. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:

    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

11. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы → Коллекторы**.

Создание конвейера

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного ранее. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление элементов конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы настроить конвейер, требуется открыть его конфигурацию в режиме черновика. Если при создании конвейера флажок **Открыть конфигурацию конвейера** не был установлен, то выполните одно из действий:

- Нажмите на кнопку ⌄ в строке конвейера. Откроется карточка конвейера. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
- Нажмите на кнопку действий ⋮ в строке конвейера и выберите опцию **Конфигурация конвейера**.

В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Далее рассмотрим сценарий настройки конфигурации конвейера для обработки исходных событий в формате JSON.

▼ *Пример исходного события*

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"IvanIvanov\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\"
,\"request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"statu
s\":\"404\",\"bytes\":22427,\"referer\":\"https://example.com\"}"
}
```

Добавление точки входа

Точка входа поставляет в систему поток сырых событий. Выбор формата точки входа зависит от

формата данных, получаемых из источника события. Точка входа имеет один выход и не имеет входов.



Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

Чтобы добавить точку входа на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
2. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
3. Введите название точки входа.
4. Выберите тип точки входа и задайте ее параметры. Набор параметров точки входа зависит от выбранного типа.
5. Если требуется отслеживать состояние источника данных, выберите политику аудита источников. В противном случае установите переключатель **Аудит** в неактивное положение.

   По умолчанию выбрана системная **Политика аудита источников по точке входа**.

   Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.
6. Если заданные настройки точки входа необходимо сохранить в качестве шаблона для дальнейшего использования, установите флажок **Сохранить как шаблон**.
7. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Пример:

Создадим источник, который будет поставлять логи HTTP-сервера в формате JSON.

Для этого выберем тип точки входа Demo Logs и формат генерируемых событий **json**. Укажем в названии точки входа "JSON".

Добавление VRL-трансформации

VRL-трансформация служит для обработки и модификации событий с применением языка VRL. Этот элемент конвейера принимает на вход события, проводит их преобразование и на выходе предоставляет уже модифицированные события.

Чтобы добавить VRL-трансформацию на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **VRL-трансформация**. Отобразится окно добавления VRL-трансформации.
2. При необходимости выберите шаблон для автоматического заполнения полей VRL-трансформации. По умолчанию выбран вариант **Без шаблона**.
3. Введите название VRL-трансформации.

4. Введите правило, по которому будет преобразовываться событие, на языке VRL.



В редакторе кода доступны горячие клавиши.



В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

5. При необходимости из выпадающего списка **Обработка ошибок** выберите шину коллектора, в которую будет отправлено событие в случае возникновения ошибок обработки. По умолчанию выбран вариант **События с ошибкой**.
6. Установите флажок **Сохранить как шаблон**, если настройку VRL-трансформации необходимо сохранить в качестве шаблона для дальнейшего использования.
7. Нажмите на кнопку **Добавить**. Новая VRL-трансформация отобразится на диаграмме конфигурации.

Пример:

Добавим условие на языке VRL, согласно которому исходное событие будет обогащаться дополнительным полем `responseStatus`. В нем будет устанавливаться значение "Bad request", если в поле `status` приходящего на VRL-трансформация события содержится значение 401 или 404.

Для этого создадим VRL-трансформацию с названием "AssignResponseStatus" и пропишем в свойствах следующее условие:

▼ *Пример условия обогащения событий, язык VRL*

```
if .status == "404" || .status == "401" {
.responseStatus = "Bad request"
}
```

▼ *Интерпретация*

`if .status == "404" || .status == "401"`:

- Условное выражение проверяет, равно ли значение поля `status` события строке "404" или "401".

`.responseStatus = "Bad request"`:

- Если условие выше выполняется, то поле `responseStatus` текущего события устанавливается в значение "Bad request".

Добавление маршрутизатора

Маршрутизатор направляет события на разные этапы обработки в зависимости от их содержимого или других условий. Маршрутизатор имеет один вход и несколько выходов. На входе каждое событие проверяется по условиям и появляется на тех выходах, где условие выполняется.

Чтобы добавить маршрутизатор на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Маршрутизатор**. Отобразится окно добавления маршрутизатора.
2. Введите название маршрутизатора.

   Для маршрутизатора недопустимо использование названий "_unmatched" и "_default".

3. Настройте список маршрутов маршрутизатора.
   a. Нажмите кнопку **Добавить маршрут**.
   b. Введите название маршрута.
   c. Задайте правило обработки событий на языке VRL.
   d. Нажмите на кнопку **Добавить**. Маршрут будет добавлен в маршрутизатор.
4. При необходимости установите флажок **Добавить маршрут по умолчанию**. В этом случае к маршрутизатору будет добавлен маршрут **По умолчанию**, на выход которого будут попадать те события, которые не удовлетворяют условиям остальных маршрутов.

   При неустановленном флажке события, не удовлетворяющие условиям ни одного из маршрутов, будут удаляться.

5. Нажмите на кнопку **Добавить**. Новый маршрутизатор отобразится на диаграмме конфигурации.

Пример:

Добавим проверку статуса события, после которой:

- События со статусами 400, 401, 403 и 404 отправляются на дальнейшую обработку.
- События со статусами 500, 502 и 504 отправляются на хранение.

Для этого создадим маршрутизатор с названием "StatusCheckRouter", добавим два маршрута, а также маршрут по умолчанию.

Первому маршруту дадим название "ClientErrorRoute" и следующее условие:

Пример условия фильтрации событий для первого маршрута, схема RObject

```
.event.status == "400" || .event.status == "401" || .event.status == "403" ||
.event.status == "404"
```

Второму маршруту дадим название "ServerErrorRoute" и следующее условие:

Пример условия фильтрации событий для второго маршрута, схема RObject

```
.event.status == "500" || .event.status == "502" || .event.status == "504"
```

Добавление фильтра

Фильтр предназначен для сокращения объема обрабатываемых событий путем их отсеивания на

основе заданных условий. Условия обычно формулируются через имена и значения полей, и для их определения используются математические и логические операторы. Например, условие `externalId == "4648" && categoryOutcome == "Success"` позволяет пропустить через фильтр только те события, которые удовлетворяют этим критериям. У фильтра есть один вход и один выход. На вход фильтра подается событие, и если оно соответствует заданным условиям, то проходит на выход. В противном случае событие отсеивается и не участвует в дальнейшей обработке.

Чтобы добавить фильтр на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Фильтр**. Отобразится окно добавления фильтра.
2. При необходимости выберите шаблон для автоматического заполнения полей фильтра. По умолчанию выбран вариант **Без шаблона**.
3. Введите название фильтра.
4. Задайте условие фильтрации входящих событий на языке VRL.

   Условия обычно формулируются через имена полей и их значения, и для их определения используются математические и логические операторы.

   В редакторе кода доступны горячие клавиши.

   В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

5. Установите флажок **Сохранить как шаблон**, если настройку фильтра необходимо сохранить в качестве шаблона для дальнейшего использования.
6. Нажмите на кнопку **Добавить**. Новый фильтр отобразится на диаграмме конфигурации.

Пример:

Добавим фильтр, который пропускает только события с хоста 198.51.100.191.

Для этого создадим фильтр с названием "FilterByHost", и в поле **Условие** пропишем следующее выражение:

Пример условия фильтрации событий, схема RObject

```
.event.host == "198.51.100.191"
```

Настройка агрегации событий

Чтобы события, проходящие через конвейер обработки событий, агрегировались, необходимо добавить агрегатор.

Чтобы добавить агрегатор, необходимо создать и установить на нем правило агрегации, согласно которому будут объединяться поступающие события.

Создание правила агрегации

Правило агрегации в системе является структурированным объектом данных в формате RObject (.ro), описывающим процесс объединения и анализа групп событий безопасности. Основная функция правила агрегации — сбор и объединение данных из различных событий в рамках заданной логики. Каждое правило агрегации включает в себя критерии для группировки событий, условия начала и завершения агрегации, а также стратегии агрегации для каждого поля. Агрегация инициируется в соответствии с условиями, определенными в правилах, и преобразует входящий поток событий в агрегированные данные.

Чтобы добавить правило агрегации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило агрегации**. Система отобразит окно создания правила агрегации.

▼ *Пример правила агрегации, схема RObject*

```
# Уникальный идентификатор правила.
id: example/network_activity_rule

# Название правила.
name: NetworkActivityAggregationRule

# Тип элемента экспертизы.
type: aggregation_rule

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Агрегация событий сетевой активности для анализа трафика

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги, связанные с правилом.
tags:
  - network
  - traffic_analysis

# Отбрасывает события, которые не соответствуют критериям.
```

```
filter: !vrl |
  # Фильтрует события, не относящиеся к категории "Network" или имеющие низкий
уровень серьезности.
  .category == "Network" &&
  to_int!(.severity) > 3

# Группировка событий по источнику и назначению.
group_by:
  - src_ip
  - dest_ip

# Условие открытия агрегационного окна.
start_when: !vrl |
  # Начинаем агрегацию при обнаружении события начала соединения.
  .event_type == "connection_start"

# Максимальное количество событий.
max_events: 5

# Время ожидания последнего события (миллисекунды).
expire_after_ms: 60000

# Интервал проверки окон агрегации (миллисекунды).
flush_period_ms: 2000

# Стратегии агрегации значений для каждого указанного поля.
merge_strategies:
  bytes_sent: sum
  bytes_received: sum
  severity: max

# Тесты для проверки работы правила.
tests:
 - name: Network Activity Test
   events:
   - {"category":"Network", "severity":5, "event_type":"connection_start",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":1500,
"bytes_received":2000}
   - {"category":"Network", "severity":4, "event_type":"data_transfer",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":3000,
"bytes_received":4000}
   - {"category":"Network", "severity":5, "event_type":"connection_end",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":500,
"bytes_received":600}
   assertion: !vrl |
     # Проверяем, что агрегированные значения соответствуют ожиданиям.
     assert_eq!(.[0], {
       "category": "Network",
       "event_type": "connection_start",
       "src_ip": "192.0.2.10",
       "dest_ip": "192.0.2.20",
```

```
        "bytes_sent": 5000,
        "bytes_received": 6600,
        "severity": 5
    })
```

▼ *Интерпретация*

Правило агрегации "NetworkActivityAggregationRule" служит для агрегации сетевых событий, соответствующих передаче данных в рамках одного сетевого соединения.

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `aggregation_rule` указывает на то, что это правило агрегации.
- `version`: версия правила.
- `description`: описывает цель правила — "Агрегация событий сетевой активности для анализа трафика".
- `author`: создатель правила.
- `data_source`: указывает источник правила:
  - `platform`: платформа — Windows.
  - `source`: источник событий — Sysmon.
  - `events`: события `EventID_1` и `EventID_2`.
- `tags`: теги `network` и `traffic_analysis` классифицируют правило по его функциональности и области применения.

Фильтрация событий (filter):

- `filter`: фильтрует события по категории "Network" и уровню серьезности более 3. Это сужает область применения правила только к событиям, которые удовлетворяют условиям фильтрации.

Группировка событий (group_by):

- `group_by`: группирует события по полям `src_ip` и `dest_ip`. Это позволяет анализировать взаимодействия между конкретными источниками и назначениями в сети.

Условие начала агрегации (starts_when):

- `starts_when`: определяет начало агрегации при появлении событий типа "connection_start", что является признаком начала нового сетевого соединения.

Параметры агрегационного окна (max_events, expire_after_ms, flush_period_ms):

- `max_events`: ограничивает количество событий в агрегации до 5. После достижения этого количества, агрегационное окно закрывается и генерируется агрегированное событие.
- `expire_after_ms`: устанавливает максимальный период ожидания последнего события в

агрегационном окне в 60000 миллисекунд (1 минута).

- `flush_period_ms`: задает частоту проверки и закрытия агрегационных окон в 2000 миллисекунд (2 секунды).

Стратегии агрегации (merge_strategies):

- `merge_strategies`: определяет, как будут объединяться поля событий. Для "bytes_sent" и "bytes_received" применяется стратегия суммирования (`sum`).

Тестирование правила (tests):

- `name`: `Network Activity Test` — название теста.
- `events`: описывает ряд тестовых событий, имитирующих реальный сетевой трафик, для проверки эффективности правила.
- `assertion`: проверяет, что правило корректно агрегирует события и вычисляет суммы переданных и полученных байтов. Это подтверждает, что правило работает как предполагалось и агрегирует данные согласно заданным критериям.

▼ *Обработка событий правилом агрегации*

Проанализируем работу правила агрегации "NetworkActivityAggregationRule", фокусируясь на процессе обработки событий.

Представим, что на агрегатор поступает поток сетевых событий, которые соответствуют критериям, заданным в фильтре `filter` правила. Эти события содержат данные о типе сетевого соединения, IP-адресах источника и назначения, а также объеме переданных и полученных данных.

```
{
  "category": "Network",
  "severity": 5,
  "event_type": "connection_start",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.20",
  "bytes_sent": 1500,
  "bytes_received": 2000
}
```

По мере поступления этих событий, они фильтруются правилом "NetworkActivityAggregationRule". Правило агрегирует события, группируя их по IP-адресам источника и назначения (`group_by`). Таким образом, события, исходящие от одной пары IP-адресов, объединяются в одно агрегированное событие.

Агрегация инициируется, когда обнаруживается событие типа `connection_start` (`starts_when`). Правило продолжает агрегировать события до тех пор, пока не будет достигнуто максимальное количество событий (`max_events`) или не истечет время ожидания (`expire_after_ms`).

Агрегированное событие включает в себя сумму значений `bytes_sent` и `bytes_received` для всех событий в группе (`merge_strategies`).

```
{
  "event_type": "aggregated_network_activity",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.20",
  "total_bytes_sent": 5000,
  "total_bytes_received": 6600
}
```

Данное агрегированное событие отражает общую активность между двумя конкретными IP-адресами, позволяя идентифицировать аномалии или необычную активность в сети.

3. Заполните поля правила агрегации, чтобы определить логику его работы, и нажмите на кнопку **Опубликовать версию**. Система создаст правило и отобразит соответствующее уведомление. Новое правило агрегации отобразится в таблице раздела **Экспертиза**.
4. Чтобы созданное правило стало доступным на конвейерах событий, включите его:
   a. Нажмите на строку с созданным правилом агрегации. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
   b. Убедитесь, что в карточке открыта вкладка **Информация**.
   c. Установите переключатель состояния в верхней части карточки в активное положение и подтвердите включение правила. Система обновит информацию о статусе правила агрегации.

Добавление агрегатора

Агрегатор представляет собой компонент конвейера, который использует правила агрегации для объединения и анализа групп событий безопасности.

Порядок применения правил агрегации определяется последовательностью их добавления в агрегатор. Это позволяет управлять процессом агрегации и определять, какие события будут объединены вместе.

Агрегатор на входе принимает поток событий, производит их агрегацию и выдает агрегированные события на выходе.

Чтобы добавить агрегатор на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Агрегатор**. Отобразится окно добавления агрегатора.
3. Введите название агрегатора.
4. Добавьте в агрегатор правила агрегации.

   Чтобы добавить правило агрегации:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило агрегации.

b. Установите флажок слева от правила агрегации, которое необходимо добавить в агрегатор.

5. Нажмите на кнопку **Добавить**. Новый агрегатор отобразится на диаграмме конфигурации.

Пример:

Добавим агрегатор с правилом `NetworkActivityAggregationRule` на конвейер. В названии агрегатора укажем "AggregateEvents".

Настройка нормализации событий

Чтобы события, проходящие через конвейер обработки событий, нормализовывались, необходимо добавить нормализатор.

Чтобы добавить нормализатор, необходимо создать и установить на нем правило нормализации, согласно которому нормализатор будет обрабатывать и нормализовывать поступающие данные.

Создание правила нормализации

Правила нормализации в системе обрабатывают и преобразовывают поступающие события согласно условиям и коду нормализации, составленным на языке VRL и условию, заданному в поле `filter`.

Чтобы добавить правило нормализации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило нормализации**. Система отобразит окно создания правила нормализации.

▼ *Пример правила нормализации, схема RObject.*

```
# Уникальный идентификатор правила.
id: example/basic_normalization_rule

# Название правила.
name: BasicNormalizationRule

# Имя автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Базовое правило нормализации

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
```

```
    - logs/default.log
    - logs/somelog.log

# Теги, связанные с правилом.
tags:
  - normalization
  - security

# Тип элемента экспертизы.
type: normalization_rule

# Блок правил на языке VRL с условиями фильтрации событий.
filter: !vrl |
  .event.host == "198.51.100.191"

# Блок с правилами на языке VRL для нормализации полей события.
normalizer: !vrl |
    e = parse_json(.event) ?? {}
    . |= {
      "userId": e."user-identifier",
      "timestamp": parse_timestamp(e.datetime, "%d/%b/%Y:%H:%M:%S") ?? now(),
      "host": e.host,
      "method": e.method,
      "path": e.request,
      "protocol": e.protocol
    }

    if e.status == "404" || e.status == "401" {
      .responseStatus = "Bad request"
    }

    if e.protocol == "HTTP/2.0" {
        .protocol = "HTTP/2"
    }
```

▼ *Интерпретация*

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `normalization_rule` указывает на то, что это правило нормализации.
- `version`: версия правила.
- `description`: описывает цель правила — "Базовое правило нормализации".
- `author`: создатель правила.
- `data_source`: указывает источник правила:
  - `platform`: платформа — Windows.
  - `source`: источник событий — Sysmon.

    - `events`: журналы событий `logs/default.log` и `logs/somelog.log`.
- `tags`: теги `normalization` и `security` классифицируют правило по его функциональности и области применения.

Фильтрация событий (`filter`):

- `.event.host == "198.51.100.191"`:

  Сравнение поля `host` со значением `198.51.100.191` в сыром событии. Если значение поля `host` в сыром событии совпадает с `198.51.100.191`, то событие будет нормализовано согласно правилам в поле `normalizer`. Если значение поля `host` не совпадает с указанным IP-адресом, правила нормализации к событию применены не будут.

Нормализация событий (`normalizer`):

- `e = parse_json(.event) ?? {}`:

  Попытка преобразовать содержимое поля `.event` из формата JSON в структуру данных VRL.

  Если преобразование не удалось или результатом является `null`, переменной `e` будет присвоен пустой объект `{}`, что достигается с помощью оператора `??`.

- `. |= { ⋯ }`:

  Оператор обновления или добавления новых полей к текущему объекту на основе значений из объекта `e`.

  Структура, которая присваивается текущему объекту:

  - `userId`: присваивает значение поля `user-identifier` из объекта `e` к полю `userId` текущего объекта.
  - `timestamp`: преобразует строковое значение поля `datetime` из объекта `e` в метку времени с использованием указанного формата. Если преобразование не удастся или значение поля `datetime` равно `null`, будет использовано текущее время, возвращаемое функцией `now()`.
  - `host`: присваивает значение поля `host` из объекта `e` к полю `host` текущего объекта.
  - `method`: присваивает значение поля `method` из объекта `e` к полю `method` текущего объекта.
  - `path`: присваивает значение поля `request` из объекта `e` к полю `path` текущего объекта.
  - `protocol`: присваивает значение поля `protocol` из объекта `e` к полю `protocol` текущего объекта.

- `if e.status == "404" || e.status == "401"`:

  Условное выражение проверяет, равно ли значение поля `status` из объекта `e` строке `404` или `401`.

- `.responseStatus = "Bad request"`:

Если условие выше выполняется, то полю `responseStatus` текущего объекта присваивается значение `Bad request`.

- `if e.protocol == "HTTP/2.0"`:

  Условное выражение проверяет, равно ли значение поля `protocol` из объекта `e` строке `HTTP/2.0`.

- `.protocol = "HTTP/2"`:

  Если условие выше выполняется, то поле `protocol` текущего объекта заменяется на значение `HTTP/2`.

▼ *Обработка событий правилом нормализации*

Рассмотрим работу правила на примере разбора базового (сырого) события.

Допустим, на нормализатор поступают события следующего вида:

Исходное событие, JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"IvanIvanov\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GE
T\",\"request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\
"status\":\"404\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}"
}
```

После применения к событию правила "BasicNormalizationRule", оно принимает следующий вид:

Нормализованное событие, JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"IvanIvanov\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GE
T\",\"request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\
"status\":\"404\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}",
  "host": "198.51.100.191",
  "method": "GET",
  "path": "/observability/metrics/production",
  "protocol": "HTTP/2",
  "referer": "https://example.url.com",
  "responseBytes": 22427,
  "responseStatus": "Bad request",
  "serverIP": "198.51.100.191",
  "timestamp": "2023-07-30T23:55:49Z",
  "userAgent": null,
  "userId": "IvanIvanov"
}
```

Событие удовлетворяет условию из поля `filter`, следовательно, оно подлежит нормализации и обрабатывается согласно правилам из поля `normalizer`. В структуре нормализованного

события в поле `event` записывается исходное событие. Прочие поля добавляются к объекту и определяются в соответствии с правилами нормализации, хранящимся в поле `normalizer`.

3. Заполните поля правила нормализации, чтобы определить логику его работы, и нажмите на кнопку **Опубликовать версию**. Система создаст правило и отобразит соответствующее уведомление. Новое правило нормализации отобразится в таблице раздела **Экспертиза**.
4. Чтобы созданное правило стало доступным на конвейерах событий, включите его:
   a. Нажмите на строку с созданным правилом нормализации. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
   b. Убедитесь, что в карточке открыта вкладка **Информация**.
   c. Установите переключатель состояния в верхней части карточки в активное положение и подтвердите включение правила. Система обновит информацию о статусе правила нормализации.

Добавление нормализатора

Нормализатор представляет собой комплекс правил нормализации. Каждое такое правило состоит из двух основных компонентов: кода нормализации, выполненного на языке VRL, и фильтра. Если в правиле есть фильтр, то нормализация инициируется, только когда входящее событие соответствует условиям этого фильтра.

Порядок применения правил нормализации определяется последовательностью их добавления в нормализатор. Таким образом, события проходят через набор правил в установленном порядке, что позволяет осуществлять сложные сценарии нормализации данных.

Нормализатор на входе принимает события, преобразует и обогащает их, и выдает преобразованные события на выходе.

Чтобы добавить нормализатор на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Нормализатор**. Отобразится окно добавления нормализатора.
3. Введите название нормализатора.
4. При необходимости из выпадающего списка **Обработка ошибок** выберите шину коллектора, в которую будет отправлено событие в случае возникновения ошибок обработки. По умолчанию выбран вариант **События с ошибкой**.
5. Если требуется отслеживать состояние источника данных, выберите политику аудита источников. В противном случае установите переключатель **Аудит** в неактивное положение.

   По умолчанию выбрана системная **Политика аудита источников по маске**.



Если политика аудита источников включена в нескольких нормализаторах на конвейере, то в метриках по источнику количество событий будет умножено на количество нормализаторов с аудитом.

Рекомендуется включать аудит только в одном нормализаторе на конвейере.



При выборе политики с методом расчета **По среднему EPS** системе необходимо собрать минимальное количество данных по источнику для корректного расчета среднего значения EPS. После включения политики система ожидает 24 часа, накапливая данные. В этот период нарушения политики по отклонениям EPS не фиксируются.

6. Добавьте в нормализатор правила нормализации.

   Чтобы добавить правило нормализации:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило нормализации.

   b. Установите флажок слева от правила нормализации, которое необходимо добавить в нормализатор.

7. Нажмите на кнопку **Добавить**. Новый нормализатор отобразится на диаграмме конфигурации.

Пример:

Добавим нормализатор с правилом `BasicNormalizationRule` на конвейер. В названии нормализатора укажем "Normalization".

Настройка корреляции событий

Чтобы события, поступающие на конвейер, коррелировались, необходимо добавить коррелятор.

Чтобы добавить коррелятор, необходимо создать и установить в нем правило корреляции. Правила корреляции определяют, как коррелятор будет обрабатывать входящие данные и формировать на их основе корреляционные события.

Создание правила корреляции

Правила корреляции в системе позволяют обнаруживать и анализировать взаимосвязанные события, соответствующие определенным условиям. Эти правила также могут инициировать действия после обнаружения, например, создание корреляционных событий и оповещений.

Чтобы добавить правило корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило корреляции**. Система отобразит окно выбора способа создания правила.
3. Выберите способ создания правила через код и нажмите на кнопку **Создать**. Система отобразит окно задания RObject-конфигурации правила корреляции.

   ▼ *Пример правила корреляции, схема RObject*

   ```
   # Уникальный идентификатор правила.
   id: example/ddos_detection_rule_basic
   ```

```
# Название правила.
name: DDoSDetectionRule

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
```

```
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в
секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip)
  .additional_info = "Высокая частота запросов или подозрительные паттерны в пути
запроса /api/highload"

# Управляет созданием оповещений по результатам корреляции.
trigger_alert: true

# Тесты для проверки работы правила.
tests:
  - name: DDoS Attack Detected
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type":
"HTTP_REQUEST", "source_ip": "192.0.2.1", "window": "120s", "request_path":
"start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type":
"HTTP_REQUEST", "source_ip": "192.0.2.1", "window": "130s", "request_path":
"start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type":
"HTTP_REQUEST", "source_ip": "192.0.2.1", "window": "140s", "request_path":
"start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type":
"HTTP_REQUEST", "source_ip": "192.0.2.111", "window": "10s", "request_path":
"/api/highload" }
    assertion: !vrl |
      assert_eq!(.[0]message, "Обнаружена потенциальная DDoS атака с источника:
192.0.2.1")
```

▼ *Интерпретация*

Правило корреляции "DDoSDetectionRule" предназначено для выявления DDoS-атак на основе анализа потока поступающих событий.

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `correlation_rule` указывает на то, что это правило корреляции.
- `version`: версия правила.
- `description`: описывает цель правила — "Обнаружение атаки DDoS на основе трафика".
- `author`: создатель правила.
- `data_source`: указывает источник правила:
  - `platform`: платформа — Windows.
  - `source`: источник событий — Sysmon.
  - `events`: события `EventID_13` и `EventID_1`.
- `tags`: теги `ddos` и `security` классифицируют правило по его функциональности и области применения.

Логика фильтрации (filter):

- Условия фильтрации `.source_ip != null` и `.request_path != null` выбирают события, у которых есть IP-адрес источника и путь запроса.

Логика фильтрации (aliases):

- Псевдоним `HighRateTraffic`: фильтр для событий с высокой частотой запросов, идентифицируемых по `event_type == "HTTP_REQUEST"` и `parse_duration!(.window) >= 100`. Отбирает события с частотой запросов, превышающей или равной 100 запросов в определенный временной интервал.
- Псевдоним `SuspiciousPatterns`: фильтр для событий с подозрительными путями запроса, определяемый как `event_type == "HTTP_REQUEST"` и `ends_with(to_string!(.request_path), "/api/highload")`. Отбирает запросы, пути которых оканчиваются на "/api/highload".

Соединение потоков событий (select и join):

- Выбор и соединение (`select` и `join`) потоков событий `HighRateTraffic` и `SuspiciousPatterns` осуществляется на основе совпадения `source_ip`. Это соединение позволяет коррелировать высокочастотные запросы с подозрительными паттернами, исходящими от одного источника.
- В поле `wait_for` определяется, что система должна ожидать в течение 30 секунд перед тем, как принять решение о соединении событий из различных псевдонимов. Это значит, что система будет накапливать события в течение указанного времени с момента получения первого события и затем анализировать их на предмет соответствия заданным критериям соединения.

- В операции соединения с `SuspiciousPatterns` используется поле `absent: true`, что указывает на то, что отсутствие события из `SuspiciousPatterns` в течение 30 секунд будет добавлено в цепочку корреляционных событий.

Группировка событий (group):

- `group`: группировка событий применяется к псевдониму `HighRateTraffic` по полю `source_ip`, что позволяет агрегировать и анализировать события, исходящие от одного источника IP. Количество событий для формирования группы в поле `count` установлено в 3, что означает, что группа будет сформирована при накоплении 3 событий от одного источника IP.
- `throttle_time_sec`: задает интервал времени в секундах между генерацией корреляционных событий в рамках каждой группы событий. Например, 60 секунд.

Срок хранения промежуточных данных корреляции (ttl):

- `ttl` на уровне правила задает общее максимальное время жизни промежуточных корреляционных данных. Например, 60 секунд. Это время определяет, как долго промежуточные результаты для всей цепочки событий будут храниться в памяти до их автоматического удаления, если корреляция не завершается.

Пост-корреляционная логика (on_correlate):

- `on_correlate`: обогащает корреляционное событие полями `.message` и `.additional_info`.
- `trigger_alert`: управляет созданием оповещений по результатам корреляции. В данном случае, с `trigger_alert: true`, система будет генерировать оповещения при успешной корреляции событий.

По накоплении 3 событий правило генерирует корреляционное событие со следующими атрибутами:

```
{
  "type": "correlation_rule",
  "severity": "high",
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена потенциальная DDoS атака с источника: 192.168.1.1",
  "additional_info": "Высокая частота запросов и подозрительные паттерны в пути
запроса /api/highload"
}
```

Тестирование правила (tests):

- Набор тестовых сценариев "DDoS Attack Detected" включает набор событий, часть из которых имитируют характеристики DDoS-атаки, а остальные являются контрольными примерами обычного трафика. Цель тестирования — подтвердить, что правило корректно идентифицирует только события, связанные с DDoS-атаками и игнорирует неподходящие события. Тесты включают события с HTTP-запросами, различающимися по `source_ip` и `request_path`, что позволяет проверить способность правила различать атакующий трафик

от легитимного.

- События, моделирующие DDoS-атаку, имеют одинаковый `source_ip` и конечный путь запроса "/api/highload", что соответствует фильтрам `HighRateTraffic` и `SuspiciousPatterns`. Это сценарий, в котором ожидается активация правила и генерация корреляционного события.
- Контрольные события с разными `source_ip` и неподозрительными путями запроса ("/home", "/contact", "/about") предназначены для подтверждения, что правило не реагирует на обычные запросы, не связанные с DDoS-активностью.

`assertion` в тестовом блоке предназначен для проверки того, что только события, соответствующие характеристикам DDoS-атаки, приводят к формированию корреляционного события с ожидаемым сообщением об обнаружении атаки.

▼ *Обработка событий правилом корреляции*

Проанализируем логику работы правила "DDoSDetectionRule", фокусируясь на процессе обработки событий.

Предположим, что за короткий промежуток времени на коррелятор, на котором установлено правило "DDoSDetectionRule", поступает поток JSON-событий от различных источников IP, включая множество запросов с IP-адреса 192.168.1.1. Эти события соответствуют критериям, заданным в фильтрах `HighRateTraffic` и `SuspiciousPatterns`.

Исходное событие, JSON

```
{
  "event_type": "HTTP_REQUEST",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:34:56Z",
  "request_path": "/api/highload",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/93.0.4472.124 Safari/537.36"
}
```

По мере поступления этих событий, они фильтруются правилом "DDoSDetectionRule". Фильтр `HighRateTraffic` отбирает события, соответствующие запросам с высокой частотой (более 100 запросов за заданный временной интервал), а `SuspiciousPatterns` фокусируется на запросах с определенным путем запроса `/api/highload`.

Когда события от одного источника IP, например, 192.168.1.1, соответствуют обоим фильтрам и проходят через механизм соединения `join`, правило "DDoSDetectionRule" приступает к группировке. Правило использует поле `group` для агрегирования событий от одного источника IP. Если количество событий от одного источника соответствует заданному порогу в 3 события, правило генерирует корреляционное событие.

Корреляционное событие, генерируемое правилом "DDoSDetectionRule"

```
{
  "type": "correlation_rule",
  "severity": "high",
```

```
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена потенциальная DDoS атака с источника: 192.168.1.1",
  "additional_info": "Высокая частота запросов и подозрительные паттерны в пути
запроса /api/highload"
}
```

Данное корреляционное событие указывает на то, что система обнаружила аномальную активность, соответствующую критериям DDoS-атаки, исходящей с одного IP-адреса. Правило "DDoSDetectionRule" коррелирует события, основываясь на двух ключевых факторах: частоте запросов и специфических путях запросов, что является индикатором потенциальной атаки.

4. Заполните поля правила корреляции, чтобы определить логику его работы, и нажмите на кнопку **Опубликовать версию**. Система создаст правило и отобразит соответствующее уведомление. Новое правило корреляции отобразится в таблице раздела **Экспертиза**.
5. Чтобы созданное правило стало доступным на конвейерах событий, включите его:
   a. Нажмите на строку с созданным правилом корреляции. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
   b. Убедитесь, что в карточке открыта вкладка **Информация**.
   c. Установите переключатель состояния в верхней части карточки в активное положение и подтвердите включение правила. Система обновит информацию о статусе правила корреляции.



В примере рассматривается способ создания правила корреляции через код. Для правила корреляции доступен также способ построения через специальный конструктор.

Добавление коррелятора

Коррелятор позволяет связывать различные события и создавать комплексные правила на их основе. На входе применяет правила корреляции к потоку событий. На выходе выдает корреляционные события.

Чтобы добавить коррелятор на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Коррелятор**. Отобразится окно добавления коррелятора.
3. Введите название коррелятора.
4. Задайте интервал очистки событий в секундах. По истечении указанного периода в императивных правилах корреляции будет происходить очистка событий с истекшим временем жизни. Время жизни задается в поле `expire_after_sec` в структуре правила.

   По умолчанию интервал очистки равен 300 секундам.

5. Если нужно проверять события по черному и белому спискам, установите флажок **Проверка исключений**.
6. Добавьте в коррелятор правила корреляции.



Если коллектор, на котором размещен коррелятор, настроен для работы в режиме распределенной корреляции, он не поддерживает работу императивных правил корреляции.

Чтобы добавить правило корреляции:

a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило корреляции.
b. Установите флажок слева от правила корреляции, которое необходимо добавить в коррелятор.

7. Нажмите на кнопку **Добавить**. Новый коррелятор отобразится на диаграмме конфигурации.

Пример:

Добавим коррелятор с правилом DDoSDetectionRule на конвейер. Укажем в названии "DDoSDetectionRule".

Настройка оповещений

Чтобы корреляционные события, получаемые после срабатывания правил корреляции, генерировали оповещения, необходимо добавить на конвейер сервис оповещений.

Чтобы добавить сервис оповещений, необходимо создать и установить на нем правило сегментации, согласно которому корреляционные события будут распределяться по оповещениям.

Создание правила сегментации

Правила сегментации в системе позволяют управлять потоком корреляционных событий, обеспечивая их агрегацию, классификацию и преобразование в оповещения согласно заданным условиям.

Чтобы добавить правило сегментации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Правило сегментации**. Система отобразит окно создания правила сегментации.

▼ *Пример правила сегментации, схема RObject*

```
# Уникальный идентификатор правила сегментации.
id: segmentation/ddos_alerts

# Название правила сегментации.
name: DDoSAlertsSegmentation.
```

```
# Тип элемента экспертизы.
type: segmentation_rule

# Версия правила сегментации.
version: 1.0.0

# Описание назначения правила сегментации.
description: Сегментация и классификация оповещений для атак типа DDoS.

# Имя и контактные данные автора правила сегментации.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - logs/default.log
    - logs/somelog.log

# Теги, связанные с правилом сегментации.
tags:
  - ddos
  - alert
  - segmentation

# Группирует корреляционные события по указанным полям. Если не указано, все
события помещаются в единую группу.
group_by:
  - source_ip

# Задает максимальное количество корреляционных событий в группе оповещения.
# При достижении максимального значения создается новое оповещение.
threshold: 1

# Задает максимальное время жизни корреляционных событий в группе.
# При достижении максимального значения создается новое оповещение.
timeout: 60

# Код VRL для формирования имени оповещения (например из частей имен событий).
# Выполняется только при создании оповещения (при обновлении существующего
оповещения не выполняется).
# Если поле не определено, оповещению будет автоматически присвоено название
правила корреляции, которое первым привело к его срабатыванию.
alert_name: !vrl |
  "DDoS_ALERT_" + to_string!(.source_ip)

# Тесты для проверки работы правила сегментации.
tests:
  - name: High traffic from single source IP
    events:
```

```
      - { type: "correlation_rule", severity: "high", event_type: "DDoS
Detection", source_ip: "198.51.100.1", datetime: "2023-09-28T12:45:00Z" }
      - { type: "correlation_rule", severity: "high", event_type: "DDoS
Detection", source_ip: "198.51.100.2", datetime: "2023-09-28T12:47:00Z" }
      - { type: "correlation_rule", severity: "high", event_type: "DDoS
Detection", source_ip: "198.51.100.1", datetime: "2023-09-28T12:49:00Z" }
    assertion: !vrl |
      assert_eq!(length(.), 3)
      assert_eq!(.[0].group_by.source_ip, "198.51.100.1")
      assert_eq!(.[0].name, "DDoS_ALERT_198.51.100.1")
      assert_eq!(.[1].group_by.source_ip, "198.51.100.2")
      assert_eq!(.[1].name, "DDoS_ALERT_198.51.100.2")
```

▼ *Интерпретация*

Правило сегментации "DDoSAlertsSegmentation" создано для обработки и классификации корреляционных событий, генерируемых правилом "DDoSDetectionRule", целью которого является выявление DDoS-атак на основе анализа потока событий.

Метаданные правила сегментации:

- `id`: уникальный идентификатор правила сегментации в системе.
- `name`: название правила сегментации, отражающее его назначение, — сегментация оповещений, связанных с DDoS-атаками.
- `type`: тип элемента экспертизы указывает, что это правило относится к категории правил сегментации.
- `version`: версия правила сегментации, позволяющая отслеживать его итерации и обновления.
- `description`: краткое описание назначения правила сегментации — классификация и сегментация оповещений для атак DDoS.
- `author`: автор правила сегментации и его контактные данные.
- `tags`: метки, связанные с правилом, такие как ddos, alert, segmentation, помогающие в классификации и поиске правила.

Параметры сегментации:

- `group_by`: указывает на поля событий, которые используются для группировки корреляционных событий. В данном случае, это `source_ip`, что позволяет сегментировать события по уникальным источникам IP.
- `threshold`: количество событий от одного источника, при достижении которого будет сгенерировано новое оповещение.
- `timeout`: временной интервал в секундах, в течение которого агрегируются события для сегментации.
- `alert_name`: код VRL для формирования имени оповещения на основе атрибутов события. В данном случае, IP-адреса источника. Если поле не определено, оповещению будет автоматически присвоено название правила корреляции, которое привело к его срабатыванию.

Тестирование правила сегментации "DDoSAlertsSegmentation":

Поле `tests` в правиле сегментации "DDoSAlertsSegmentation" предназначено для верификации и валидации логики правила путем имитации реальных сценариев обработки корреляционных событий. Это позволяет убедиться, что логика сегментации работает корректно и соответствует заранее определенным требованиям к обработке и классификации событий.

В данном случае, тестовый блок содержит следующие ключевые компоненты:

- `name`: имя теста, в данном случае "High traffic from single source IP", что указывает на цель теста — проверить способность правила сегментации агрегировать и классифицировать события, характерные для DDoS-атак.
- `events`: перечисление событий, которые имитируют корреляционные события, сгенерированные правилом обнаружения DDoS. Эти события содержат различные `source_ip` и метаданные, характерные для событий обнаружения DDoS, такие, как тип события "DDoS Detection", уровень серьезности "high", и временная метка.
- `assertion`: содержит скрипт на языке VRL, предназначенный для дополнительной валидации атрибутов каждого сгенерированного оповещения. В частности, проверяется, что имя оповещения соответствует ожидаемому формату, основанному на IP-адресе источника события.

Таким образом, поле `tests` обеспечивает комплексную проверку правила сегментации "DDoSAlertsSegmentation", демонстрируя его способность идентифицировать и классифицировать события, ассоциированные с DDoS-атаками, и соответственно генерировать оповещения для дальнейшего анализа и реагирования.

3. Заполните поля правила сегментации, чтобы определить логику его работы, и нажмите на кнопку **Опубликовать версию**. Система создаст правило и отобразит соответствующее уведомление. Новое правило сегментации отобразится в таблице раздела **Экспертиза**.
4. Чтобы созданное правило стало доступным на конвейерах событий, включите его:
   a. Нажмите на строку с созданным правилом сегментации. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
   b. Убедитесь, что в карточке открыта вкладка **Информация**.
   c. Установите переключатель состояния правила в верхней части карточки в активное положение и подтвердите включение правила. Система обновит информацию о статусе правила сегментации.

Добавление сервиса оповещений

Сервис оповещений позволяет настраивать связи между правилами корреляции и правилами сегментации для распределения корреляционных событий по оповещениям.

Чтобы добавить сервис оповещений на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Сервис оповещений**. Отобразится окно добавления сервиса оповещений.

3. Введите название сервиса оповещений.
4. Выберите из выпадающего списка хранилище с событиями, на основе которых будут формироваться оповещения.
5. При необходимости выберите интеграции для отправки оповещений во внешние системы.
6. При необходимости выберите правило сегментации по умолчанию, которое будет применяться ко всем несвязанным правилам корреляции.

   Если правило по умолчанию не выбрано, то все корреляционные события, сгенерированные одним и тем же правилом корреляции и не удовлетворяющие условиям правил сегментации, будут объединяться в одно и то же оповещение.

7. Настройте связи с правилами сегментации.
   a. Нажмите на кнопку **Добавить связь**. Откроется окно добавления связи.
   b. Выберите требуемое правило сегментации из выпадающего списка. В окне отобразится дерево всех правил корреляции в системе, которые еще не связаны с правилами сегментации в текущем сервисе оповещений.
   c. Выберите требуемые правила корреляции, установив флажки слева от их названий.
   d. Нажмите на кнопку **Добавить**. Добавленная связь с правилом сегментации отобразится в окне настройки сервиса оповещений.
8. Нажмите на кнопку **Добавить**. Новый сервис оповещений отобразится на диаграмме конфигурации.

Пример:

Добавим на конвейер сервис оповещений с правилом сегментации DDoSAlertsSegmentation, связанным с правилом корреляции DDoSDetectionRule. Укажем в названии "DDoSAlerts".

Добавление конечной точки

Конечная точка определяет, куда направляются обработанные события, например, в хранилище или на внешний интерфейс. Конечная точка, в зависимости от настроек, пересылает нормализованное событие в хранилище событий, внешнюю систему или сервис оповещений. Конечная точка имеет один вход и не имеет выходов.



Для отладки конвейера вы можете использовать конечную точку Blackhole, которая не сохраняет и не отправляет события.

Чтобы добавить конечную точку на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная точка**. Отобразится окно добавления конечной точки.
3. При необходимости выберите шаблон для автоматического заполнения полей конечной точки. По умолчанию выбран вариант **Без шаблона**.
4. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):

- Название конечной точки.
- Тип конечной точки.
- Формат конечной точки.

5. Установите флажок **Сохранить как шаблон**, если настройку конечной точки необходимо сохранить в качестве шаблона для дальнейшего использования.
6. Нажмите **Добавить**. Новая конечная точка отобразится на диаграмме конфигурации.

Пример:

Создадим пять конечных точек со следующими названиями: "ServerError", "UnknownError", "HostUnavailable", "DDosDetections" и "198.51.100.191 Status".

Объединение элементов конвейера

Каждый элемент конвейера отвечает за конкретный этап обработки событий. Элемент имеет хотя бы один вход или выход, при этом вход может быть только один, а выходов может быть несколько. Входы и выходы элементов связаны между собой направляющими линиями, которые определяют порядок и направление обработки событий.



Чтобы приблизить или отдалить блоки на диаграмме конфигурации, используйте колесо мыши.

Чтобы связать элементы:

1. Наведите курсор на метку, расположенную на его боковой стороне. Курсор примет вид крестика.
2. Перетащите крестик к метке на другом элементе, чтобы установить связь. Порядок обработки событий будет определен логикой этих связей.

Пример настройки конвейера:

1. Соедините точку входа **JSON** с маршрутизатором **StatusCheckRouter**.
2. Свяжите точку входа **JSON** с агрегатором **AggregateEvents**.
3. Объедините агрегатор **AggregateEvents** с коррелятором **DDoSDetectionRule**.
4. Направьте поток событий из **DDoSDetectionRule** к конечной точке **DDosDetections** и к сервису оповещений **DDoSAlerts**.
5. Направьте поток из маршрута **ClientErrorRoute** маршрутизатора к фильтру **FilterByHost**.
6. Направьте поток из маршрута **ServerErrorRoute** маршрутизатора к конечной точке **ServerError**.
7. Направьте поток из маршрута **По умолчанию** маршрутизатора к конечной точке **UnknownError**.
8. Направьте поток данных из **FilterByHost** к конечной точке **198.51.100.191 Status** и к VRL-трансформации **AssignResponseStatus**.
9. Направьте поток из **AssignResponseStatus** к нормализатору **Normalization**.
10. Соедините нормализатор **Normalization** с конечной точкой **HostUnavailable**.



Если нужно удалить связь, выберите ее и нажмите клавишу DELETE (на Windows

можно также использовать BACKSPACE).



На этот конвейер события поступают через точку входа **JSON** и разделяются на две ветки: к маршрутизатору **StatusCheckRoute** и агрегатору **AggregateEvents**.

Агрегированные события попадают на коррелятор **DDoSDetectionRule**. Если коррелятор генерирует событие, оно сохраняется в конечную точку **DDosDetections**, а сервис оповещений **DDoSAlerts** генерирует соответствующее оповещение. Оповещение можно посмотреть в разделе **Оповещения**.

В маршрутизаторе **StatusCheckRoute** данные фильтруются по условиям маршрутов:

- **ClientErrorRoute**: `.event.status == "400" || .event.status == "401" || .event.status == "403" || .event.status == "404"`

  Соответствующие условию события двигаются к фильтру **FilterByHost**.

- **ServerErrorRoute**: `.event.status == "500" || .event.status == "502" || .event.status == "504"`

  Соответствующие условию события отправляются в конечную точку **ServerError**.

- **По умолчанию**: через этот маршрут проходят события, не удовлетворяющие условиям остальных маршрутов. Соответствующие события отправляются в конечную точку **UnknownError**.

В фильтре **FilterByHost** данные фильтруются по условию `.event.host == "198.51.100.191"`. Соответствующие события двигаются далее к конечной точке **198.51.100.191 Status**, где они записываются в хранилище, и к VRL-трансформации **AssignResponseStatus**, где им присваивается новое поле `responseStatus`.

События из **AssignResponseStatus** идут к нормализатору **Normalization**. На этапе нормализации поля события трансформируются. Если поле `status` имеет значение "Bad request", полю `severity` присваивается значение "high". Затем эти события отправляются в конечную точку **HostUnavailable**.

Включение коллектора и установка конфигурации конвейера



Для включения конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события начали поступать в систему и обрабатываться:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.

2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. Установите переключатель состояния коллектора в верхней части карточки в активное положение. Отобразится окно подтверждения включения коллектора.
5. Нажмите на кнопку **Включить**. Система включит коллектор и отобразит соответствующее уведомление.
6. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
7. Нажмите на кнопку ⌄ в строке конвейера. Отобразится карточка конвейера.
8. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации, над которой расположены кнопки установки или отключения конфигурации конвейера.
9. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
10. Нажмите на кнопку **Установить конфигурацию** над диаграммой конфигурации. Отобразится окно подтверждения установки конфигурации.
11. Нажмите на кнопку **Установить**. Система установит конфигурацию конвейера и отобразит соответствующее уведомление.

Теперь события начнут поступать в систему и обрабатываться согласно настройкам конвейера. Все события, которые проходят через конвейер или генерируются на нем, будут сохраняться согласно настройкам конечных точек. Поиск по этим событиям можно осуществлять с помощью языка запросов RQL в разделе **Поиск**.

При обнаружении последовательности событий, соответствующих заданному правилу корреляции, система будет генерировать оповещения.

Обработка событий на нескольких конвейерах

Данное руководство описывает процесс настройки обработки событий на нескольких конвейерах и создания шины для связи их друг с другом при поступлении событий из нескольких источников.

Об обработке событий на нескольких конвейерах

Обработка событий на основе нескольких конвейеров рекомендуется, когда данные собираются с нескольких источников. В этом случае настраивается конфигурация из нескольких взаимосвязанных конвейеров внутри одного коллектора, что упрощает отладку.

Для каждого источника данных создается конвейер, на котором настраивается сбор и нормализация событий. Итоговая корреляция всех нормализованных событий выполняется на отдельном конвейере. Конвейеры связаны друг с другом посредством шины.

Настройка конфигурации для обработки событий на основе нескольких конвейеров включает в себя следующие этапы:

1. Создание коллектора
2. Настройка шины
3. Добавление источников
    a. Создание конвейера
    b. Добавление точки входа
    c. Настройка обработки данных
        i. Добавление маршрутизатора
        ii. Нормализация данных
        iii. Настройка агрегации событий
    d. Настройка соединения с шиной
4. Настройка корреляции событий
    a. Создание конвейера для корреляции
    b. Настройка отправки данных из шины
    c. Добавление коррелятора
    d. Настройка конечных точек и оповещений
5. Включение коллектора и установка конфигурации конвейеров

Создание коллектора

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором будет создан коллектор. По умолчанию выбрано подключение к центральному пространству.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.



- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов на каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.

7. Для работы в режиме распределенной корреляции установите переключатель **Распределенный коррелятор** в активное положение.



Изменение этой настройки потребует перезапуска коллектора.



Если система работает на базе кластера Kubernetes с одним узлом, включение режима распределенной корреляции недоступно.

8. Укажите количество ядер процессора (**CPU**) и объем оперативной памяти (**RAM**), выделяемый для работы распределенного коррелятора. Это можно сделать непосредственно в поле или с помощью кнопок + и —.



Операции полей `join` и `group` декларативного правила вычисляются на коррелляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. В составе распределенного коррелятора может быть только один коррелятор, но несколько предпроцессоров.

   a. Укажите ресурсы для коррелятора:

      i. Выберите узел для размещения коррелятора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы коррелятора. Количество доступных на узле ресурсов отображается под полями ввода.

   b. Укажите ресурсы для предпроцессора:

      i. Выберите узел для размещения предпроцессора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы предпроцессора. Количество доступных на узле ресурсов отображается под полями ввода.

9. Нажмите на кнопку **Добавить предпроцессор**, если необходимо добавить дополнительный предпроцессор.

10. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:

    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

11. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее

уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Настройка шины

Шина позволяет направлять потоки событий между конвейерами. В коллекторе может быть одна или несколько шин. Шина поддерживает два типа соединения: на получение и на отправку событий. Два конвейера событий могут сообщаться друг с другом посредством шины.

Чтобы создать шину:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, созданного на предыдущем этапе. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления шины.
5. Введите название шины.
6. При необходимости введите описание шины.
7. Нажмите на кнопку **Добавить**. Шина будет создана и отобразится в списке шин на вкладке **Шины** в карточке коллектора.

Добавление источников

Настройте конвейеры для каждого источника данных, повторив следующие шаги.

Создание конвейера

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного ранее. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление точки входа

Точка входа поставляет в систему поток сырых событий. Выбор формата точки входа зависит от формата данных, получаемых из источника события.



Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

Чтобы добавить точку входа на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
2. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
3. Введите название точки входа.
4. Выберите тип точки входа и задайте ее параметры. Набор параметров точки входа зависит от выбранного типа.
5. Если требуется отслеживать состояние источника данных, выберите политику аудита источников. В противном случае установите переключатель **Аудит** в неактивное положение.

   По умолчанию выбрана системная **Политика аудита источников по точке входа**.

   Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.
6. Если заданные настройки точки входа необходимо сохранить в качестве шаблона для дальнейшего использования, установите флажок **Сохранить как шаблон**.
7. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Пример:

Добавим соединение с источником, который поставляет данные в формате системного журнала. Для этого выберем тип точки входа Syslog.

Настройка обработки данных

Чтобы привести события к модели, удобной для обогащения, корреляции и хранения, их нужно обработать с помощью нормализатора.

Данные из различных событий можно объединять согласно заданной логике. Для этого используется агрегатор.

Если события необходимо направлять на разные этапы обработки в зависимости от их содержимого, настройте условия фильтрации с помощью маршрутизатора.

Пример:

Рассмотрим сценарий обработки событий в зависимости от их формата.

Добавление маршрутизатора

Чтобы настроить маршрутизацию событий:

1. Добавьте на конвейер маршрутизатор и задайте в нем маршруты, фильтрующие события.
2. Соедините точку входа со входом маршрутизатора.

Пример:

Рассмотрим пример условия, которое выполняется, если в поле "appname" текста базового события записано значение "some-syslog".

Условие на VRL

```
.raw.appname == "some-syslog"
```

Нормализация данных

Чтобы настроить нормализацию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило нормализации.



Если поток событий направляется по разным маршрутам, создайте несколько соответствующих правил нормализации.

2. Добавьте на конвейер нормализаторы, используя ранее созданные правила нормализации. Необходимо добавить на конвейер по одному нормализатору на каждый маршрут.
3. Соедините выходы маршрутизатора с входами нужных нормализаторов.

Пример:

Рассмотрим фрагмент правила нормализации.

Код VRL

```
...
.dvendor = "Linux"
.dproduct = "Some Product"
.dvchost = .raw.hostname
.msg = .raw.message
...
```

Настройка агрегации событий

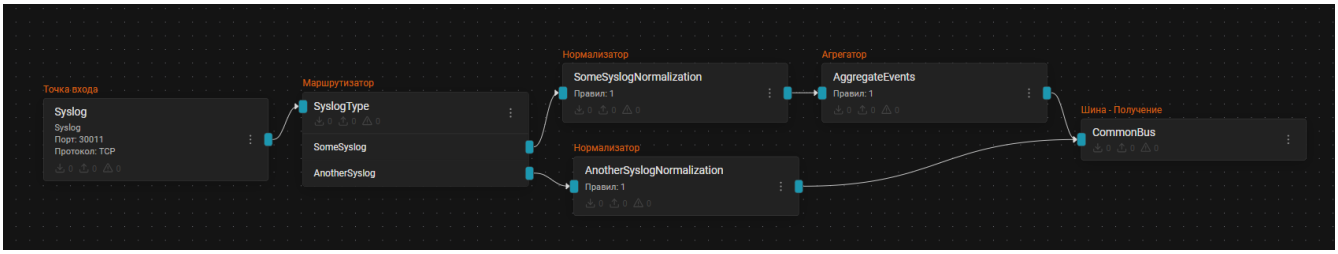
Чтобы настроить агрегацию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило агрегации.
2. Добавьте на конвейер агрегатор, используя ранее созданное правило агрегации.
3. Соедините выход соответствующего нормализатора со входом агрегатора.

Настройка соединения с шиной

Чтобы настроить соединение с шиной:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Шина**.
2. Выберите из выпадающего списка ранее созданную шину. На выбор доступны только шины текущего коллектора.
3. Введите название соединения с шиной. Выберите из выпадающего списка тип соединения **Получение**.
4. Нажмите на кнопку **Добавить**. Соединение с шиной отобразится на диаграмме конфигурации.
5. Соедините выходы элементов, в которых происходила предварительная обработка данных, со входом шины.



Настройка корреляции событий

Создание конвейера для корреляции

Добавьте конвейер, в котором будет происходить корреляция и отправка событий.

Настройка отправки данных из шины

Чтобы настроить отправку данных из шины:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Шина**.
2. Выберите из выпадающего списка ранее созданную шину. На выбор доступны только шины текущего коллектора.
3. Введите название соединения с шиной. Выберите из выпадающего списка тип соединения **Отправка**.
4. Нажмите на кнопку **Добавить**. Соединение с шиной отобразится на диаграмме конфигурации.

Добавление коррелятора

Чтобы настроить корреляцию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило корреляции.
2. Добавьте на конвейер коррелятор, используя ранее созданное правило корреляции.
3. Соедините выход шины со входом коррелятора.



При необходимости добавьте на конвейер фильтры и прочие элементы для дополнительной обработки и анализа событий.

Настройка конечных точек и оповещений

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему.

Чтобы настроить отправку событий:

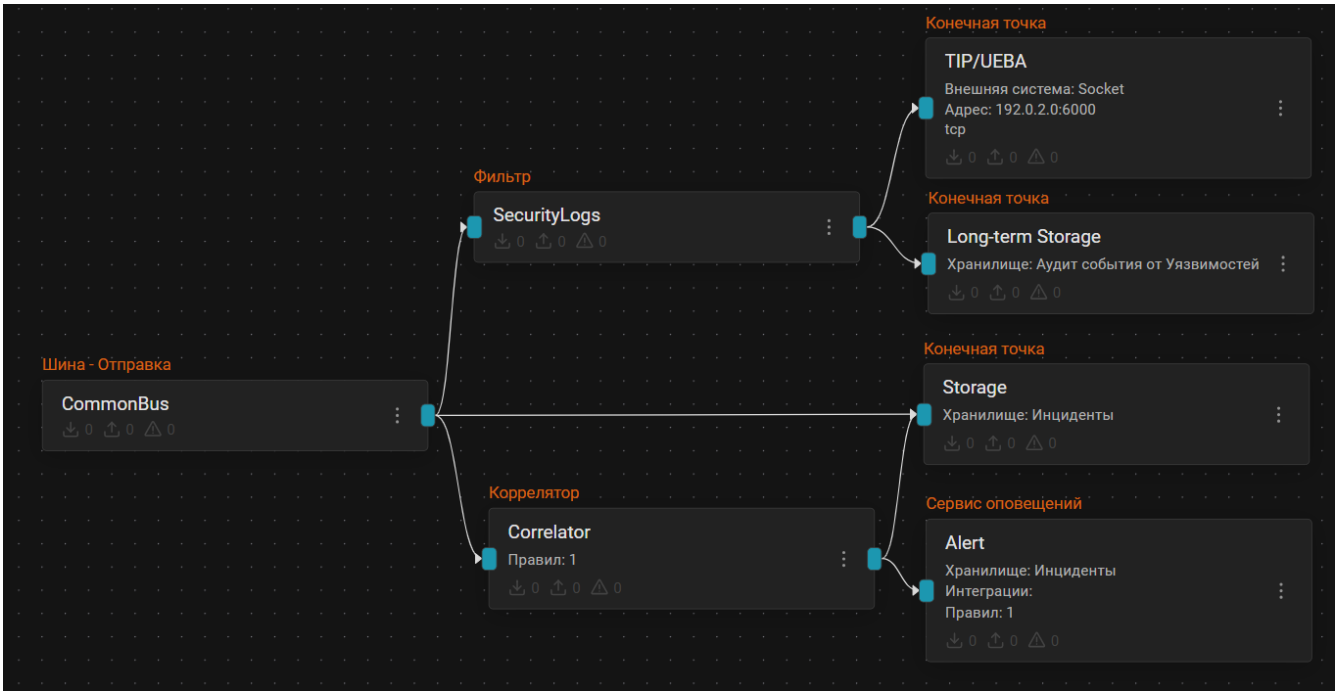
1. Добавьте на конвейер одну или несколько конечных точек, в которых будут сохраняться события.
2. Соедините выходы из шины, коррелятора, созданных на предыдущих шагах, и/или из дополнительных элементов, со входами соответствующих конечных точек.

Сервис оповещений позволяет настраивать связи между правилами корреляции и правилами сегментации для распределения корреляционных событий по оповещениям.

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Сервис оповещений**. Отобразится окно добавления сервиса оповещений.
2. Введите название сервиса оповещений.
3. Выберите из выпадающего списка хранилище с событиями, на основе которых будут формироваться оповещения.
4. При необходимости выберите интеграции для отправки оповещений во внешние системы.
5. При необходимости выберите правило сегментации по умолчанию, которое будет применяться ко всем несвязанным правилам корреляции.

   Если правило по умолчанию не выбрано, то все корреляционные события, сгенерированные одним и тем же правилом корреляции и не удовлетворяющие условиям правил сегментации, будут объединяться в одно и то же оповещение.

6. Настройте связи с правилами сегментации.
   a. Нажмите на кнопку **Добавить связь**. Откроется окно добавления связи.
   b. Выберите требуемое правило сегментации из выпадающего списка. В окне отобразится дерево всех правил корреляции в системе, которые еще не связаны с правилами сегментации в текущем сервисе оповещений.
   c. Выберите требуемые правила корреляции, установив флажки слева от их названий.
   d. Нажмите на кнопку **Добавить**. Добавленная связь с правилом сегментации отобразится в окне настройки сервиса оповещений.
7. Нажмите на кнопку **Добавить**. Новый сервис оповещений отобразится на диаграмме конфигурации.
8. Соедините выходы из коррелятора, созданного на предыдущем шаге, или из дополнительных элементов, со входом сервиса оповещений.



Включение коллектора и установка конфигурации конвейеров

Чтобы события начали поступать в коллектор и обрабатываться в соответствии с заданными правилами и условиями:

1. Перейдите в карточку коллектора и включите его.
2. Установите текущую конфигурацию на каждом конвейере.

Обработка событий на нескольких коллекторах

Данное руководство описывает процесс настройки обработки событий на нескольких коллекторах при поступлении событий из множественных источников.

Об обработке событий на нескольких коллекторах

Обработка событий на основе нескольких коллекторов рекомендуется в тех случаях, когда данные собираются с пяти и более источников. В этом случае настраивается конфигурация из нескольких взаимосвязанных конвейеров, которые распределены по нескольким коллекторам.

Для каждого источника данных создается отдельный коллектор, на котором настраивается сбор и нормализация событий. Итоговая корреляция данных выполняется на дополнительном коллекторе. Связи между коллекторами задаются через конечные точки и точку входа типа Vector.

Такая конфигурация удобна для настройки и масштабирования, а также обладает высокой отказоустойчивостью за счет независимости от изменений в одном из источников.

Настройка конфигурации для обработки событий на основе нескольких коллекторов включает в себя следующие этапы:

1. Добавление источников
    a. Создание коллектора и конвейера

   b. Добавление точки входа
   c. Настройка обработки данных
   d. Добавление конечной точки Vector
2. Настройка корреляции событий
   a. Создание коллектора и конвейера для корреляции
   b. Добавление точки входа Vector
   c. Добавление коррелятора
   d. Настройка конечных точек и оповещений
3. Включение коллектора и установка конфигурации конвейеров

Добавление источников

Настройте коллекторы для каждого источника данных, повторив следующие шаги.



Если требуется получать данные на нескольких конвейерах в рамках одного коллектора, соедините их с помощью шины, как описано в разделе Настройка отправки данных из шины. В отдельном конвейере соедините шину, настроенную на получение данных, с конечной точкой Vector.

Создание коллектора и конвейера

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором будет создан коллектор. По умолчанию выбрано подключение к центральному пространству.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.



- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов на каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.

7. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.
8. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного ранее. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление точки входа

Точка входа поставляет в систему поток сырых событий. Выбор формата точки входа зависит от формата данных, получаемых из источника события.



Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

Чтобы добавить точку входа на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
2. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По

умолчанию выбран вариант **Без шаблона**.

3. Введите название точки входа.
4. Выберите тип точки входа и задайте ее параметры. Набор параметров точки входа зависит от выбранного типа.
5. Если требуется отслеживать состояние источника данных, выберите политику аудита источников. В противном случае установите переключатель **Аудит** в неактивное положение.

   По умолчанию выбрана системная **Политика аудита источников по точке входа**.

   Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.

6. Если заданные настройки точки входа необходимо сохранить в качестве шаблона для дальнейшего использования, установите флажок **Сохранить как шаблон**.
7. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Пример:

Добавим соединение с источником, который поставляет данные в формате системного журнала. Для этого выберем тип точки входа Syslog.

Настройка обработки данных

Чтобы привести события к модели, удобной для обогащения, корреляции и хранения, их нужно обработать с помощью нормализатора.

Чтобы настроить нормализацию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило нормализации.

   Если поток событий направляется по разным маршрутам, создайте несколько соответствующих правил нормализации.

2. Добавьте на конвейер нормализаторы, используя ранее созданные правила нормализации. Необходимо добавить на конвейер по одному нормализатору на каждый маршрут.
3. Соедините выходы маршрутизатора с входами нужных нормализаторов.

Пример:

Рассмотрим фрагмент правила нормализации.

Код VRL

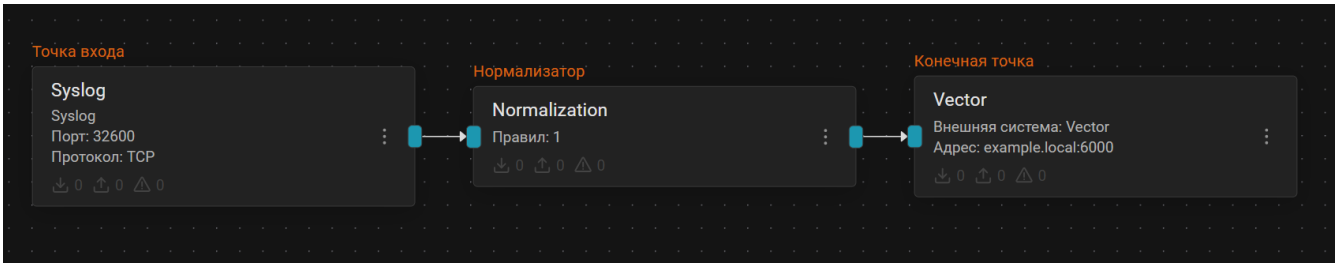
```
...
.dvendor = "Linux"
.dproduct = "Some Product"
.dvchost = .raw.hostname
.msg = .raw.message
...
```

При необходимости добавьте на конвейер маршрутизаторы, агрегаторы и другие элементы для дополнительной обработки событий.

Добавление конечной точки Vector

Чтобы настроить отправку данных из конвейера:

1. Добавьте на конвейер конечную точку, выбрав тип **Внешняя система** и тип внешней системы Vector.
2. Соедините выходы из нормализатора, созданного на предыдущем шаге, и/или из дополнительных элементов, со входом конечной точки.



Настройка корреляции событий

Создание коллектора и конвейера для корреляции

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором будет создан коллектор. По умолчанию выбрано подключение к центральному пространству.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.

   - При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов на каждом узле кластера Kubernetes.
   - Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
   - Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. Для работы в режиме распределенной корреляции установите переключатель **Распределенный**

коррелятор в активное положение.



Изменение этой настройки потребует перезапуска коллектора.



Если система работает на базе кластера Kubernetes с одним узлом, включение режима распределенной корреляции недоступно.

8. Укажите количество ядер процессора (**CPU**) и объем оперативной памяти (**RAM**), выделяемый для работы распределенного коррелятора. Это можно сделать непосредственно в поле или с помощью кнопок + и —.



Операции полей `join` и `group` декларативного правила вычисляются на коррелляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. В составе распределенного коррелятора может быть только один коррелятор, но несколько предпроцессоров.

   a. Укажите ресурсы для коррелятора:

      i. Выберите узел для размещения коррелятора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы коррелятора. Количество доступных на узле ресурсов отображается под полями ввода.

   b. Укажите ресурсы для предпроцессора:

      i. Выберите узел для размещения предпроцессора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы предпроцессора. Количество доступных на узле ресурсов отображается под полями ввода.

9. Нажмите на кнопку **Добавить предпроцессор**, если необходимо добавить дополнительный предпроцессор.

10. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:

    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

11. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы → Коллекторы**.

12. Добавьте конвейер, в котором будет происходить корреляция и отправка событий.

Добавление точки входа Vector

Добавьте на конвейер точку входа, выбрав тип Vector.

Добавление коррелятора

Чтобы настроить корреляцию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило корреляции.
2. Добавьте на конвейер коррелятор, используя ранее созданное правило корреляции.
3. Соедините выход из точки входа Vector со входом коррелятора.

Настройка конечных точек и оповещений

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему.

Чтобы настроить отправку событий:

1. Добавьте на конвейер одну или несколько конечных точек, в которых будут сохраняться события.
2. Соедините выход коррелятора, созданного на предыдущем шаге, со входами конечных точек.

Сервис оповещений позволяет настраивать связи между правилами корреляции и правилами сегментации для распределения корреляционных событий по оповещениям.

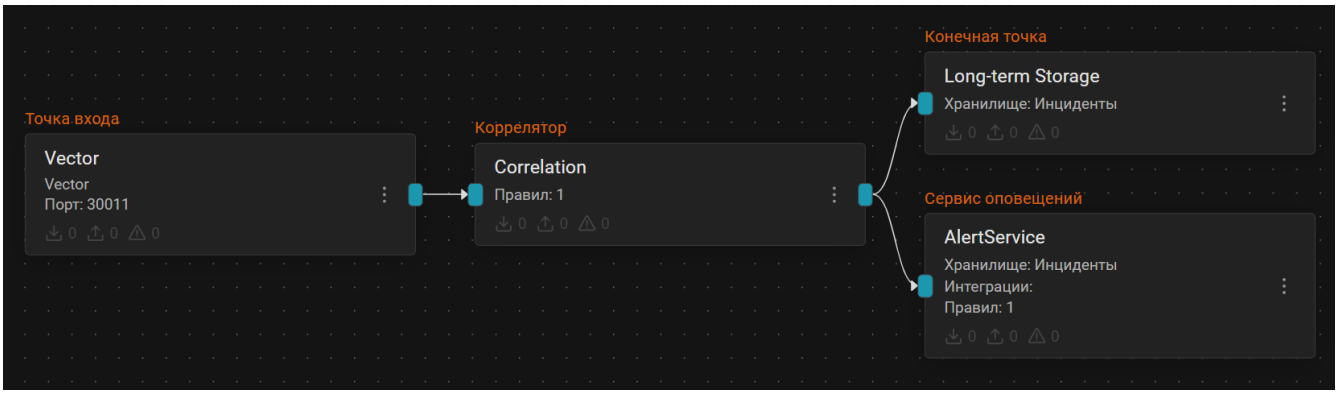
1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Сервис оповещений**. Отобразится окно добавления сервиса оповещений.
2. Введите название сервиса оповещений.
3. Выберите из выпадающего списка хранилище с событиями, на основе которых будут формироваться оповещения.
4. При необходимости выберите интеграции для отправки оповещений во внешние системы.
5. При необходимости выберите правило сегментации по умолчанию, которое будет применяться ко всем несвязанным правилам корреляции.

   Если правило по умолчанию не выбрано, то все корреляционные события, сгенерированные одним и тем же правилом корреляции и не удовлетворяющие условиям правил сегментации, будут объединяться в одно и то же оповещение.

6. Настройте связи с правилами сегментации.
   a. Нажмите на кнопку **Добавить связь**. Откроется окно добавления связи.
   b. Выберите требуемое правило сегментации из выпадающего списка. В окне отобразится дерево всех правил корреляции в системе, которые еще не связаны с правилами сегментации в текущем сервисе оповещений.
   c. Выберите требуемые правила корреляции, установив флажки слева от их названий.

d. Нажмите на кнопку **Добавить**. Добавленная связь с правилом сегментации отобразится в окне настройки сервиса оповещений.

7. Нажмите на кнопку **Добавить**. Новый сервис оповещений отобразится на диаграмме конфигурации.
8. Соедините выход коррелятора, созданного на предыдущем шаге, со входом сервиса оповещений.



Включение коллектора и установка конфигурации конвейеров

Чтобы события начали поступать в коллектор и обрабатываться в соответствии с заданными правилами, повторите для каждого коллектора следующие действия:

1. Перейдите в карточку коллектора и включите его.
2. Установите текущую конфигурацию на конвейере.

Обработка событий с нескольких IP-адресов

Данное руководство описывает процесс настройки конвейера для обработки событий, поступающих на одну точку входа с нескольких IP-адресов.

Об обработке событий с нескольких IP-адресов

Система позволяет принимать в одну точку входа события с нескольких IP-адресов и настраивать для каждого IP-адреса отдельные правила обработки событий в рамках одного конвейера, без необходимости создания разных конвейеров с отдельными точками входа на каждый IP-адрес.

Для настройки обработки событий с разных IP-адресов понадобятся следующие элементы конвейера:

1. **Точка входа**: подключается к источнику данных и выдает поток исходных событий источника.

   Возможность определения IP-адреса доступна только для событий в точках входа Socket и Syslog.

2. **Маршрутизатор**: проверяет каждое событие по условиям маршрутов и направляет его на выходы тех маршрутов, где условие фильтрации выполняется.

   Каждый маршрут должен быть настроен на проверку нужного IP-адреса. Таким образом, события, пришедшие с одного IP-адреса, будут перенаправляться на один и тот же маршрут.

3. **Нормализаторы**: приводят структуру полученных событий к единому формату для упрощения последующей работы.

   На каждый маршрут должен быть настроен отдельный нормализатор.

4. **Конечные точки**: отправляют обработанные события в хранилища данных и внешние системы.
5. **Дополнительные элементы**: зависят от целей и задач пользователя по обработке и анализу событий.

Таким образом, настройка конвейера для обработки событий с нескольких IP-адресов включает следующие этапы:

1. Добавление конвейера
2. Настройка конфигурации конвейера
   a. Точка входа
   b. Маршрутизатор
   c. Нормализация данных
   d. Дополнительные элементы
   e. Конечные точки
3. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, в котором будет выполняться настройка конвейера. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система добавит конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера с помощью кнопки ⌄ в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Точка входа

Точка входа — это элемент конвейера, который отвечает за доставку в систему исходных событий. Имеет один выход и не имеет входов.

Группировка поступающих событий по IP-адресу доступна только для точек входа следующих типов:

- Socket;
- Syslog.

Добавьте на конвейер точку входа. При создании выберите нужный тип точки входа и заполните ее параметры.

Маршрутизатор

Маршрутизатор — это элемент конвейера, который позволяет на основе заданных фильтров (маршрутов) направлять события на разные этапы обработки в зависимости от их содержимого или других условий. На входе маршрутизатора каждое событие проверяется по условиям маршрутов и появляется на выходах тех маршрутов, где условие выполняется.

Чтобы настроить маршрутизацию событий:

1. Добавьте на конвейер маршрутизатор и задайте в нем маршруты, фильтрующие события по IP-адресам.
2. Соедините выход из точки входа со входом маршрутизатора.

Настройки маршрутов

Для задания условий маршрутов по IP-адресам используются поля метаданных событий. Обращение к полям метаданных осуществляется через символ процента (`%`). Набор полей метаданных зависит от типа точки входа.

Метаданные Socket

Поле	Тип	Описание	Пример
`socket.host`	String	IP-адрес источника события.	192.0.2.1

Метаданные Syslog

Поле	Тип	Описание	Пример
`socket.host`	String	IP-адрес источника события.	192.0.2.1

Примеры условий маршрутов

1. Фильтрация событий по IP-адресу 192.0.2.1:

   Пример условия

   ```
   %socket.host == "192.0.2.1"
   ```

2. Фильтрация событий по IP-адресу 192.0.2.1 или 192.0.2.2:

   Пример условия

   ```
   %socket.host == "192.0.2.1" || %socket.host == "192.0.2.2"
   ```

Нормализация данных

На этом этапе поступающие события приводятся к единому формату.

Нормализатор преобразовывает события с помощью установленных на нем правил нормализации. Этот элемент конвейера принимает на вход события, проводит их нормализацию и на выходе предоставляет нормализованные события.

Чтобы добавить нормализатор, необходимо создать и установить на нем правило нормализации, согласно которому нормализатор будет обрабатывать поступающие на него данные и нормализовывать их.

Чтобы настроить нормализацию данных:

1. Перейдите в раздел **Экспертиза**.
2. Создайте правило нормализации.

   Если события с разных IP-адресов необходимо нормализовать по-разному, создайте несколько соответствующих правил нормализации.

3. Откройте окно конфигурации конвейера, созданного ранее. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
4. Добавьте на конвейер нормализаторы, используя ранее созданные правила нормализации. Необходимо добавить на конвейер по одному нормализатору на каждый маршрут.
5. Соедините выходы маршрутизатора с входами нужных нормализаторов.

Дополнительные элементы

При необходимости добавьте на конвейер прочие элементы для дополнительной обработки и анализа событий.

Конечные точки

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему. Имеет один вход и не имеет выходов.

Чтобы настроить отправку событий:

1. Добавьте на конвейер одну или несколько конечных точек, в которых будут сохраняться события.
2. Соедините выходы из нормализаторов или из дополнительных элементов со входами конечных точек.

Включение коллектора и установка конфигурации конвейера



Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события начали поступать на конвейер и обрабатываться в соответствии с заданными правилами и условиями:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

Сбор событий из вынесенного коллектора

Данное руководство описывает процесс настройки сбора событий из коллектора, расположенного в сателлите, и их обработки в коллекторе в центральном пространстве системы.

О сборе событий из вынесенного коллектора

Система позволяет создавать коллекторы, расположенные в сателлитах. Такие коллекторы называются **вынесенными**.

Сателлиты не обладают полным объемом функциональности системы. В частности, в сателлитах отсутствует возможность сохранять события в хранилище, так как хранилища событий подключаются к центральному пространству. Таким образом, чтобы отправлять на хранение события, собранные в вынесенном коллекторе, необходимо настроить их отправку из вынесенного коллектора в коллектор центрального пространства через глобальную шину.

Настройка конфигурации для сбора событий из вынесенного коллектора и их дальнейшей обработки в коллекторе центрального пространства системы включает следующие этапы:

1. Подключение к пространству сателлита
2. Настройка отправки событий из вынесенного коллектора
    a. Создание вынесенного коллектора
    b. Создание конвейера в вынесенном коллекторе
    c. Добавление точки входа для сбора событий

d. Настройка отправки событий через глобальную шину
3. Настройка сбора и обработки событий в центральном пространстве
   a. Создание коллектора в центральном пространстве
   b. Создание конвейера в коллекторе центрального пространства
   c. Настройка получения событий из глобальной шины
   d. Настройка обработки событий
   e. Настройка отправки событий на хранение
4. Включение коллектора и установка конфигурации конвейеров

Подключение к пространству сателлита

Управление сателлитом осуществляется за счет запускаемых в нем сервисов через веб-интерфейс основного экземпляра системы. Чтобы подключиться к сателлиту, необходимо создать подключение к его пространству. Для этого:

1. Войдите в веб-интерфейс основного экземпляра системы.
2. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**.
3. Нажмите на кнопку **Создать** (┼). Отобразится окно создания подключения.
4. Введите идентификатор пространства сателлита в поле **ID**.
5. Введите название подключения. Название может быть любым.
6. В поле **IP или домен** введите IP-адрес или доменное имя (FQDN) одного из узлов кластера, в котором развернут сателлит.
7. В поле **Порт** введите номер TCP-порта, указанного на этапе **NATS satellite setup** установки сателлита.
8. При необходимости введите описание подключения к пространству.
9. Нажмите на кнопку **Создать**. Система создаст подключение к пространству и отобразит соответствующее уведомление. Подключение отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Пространства**.

У созданного подключения отобразится статус **Подключается**. По завершении подключения статус изменится на **Активен**.



Время на инициализацию подключения зависит загруженности кластеров и каналов связи и может достигать 5—10 минут.

Настройка отправки событий из вынесенного коллектора

Настройте отправку событий в вынесенном коллекторе, выполнив следующие шаги.

Создание вынесенного коллектора

Чтобы создать вынесенный коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся

коллекторах.

2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, созданное на этапе Подключение к пространству сателлита текущей инструкции.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и — .



- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов на каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.

6. При необходимости введите описание коллектора.
7. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
   - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
   - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
   - **Info** — записывается общая информация о работе и состоянии коллектора.
   - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
   - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.
8. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Создание конвейера в вынесенном коллекторе

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного на предыдущем шаге. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление точки входа для сбора событий

Точка входа поставляет в систему поток сырых событий. Выбор формата точки входа зависит от формата данных, получаемых из источника события.

Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

Чтобы добавить точку входа на конвейер:

1. Откройте конфигурацию конвейера, созданного на предыдущем шаге. Для этого:
   a. Находясь на вкладке **Конвейеры** в карточке коллектора, нажмите на кнопку ⌄ в строке конвейера. Отобразится карточка конвейера.
   b. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
4. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
5. Введите название точки входа.
6. Выберите тип точки входа и задайте ее параметры. Набор параметров точки входа зависит от выбранного типа.
7. Если требуется отслеживать состояние источника данных, выберите политику аудита источников. В противном случае установите переключатель **Аудит** в неактивное положение.

   По умолчанию выбрана системная **Политика аудита источников по точке входа**.

   Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.

8. Если заданные настройки точки входа необходимо сохранить в качестве шаблона для дальнейшего использования, установите флажок **Сохранить как шаблон**.
9. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

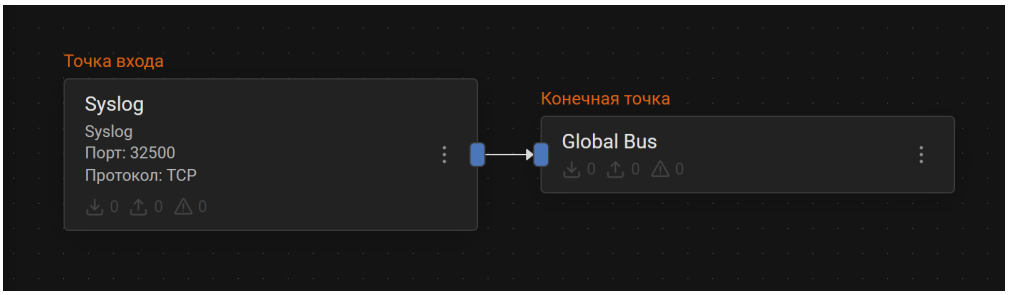
Пример:

Добавим соединение с источником, который поставляет данные в формате системного журнала. Для этого выберем тип точки входа Syslog.

Настройка отправки событий через глобальную шину

Чтобы отправлять события из вынесенного коллектора в коллектор центрального пространства, необходимо добавить конечную точку типа **Глобальная шина**. Для этого:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная точка**. Отобразится окно добавления конечной точки.
2. Введите название конечной точки.
3. Выберите тип конечной точки **Глобальная шина**.
4. Нажмите на кнопку **Добавить**. Новая конечная точка отобразится в конфигурации конвейера.
5. Соедините выход из точки входа, созданной на предыдущем шаге, со входом конечной точки.



Настройка сбора и обработки событий в центральном пространстве

Настройте сбор событий из вынесенного коллектора в коллекторе центрального пространства, выполнив следующие шаги.

Создание коллектора в центральном пространстве

Чтобы создать коллектор в центральном пространстве:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству **Центральное пространство**.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.



- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов на каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.

6. При необходимости введите описание коллектора.
7. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:

- **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
- **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
- **Info** — записывается общая информация о работе и состоянии коллектора.
- **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
- **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

8. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Создание конвейера в коллекторе центрального пространства

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного на предыдущем шаге. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Настройка получения событий из глобальной шины

Чтобы получать события из вынесенного коллектора, добавьте на конвейер точку входа Global Bus. Для этого:

1. Откройте конфигурацию конвейера, созданного на предыдущем шаге. Для этого:
   a. Находясь на вкладке **Конвейеры** в карточке коллектора, нажмите на кнопку ⌄ в строке конвейера. Отобразится карточка конвейера.
   b. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**.

Отобразится окно добавления точки входа.

4. Введите название точки входа.
5. Выберите тип точки входа **Global Bus**.
6. Выберите коллектор, настроенный на этапе Настройка отправки событий из вынесенного коллектора текущей инструкции.
7. Нажмите на кнопку **Добавить**. Новая точка входа отобразится в конфигурации конвейера.

Настройка обработки событий

При необходимости добавьте на конвейер прочие элементы для дополнительной обработки и анализа событий.

Пример:

Добавим VRL-трансформацию со следующим кодом:

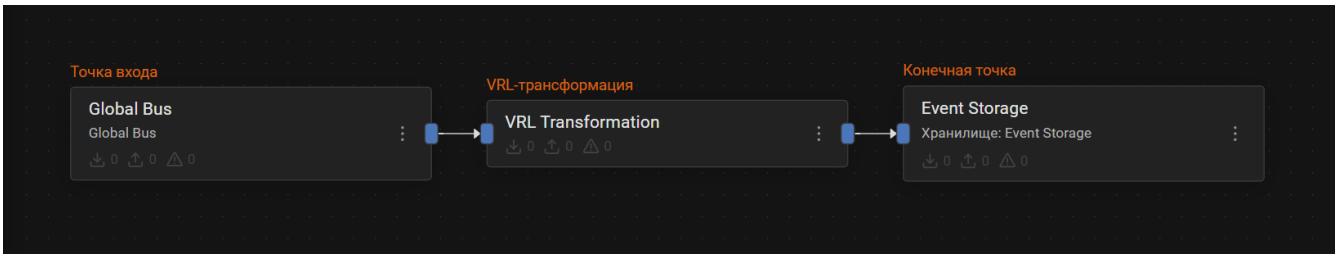
```
.dvendor = "Linux"
.dproduct = "Some Product"
.dvchost = .raw.hostname
.msg = .raw.message
```

Соединим выход точки входа со входом VRL-трансформации.

Настройка отправки событий на хранение

Чтобы отправить событие на хранение, необходимо добавить на конвейер конечную точку типа **Хранилище событий**. Для этого:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная точка**. Отобразится окно добавления конечной точки.
2. Введите название конечной точки.
3. Выберите тип конечной точки **Хранилище событий**.
4. Выберите хранилище событий, в котором необходимо сохранять события.
5. Нажмите на кнопку **Добавить**. Новая конечная точка отобразится на диаграмме конфигурации.
6. Соедините выходы точки входа или дополнительных элементов со входом конечной точки.



Включение коллектора и установка конфигурации конвейеров

Чтобы события начали поступать в вынесенный коллектор и отправляться в коллектор

центрального пространства, повторите для каждого коллектора следующие действия:

1. Перейдите в карточку коллектора и включите его.
2. Установите текущую конфигурацию на конвейере.

Сбор событий с агентов

Данное руководство описывает процесс настройки сбора событий с агентов и их дальнейшей обработки в коллекторе.

О сборе событий с агентов

Агенты позволяют собирать следующие события на устройствах, на которых они установлены:

- входы и выходы пользователей;
- создание процессов;
- добавление или изменение сервисов;
- управление пользователями;
- действия с объектами.



Настройки сбора событий задаются в конфигурации группы агентов.



Подробности о сборе событий на агентах и их структуре приведены в разделе Сбор событий ИБ на агентах.

Для сбора и обработки событий с агентов на конвейере необходимо подключить агент к центральному кластеру системы с помощью шлюза, а затем настроить получение событий из этого шлюза через точку входа R-Vision Endpoint.

Настройка конфигурации для сбора событий с агентов и их дальнейшей обработки в коллекторе включает следующие этапы:

1. Подключение агента к центральному кластеру
2. Настройка сбора событий и обработки событий с агента
    a. Создание коллектора
    b. Создание конвейера
    c. Добавление точки входа R-Vision Endpoint
    d. Настройка обработки событий
    e. Настройка отправки событий на хранение
3. Включение коллектора и установка конфигурации конвейеров

Подключение агента к центральному кластеру

Чтобы подключить агент к центральному кластеру системы, обратитесь к следующим разделам:

- Подключение агента к системе — если необходимо подключение без шифрования.
- Подключение агента с шифрованием — если необходимо TLS-шифрование соединения.

После подключения агента откройте карточку шлюза и скопируйте его идентификатор. Для этого:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на строку шлюза, через который агент был подключен к кластеру. Система отобразит в правой части экрана карточку этого шлюза с подробной информацией о нем.
3. Скопируйте значение поля **ID шлюза** в буфер обмена.

Настройка сбора событий и обработки событий с агента

Создание коллектора

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (＋). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к центральному пространству.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок ＋ и —.



- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов на каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.

6. При необходимости введите описание коллектора.
7. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
   - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
   - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
   - **Info** — записывается общая информация о работе и состоянии коллектора.
   - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
   - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.
8. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Создание конвейера

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного на предыдущем шаге. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление точки входа R-Vision Endpoint

Точка входа типа R-Vision Endpoint предназначена для получения событий с агентов.

Чтобы добавить точку входа R-Vision Endpoint на конвейер:

1. Откройте конфигурацию конвейера, созданного на предыдущем шаге. Для этого:
   a. Находясь на вкладке **Конвейеры** в карточке коллектора, нажмите на кнопку ⌄ в строке конвейера. Отобразится карточка конвейера.
   b. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
4. Введите название точки входа.
5. Выберите тип точки входа **R-Vision Endpoint**.
6. Введите домен для подключения к шлюзу в формате `gw-<gateway_id>`. Здесь `<gateway_id>` — идентификатор шлюза, скопированный на шаге Подключение агента к центральному кластеру.



Например, для шлюза с идентификатором **0f1c5440-5e56-4d0e-bd8b-ea9eb43f28d2** необходимо указать следующий домен:

```
gw-0f1c5440-5e56-4d0e-bd8b-ea9eb43f28d2
```

7. Нажмите на кнопку **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Настройка обработки событий

При необходимости добавьте на конвейер прочие элементы для дополнительной обработки и анализа событий.

Пример:

Представим, что агент установлен на устройстве с ОС Windows и с него поступают события авторизации пользователей.

▼ *Пример события*


```
{
  "raw": {
    "agent": {
      "id": "0ccaa170-25a7-5a0e-a748-f52d118833b1",
      "ip": "198.51.100.0",
      "name": "rpoint-agent-windows.rvision.local"
    },
    "data": {
      "win": {
        "eventdata": {
          "authenticationPackageName": "NTLM",
          "elevatedToken": "%%1843",
          "impersonationLevel": "%%1833",
          "ipAddress": "203.0.113.0",
          "ipPort": "0",
          "keyLength": "128",
          "lmPackageName": "NTLM V2",
          "logonGuid": "{00000000-0000-0000-0000-000000000000}",
          "logonProcessName": "NtLmSsp",
          "logonType": "3",
          "processId": "0x0",
          "subjectLogonId": "0x0",
          "subjectUserSid": "S-1-0-0",
          "targetDomainName": "rpoint-agent-windows.rvision.local",
          "targetLinkedLogonId": "0x0",
          "targetLogonId": "0x1449da49",
          "targetUserName": "user",
          "targetUserSid": "S-1-5-21-3815204242-447739961-2007627207-1002",
          "virtualAccount": "%%1843"
        },
        "system": {
          "channel": "Security",
          "computer": "rpoint-agent-windows.rvision.local",
          "eventID": "4624",
          "eventRecordID": "23886",
          "keywords": "0x8020000000000000",
          "level": "0",
```

```
          "message": "\"\"",
          "opcode": "0",
          "processID": "688",
          "providerGuid": "{54849625-5478-4994-a5ba-3e3b0328c30d}",
          "providerName": "Microsoft-Windows-Security-Auditing",
          "severityValue": "AUDIT_SUCCESS",
          "systemTime": "2025-06-13T12:19:13.0793292Z",
          "task": "12544",
          "threadID": "11552",
          "version": "2"
        }
      }
    },
    "decoder": {
      "name": "windows_eventchannel"
    },
    "id": "1663071375.6890634",
    "location": "EventChannel",
    "manager": {
      "name": "rpoint-agent.rvision.local"
    },
    "rule": {
      "description": "Successful Remote Logon Detected - NTLM authentication,
possible pass-the-hash attack.",
      "firedtimes": 1,
      "gdpr": [
        "IV_32.2"
      ],
      "gpg13": [
        "7.1",
        "7.2"
      ],
      "groups": [
        "win_evt_channel",
        "authentication_success"
      ],
      "hipaa": [
        "164.312.b"
      ],
      "id": "92652",
      "level": 6,
      "mail": false,
      "mitre": {
        "id": [
          "T1550.002"
        ],
        "tactic": [
          "Defense Evasion",
          "Lateral Movement"
        ],
        "technique": [
```

```
          "Pass the Hash"
        ]
      },
      "nist_800_53": [
        "AC.7",
        "AU.14"
      ],
      "pci_dss": [
        "10.2.5"
      ],
      "tsc": [
        "CC6.8",
        "CC7.2",
        "CC7.3"
      ]
    },
    "timestamp": "2025-06-13T15:16:15.696+0300"
  }
}
```


Пусть нам требуется сохранять только записи об успешном входе пользователей в систему. Для этого добавим дополнительные элементы конвейера:

1. Основные данные о событии хранятся в поле `raw`. Чтобы упростить работу с этими данными, мы можем перезаписать все событие только данными из поля raw.

   Для этого добавим VRL-трансформацию со следующим кодом:

   ```
   . = .raw
   ```

   ▼ *Пример события после VRL-преобразования*


   ```
   {
     "agent": {
       "id": "0ccaa170-25a7-5a0e-a748-f52d118833b1",
       "ip": "198.51.100.0",
       "name": "rpoint-agent-windows.rvision.local"
     },
     "data": {
       "win": {
         "eventdata": {
           "authenticationPackageName": "NTLM",
           "elevatedToken": "%%1843",
           "impersonationLevel": "%%1833",
           "ipAddress": "203.0.113.0",
           "ipPort": "0",
           "keyLength": "128",
           "lmPackageName": "NTLM V2",
           "logonGuid": "{00000000-0000-0000-0000-000000000000}",
   ```

```
        "logonProcessName": "NtLmSsp",
        "logonType": "3",
        "processId": "0x0",
        "subjectLogonId": "0x0",
        "subjectUserSid": "S-1-0-0",
        "targetDomainName": "rpoint-agent-windows.rvision.local",
        "targetLinkedLogonId": "0x0",
        "targetLogonId": "0x1449da49",
        "targetUserName": "user",
        "targetUserSid": "S-1-5-21-3815204242-447739961-2007627207-1002",
        "virtualAccount": "%%1843"
      },
      "system": {
        "channel": "Security",
        "computer": "rpoint-agent-windows.rvision.local",
        "eventID": "4624",
        "eventRecordID": "23886",
        "keywords": "0x8020000000000000",
        "level": "0",
        "message": "\"\"",
        "opcode": "0",
        "processID": "688",
        "providerGuid": "{54849625-5478-4994-a5ba-3e3b0328c30d}",
        "providerName": "Microsoft-Windows-Security-Auditing",
        "severityValue": "AUDIT_SUCCESS",
        "systemTime": "2025-06-13T12:19:13.0793292Z",
        "task": "12544",
        "threadID": "11552",
        "version": "2"
      }
    }
  },
  "decoder": {
    "name": "windows_eventchannel"
  },
  "id": "1663071375.6890634",
  "location": "EventChannel",
  "manager": {
    "name": "rpoint-agent.rvision.local"
  },
  "rule": {
    "description": "Successful Remote Logon Detected - NTLM authentication,
possible pass-the-hash attack.",
    "firedtimes": 1,
    "gdpr": [
      "IV_32.2"
    ],
    "gpg13": [
      "7.1",
      "7.2"
    ],
```

```
    "groups": [
      "win_evt_channel",
      "authentication_success"
    ],
    "hipaa": [
      "164.312.b"
    ],
    "id": "92652",
    "level": 6,
    "mail": false,
    "mitre": {
      "id": [
        "T1550.002"
      ],
      "tactic": [
        "Defense Evasion",
        "Lateral Movement"
      ],
      "technique": [
        "Pass the Hash"
      ]
    },
    "nist_800_53": [
      "AC.7",
      "AU.14"
    ],
    "pci_dss": [
      "10.2.5"
    ],
    "tsc": [
      "CC6.8",
      "CC7.2",
      "CC7.3"
    ]
  },
  "timestamp": "2025-06-13T15:16:15.696+0300"
}
```


2. Соединим выход из точки входа со входом VRL-трансформации.
3. Чтобы из всего потока событий получить только те, которые относятся к успешному входу в систему, необходимо настроить фильтрацию.

   В Windows Security Log записи об успешном входе имеют идентификатор 4624. Идентификатор записи хранится в поле `.data.win.system.eventID`. Таким образом, нам необходимо получить события, в поле `.data.win.system.eventID` которых содержится значение `4624`.

   Для этого добавим фильтр со следующим условием:

   ```
   (to_string(.data.win.system.eventID) ?? "") == "4624"
   ```

4. Соединим выход из VRL-трансформации со входом фильтра.

Настройка отправки событий на хранение

Чтобы отправить событие на хранение, необходимо добавить на конвейер конечную точку типа **Хранилище событий**. Для этого:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная точка**. Отобразится окно добавления конечной точки.
2. Введите название конечной точки.
3. Выберите тип конечной точки **Хранилище событий**.
4. Выберите хранилище событий, в котором необходимо сохранять события.
5. Нажмите на кнопку **Добавить**. Новая конечная точка отобразится на диаграмме конфигурации.
6. Соедините выходы точки входа или дополнительных элементов со входом конечной точки.



Включение коллектора и установка конфигурации конвейеров

Чтобы события начали собираться на агенте и поступать в коллектор, выполните следующие действия:

1. Перейдите в карточку коллектора и включите его.
2. Установите текущую конфигурацию на конвейере.

Аналитика

В данном разделе описаны методы работы с инструментами аналитики данных в системе.

Содержание раздела:

- Работа с дашбордами
  - Работа с виджетами
  - Рекомендации по работе с дашбордами
- Работа с оповещениями
- Работа с отчетами
- Работа с разделом Поиск

Работа с дашбордами

Данный раздел описывает процесс работы с дашбордами в системе. Работа осуществляется в разделе **Дашборды** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О дашбордах

Дашборды — специальные панели, предназначенные для представления информации по метрикам системы, оповещениям, событиям и активным спискам в графическом виде.

Чтобы визуализировать данные, на дашборд должны быть добавлены **виджеты**, позволяющие отображать эти данные с помощью графиков, диаграмм, карт и таблиц.

Интерфейс раздела

В верхней части раздела отображается название текущего дашборда. Для выбора другого дашборда нажмите на кнопку  справа от названия. Система отобразит выпадающий список доступных дашбордов.



Пользователю доступны общие дашборды и созданные им персональные дашборды. Персональные дашборды обозначены меткой **Персональный**. Метка отображается на панели инструментов раздела, а также в выпадающем списке дашбордов в верхней части раздела.

Справа от названия дашборда отображаются следующие кнопки:

- Кнопка  позволяет добавить новый дашборд.
- Кнопка  позволяет импортировать дашборды в систему.

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Добавить виджет** (+) позволяет создать новый виджет на текущем дашборде.
- Кнопка позволяет изменить текущий дашборд.
- Кнопка позволяет удалить текущий дашборд.
- Кнопка позволяет экспортировать дашборд в формате JSON.
- Кнопка позволяет сохранить дашборд как схему.
- Метка **Персональный** отображается, если дашборд доступен только текущему пользователю.
- Описание дашборда (при наличии) предоставляет краткую информацию о его функциональности.
- Кнопка **Переменные** ({;}) позволяет открыть боковую панель для задания переменных, которые можно использовать в RQL-запросах виджетов.
- Кнопка предназначена для расширения панели дашбордов на всю страницу браузера.

В рабочей области отображаются виджеты выбранного дашборда. Можно перемещать виджеты по рабочей области и изменять их размеры. В верхней части виджета отображается его наименование, а также кнопки:

- — для обновления данных виджета.
- — для изменения или удаления виджета, а также скрытия или отображения его легенды.

Работа с дашбордами

Доступные операции над дашбордами:

- Добавление дашборда
- Выбор и просмотр дашборда
- Расширение дашборда на полный экран
- Задание переменных
- Изменение дашборда
- Настройка доступности дашборда
- Удаление дашборда
- Импорт дашборда
- Экспорт дашборда
- Сохранение дашборда как схемы

Добавление дашборда

Создать дашборд можно следующими способами:

- Добавить дашборд и виджеты вручную.
- Использовать предварительно созданную схему.

Создание дашборда вручную

Чтобы создать дашборд вручную:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на кнопку + в верхней части раздела. Отобразится окно создания дашборда.
3. Введите название дашборда.
4. При необходимости введите описание дашборда.
5. Установите флажок **Персональный**, если дашборд необходимо сделать доступным только для текущего пользователя.
6. Нажмите на кнопку **Добавить**. Система создаст новый дашборд и отобразит соответствующее уведомление. Дашборд отобразится в выпадающем списке в верхней части раздела **Дашборды**. Система автоматически откроет добавленный дашборд для просмотра.

После создания дашборда добавьте в него необходимые виджеты.

Создание дашборда из схемы

Чтобы создать дашборд из схемы:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на кнопку + в верхней части раздела. Отобразится окно создания дашборда.
3. Выберите из выпадающего списка схему дашборда.



Если до выбора схемы были заполнены название и описание дашборда, они будут заменены на данные из схемы.

4. Сопоставьте переменные виджетов с источниками информации — хранилищем событий или активным списком в зависимости от типа виджетов.
5. При необходимости введите название и описание дашборда вместо заполненных значений из схемы.
6. Установите флажок **Персональный**, если дашборд необходимо сделать доступным только для текущего пользователя.
7. Нажмите на кнопку **Добавить**. Система создаст новый дашборд и отобразит соответствующее уведомление. Дашборд отобразится в выпадающем списке в верхней части раздела **Дашборды**. Система автоматически откроет добавленный дашборд для просмотра.

Выбор и просмотр дашборда

Чтобы просмотреть дашборд:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на кнопку ⌄ в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.



Для быстрого поиска дашборда начните вводить его название в строку поиска, расположенную над списком.

Расширение дашборда на полный экран

Чтобы раскрыть дашборд на всю страницу браузера:

1. Перейдите в раздел **Дашборды**
2. Выберите нужный дашборд с помощью кнопки ⌄ в верхней части раздела.
3. Нажмите на кнопку ⤢ в правом верхнем углу раздела. Содержимое дашборда будет расширено на всю страницу браузера.

   В расширенном режиме из верхней части раздела и с панели инструментов скрываются все компоненты, кроме выпадающего списка для выбора дашборда и кнопки ⤡ для возвращения в обычный режим.

В расширенном режиме кнопка ⤢ меняется на ⤡ и предназначена для возвращения дашборда в обычный режим отображения.

Задание переменных

Переменные можно использовать в RQL-запросах при построении виджетов событий и виджетов активных списков.

Чтобы задать переменные:

1. Перейдите в раздел **Дашборды**
2. Выберите нужный дашборд с помощью кнопки ⌄ в верхней части раздела.
3. Нажмите на кнопку **Переменные** ({;}) в верхней части раздела. Система отобразит в правой части экрана панель настройки переменных.
4. Задайте аргументы и значения переменных.

   Чтобы добавить переменную:

   a. Нажмите на кнопку **Добавить**. Появится строка настройки переменной.

   b. Введите аргумент переменной. Аргумент может содержать только латинские буквы (A–Z, a–z) и цифры (0–9).

   c. Выберите тип значения переменной. Доступные типы:

      - **Строка**;
      - **Число**;
      - **Дата**;
      - **Логическое**.

   d. Укажите значение переменной. Способ задания значения зависит от его типа:

      - **Строка** и **Число** — введите значение вручную.
      - **Дата** — введите значение вручную или задайте с помощью панели выбора даты и

времени.

- **Логическое** — выберите значение из выпадающего списка: **True** или **False**.

Чтобы удалить переменную, нажмите на кнопку в строке ее настроек.

5. Нажмите на кнопку **Применить**, чтобы применить заданные переменные и перестроить дашборд с учетом их значений.

   Примененные переменные отобразятся в верхней части рабочей области раздела.

Чтобы сбросить примененную переменную, нажмите на кнопку × справа от нее в верхней части рабочей области.

Чтобы сбросить сразу все переменные, выполните одно из действий:

- Нажмите на кнопку **Сбросить все** (×) в верхней части рабочей области.
- Нажмите на кнопку **Сбросить все** в нижней части панели настроек переменных.

Изменение дашборда

Вы можете изменить дашборд через кнопку на панели инструментов раздела.

Настройка доступности дашборда

Только пользователь, создавший дашборд, может изменить его доступность.

Чтобы настроить доступность дашборда:

1. Перейдите в раздел **Дашборды**
2. Выберите нужный дашборд с помощью кнопки в верхней части раздела.
3. Нажмите на кнопку в верхней части раздела. Отобразится окно изменения дашборда.
4. Настройте доступность дашборда:
   - Чтобы сделать дашборд доступным только для текущего пользователя, установите флажок **Персональный**.
   - Чтобы сделать дашборд доступным для всех пользователей системы, снимите флажок **Персональный**.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения настроек и отобразит соответствующее уведомление. Обновленная информация отобразится на панели инструментов.

Удаление дашборда

Вы можете удалить дашборд через кнопку на панели инструментов раздела.

Вместе с дашбордом также удаляются и все добавленные на него виджеты.

Импорт дашборда

Чтобы импортировать дашборд:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на кнопку в верхней части раздела. Отобразится окно импорта дашборда.
3. Нажмите на кнопку **Выбрать файл** и выберите файл дашборда для импорта. Для импорта доступны только файлы формата JSON размером не более 10 МБ.
4. Нажмите на кнопку **Импортировать**. Система импортирует дашборд и отобразит соответствующее уведомление. Новый дашборд появится в выпадающем списке дашбордов.



Если вы импортируете дашборд с виджетом событий, который ссылается на недоступное хранилище, то вместо данных виджет отобразит ошибку о том, что хранилище не найдено.

Экспорт дашборда

Чтобы экспортировать дашборд:

1. Перейдите в раздел **Дашборды**
2. Выберите нужный дашборд с помощью кнопки в верхней части раздела.
3. Нажмите на кнопку в верхней части раздела. Отобразится окно экспорта дашборда.
4. Нажмите на кнопку **Экспортировать**. Система загрузит данные дашборда на устройство пользователя в формате JSON и отобразит соответствующее уведомление.

Сохранение дашборда как схемы

Чтобы сохранить дашборд как схему:

1. Перейдите в раздел **Дашборды**
2. Выберите нужный дашборд с помощью кнопки в верхней части раздела.
3. Нажмите на кнопку в верхней части раздела. Отобразится окно создания схемы.



Чтобы создать схему на основе дашборда, дашборд должен содержать хотя бы один виджет.

4. По умолчанию название и описание создаваемой схемы идентичны названию и описанию дашборда. При необходимости измените название и описание схемы.
5. Выберите из выпадающего списка каталог, в котором будет создана схема.
6. Для виджетов событий и виджетов активных списков автоматически генерируются переменные, которые при создании дашборда из схемы нужно связать с источниками данных. При необходимости измените переменные виджетов.



Чтобы при создании дашборда несколько виджетов получали данные из одного источника, задайте для них одинаковые переменные.

7. Нажмите на кнопку **Далее**. Откроется окно создания схемы с заполненными данными.
8. Задайте идентификатор схемы дашборда в поле `id`.
9. При необходимости внесите изменения в поля схемы.

10. Нажмите на кнопку **Опубликовать версию**. Система создаст схему дашборда и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.



Чтобы схема дашборда стала доступной для создания дашбордов, ее необходимо включить.

Вы можете создать черновик схемы дашборда без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Работа с виджетами

Данный раздел описывает процесс работы с виджетами в системе. Работа осуществляется в разделе **Дашборды** веб-интерфейса системы.

О виджетах

Виджеты — это элементы дашборда, которые позволяют отображать информацию по метрикам системы, оповещениям, событиям и активным спискам в графическом виде. Виджеты позволяют отображать данные с помощью графиков, диаграмм, карт и таблиц.

Тип виджета зависит от выбранного источника данных. Доступны следующие типы виджетов:

- **Виджеты метрик** — отображают сведения об основных метриках системы.
- **Виджеты оповещений** — отображают сведения об оповещениях в системе.
- **Виджеты событий** — отображают сведения о событиях в системе на основе RQL-запроса. События загружаются из выбранного хранилища событий.
- **Виджеты активных списков** — отображают сведения об активных списках на основе RQL-запроса. Данные загружаются из выбранного активного списка.

Доступные операции над виджетами:

- Добавление виджета
- Обновление виджета
- Изменение размеров виджета
- Перемещение виджета
- Изменение настроек виджета
- Настройка отображения легенды виджета
- Настройка отображения рядов данных виджета
- Просмотр деталей ряда данных виджета
- Создание копии виджета
- Удаление виджета
- Экспорт виджета

Добавление виджета

Способ создания виджета зависит от его источника данных:

- Добавление виджета метрик
- Добавление виджета оповещений
- Добавление виджета событий
- Добавление виджета активного списка

Для виджетов событий и виджетов активных списков доступен выбор типа диаграммы. Особенности каждого типа приведены в разделе Типы диаграмм.

При задании RQL-запроса в виджетах событий и виджетах активных списков доступно использование переменных. Подробности приведены в разделе Работа с переменными в виджетах.

Добавление виджета метрик

Чтобы добавить виджет метрик:

1. Откройте требуемый дашборд.
2. Нажмите на кнопку **Добавить виджет** (+) в верхней части раздела. Отобразится окно создания виджета.
3. Введите название виджета.
4. При необходимости введите описание виджета.
5. Выберите **Метрики** из выпадающего списка **Источник данных**.
6. Выберите метрику для виджета из выпадающего списка:
   - **Хост БД CPU** — график использования ресурсов процессора (CPU) контейнерами БД.
   - **Хост БД RAM** — график использования ресурсов оперативной памяти (RAM) контейнерами БД.
   - **Хост Коллектор CPU** — график использования ресурсов процессора (CPU) контейнерами коллектора.
   - **Хост Коллектор RAM** — график использования ресурсов оперативной памяти (RAM) контейнерами коллектора.
   - **Пайплайн EPS** — график событий в секунду (EPS).
   - **Пайплайн объем трафика** — график объема трафика, поступающего в систему через конвейер.
   - **Пайплайн ErrPS** — график ошибок в секунду (ErrPS).
7. Выберите из выпадающего списка способ задания периода и установите период:
   - **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.
   - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.

8. Для регулярного обновления данных на виджете установите переключатель **Включить автообновление** в активное положение и укажите периодичность обновления:

   a. Задайте частоту обновления вручную или с помощью кнопок — и +.

   b. Выберите из выпадающего списка единицы времени периода.

   Не рекомендуется включать автоматическое обновление данных для каждого виджета дашборда, так как это может излишне нагрузить систему. Подробности приведены в разделе Обновление данных.

9. Чтобы предварительно просмотреть, как будут отображаться данные на виджете в соответствии с заданными настройками, нажмите на кнопку в правой верхней части окна создания виджета.
10. Нажмите на кнопку **Добавить виджет**. Система создаст виджет и отобразит соответствующее уведомление. Новый виджет появится на дашборде.

Добавление виджета оповещений

Чтобы добавить виджет оповещений:

1. Откройте требуемый дашборд.
2. Нажмите на кнопку **Добавить виджет** (+) в верхней части раздела. Отобразится окно создания виджета.
3. Введите название виджета.
4. При необходимости введите описание виджета.
5. Выберите **Оповещения** из выпадающего списка **Источник данных**.
6. Выберите данные оповещений для виджета из выпадающего списка:
   - **Активные** — круговая диаграмма, отображающая данные по количеству оповещений со статусами **Новое** и **В работе**.
   - **Новые** — круговая диаграмма, отображающая данные по количеству оповещений со статусом **Новое** для каждого уровня угрозы.
   - **Последние** — таблица, отображающая сведения о последних оповещениях.

     Необходимо указать количество оповещений и выбрать способ их сортировки: по дате создания или по дате изменения.

     Таблица содержит следующие столбцы:

     - **Создано**/**Изменено** — дата и время создания или последнего изменения оповещения, в зависимости от выбранного способа сортировки.
     - **Название** — название оповещения.
     - **Уровень угрозы** — уровень угрозы оповещения: критический, высокий, средний или низкий.
     - **Статус** — текущий статус оповещения: **Ложное срабатывание**, **В работе**, **Новое** или **Закрыто**.

- **Количество событий** — количество корреляционных событий в сегменте группировки для генерации оповещения.

◦ **Распределение по времени** — график, отображающий количество оповещений за указанный период времени.

Выберите из выпадающего списка способ задания периода и установите период:

- **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.
- **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.

При необходимости можно сгруппировать оповещения по статусам, уровню угрозы или правилам сегментации.

◦ **Распределение по атрибуту** — круговая диаграмма, отображающая данные по количеству оповещений для каждого значения выбранного атрибута.

Необходимо выбрать атрибут для группировки оповещений: статус, уровень угрозы или правило сегментации.

◦ **Итоги** — итоги, отображающие количество оповещений с заданными настройками.

Необходимо выбрать атрибут оповещения и указать значение атрибута. Доступны следующие атрибуты и их значения:

- **Статус**: новое, в работе, закрыто, ложное срабатывание.
- **Уровень угрозы**: критический, высокий, средний, низкий.
- **Правило сегментации**: идентификатор правила сегментации.

7. Для регулярного обновления данных на виджете установите переключатель **Включить автообновление** в активное положение и укажите периодичность обновления:
   a. Задайте частоту обновления вручную или с помощью кнопок — и +.
   b. Выберите из выпадающего списка единицы времени периода.



Не рекомендуется включать автоматическое обновление данных для каждого виджета дашборда, так как это может излишне нагрузить систему. Подробности приведены в разделе Обновление данных.

8. Чтобы предварительно просмотреть, как будут отображаться данные на виджете в соответствии с заданными настройками, нажмите на кнопку ⟳ в правой верхней части окна создания виджета.
9. Нажмите на кнопку **Добавить виджет**. Система создаст виджет и отобразит соответствующее уведомление. Новый виджет появится на дашборде.

Добавление виджета событий

Чтобы добавить виджет событий:

1. Откройте требуемый дашборд.
2. Нажмите на кнопку **Добавить виджет** (+) в верхней части раздела. Отобразится окно создания виджета.
3. Введите название виджета.
4. При необходимости введите описание виджета.
5. Выберите **События** из выпадающего списка **Источник данных**.
6. Выберите хранилище событий, из которого требуется брать информацию о событиях, из выпадающего списка.
7. Выберите тип визуализации для виджета из выпадающего списка. Доступные типы:
   - **Столбчатая диаграмма** — отображает данные в виде столбцов. На один столбец приходится одно значение атрибута из RQL-запроса.
   - **График** — отображает данные в виде кривой на графике с осями X и Y.
   - **Круговая диаграмма** — отображает данные в виде секторов на круговой диаграмме.
   - **Столбчатая диаграмма с накоплением** — отображает данные в виде сгруппированных столбцов. Столбцы в каждой группе расположены друг поверх друга, формируя единый столбец в рамках группы.
   - **Таблица** — отображает данные в табличном виде.
   - **Итоги** — отображает одно выбранное значение на виджете.
   - **Карта с группировкой** — отображает данные на карте мира с группировкой по координатам.
   - **Карта стран мира** — отображает данные на карте мира с группировкой по странам.
8. Задайте лимит строк для виджетов вручную или с помощью кнопок — и +. Лимит строк определяет максимальное количество строк, которые будут использованы для построения виджета. Допустимый диапазон лимита строк: 1 — 100 000.



Рекомендуется задавать адекватные лимиты или использовать лимиты по умолчанию, чтобы не перегрузить систему. Подробности приведены в разделе Запросы к данным.

По умолчанию для таблиц и круговых диаграмм установлен лимит в 100 строк, для остальных типов визуализации — 1000 строк.



Лимит строк также можно задать в RQL-запросе с помощью оператора `LIMIT`. В этом случае лимит определяется минимальным из значений поля **Лимит строк** и оператора `LIMIT`. Например:

- Если в поле **Лимит строк** задано значение `100`, а в запросе указано `LIMIT 20`, то будет установлен лимит в 20 строк.
- Если в поле **Лимит строк** задано значение `20`, а в запросе указано `LIMIT 100`, то также будет установлен лимит в 20 строк.



Если задан лимит более 6 строк, то на диаграммах типов **График**, **Столбчатая диаграмма** и **Столбчатая диаграмма с накоплением** будет отображаться

только часть подписей, с равномерными пропусками, чтобы избежать наложения.

9. Введите запрос на языке RQL в поле запроса.

   При необходимости вы можете вставить RQL-запрос из поискового шаблона. Для этого выберите нужный шаблон из выпадающего списка **Шаблон запроса** ( ). Для быстрого поиска шаблона в выпадающем списке начните вводить его название в строке поиска.

   При составлении RQL-запроса необходимо учитывать особенности каждого типа диаграммы. Подробности приведены в разделе Типы диаграмм.

   При необходимости в секции **Переменные** вы можете задать переменные, которые можно использовать в RQL-запросе. Подробности приведены в разделе Работа с переменными в виджетах.

10. Заполните поля настроек диаграммы.

    Для каждого типа диаграммы доступен свой набор полей. Подробности приведены в разделе Типы диаграмм.

11. Выберите из выпадающего списка способ задания периода и установите период:

    - **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.
    - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.

12. Для регулярного обновления данных на виджете установите переключатель **Включить автообновление** в активное положение и укажите периодичность обновления:

    a. Задайте частоту обновления вручную или с помощью кнопок — и +.

    b. Выберите из выпадающего списка единицы времени периода.

    Не рекомендуется включать автоматическое обновление данных для каждого виджета дашборда, так как это может излишне нагрузить систему. Подробности приведены в разделе Обновление данных.

13. Чтобы предварительно просмотреть, как будут отображаться данные на виджете в соответствии с заданными настройками, нажмите на кнопку в правой верхней части окна создания виджета. Доступны следующие варианты предпросмотра:

    - **Диаграмма** — отображает визуальное представление виджета на дашборде.
    - **Данные** — отображает результаты RQL-запроса в табличной форме.

14. Нажмите на кнопку **Добавить виджет**. Система создаст виджет и отобразит соответствующее уведомление. Новый виджет появится на дашборде.



Вы также можете создать виджет событий в разделе **Поиск**:

- на основе результатов поиска: с помощью кнопки на панели инструментов раздела.
- на основе поля статистики: с помощью кнопки **Создать виджет** () на панели детализированной статистики по полю.

Добавление виджета активного списка

Чтобы добавить виджет активного списка:

1. Откройте требуемый дашборд.
2. Нажмите на кнопку **Добавить виджет** ( ) в верхней части раздела. Отобразится окно создания виджета.
3. Введите название виджета.
4. При необходимости введите описание виджета.
5. Выберите **Активный список** из выпадающего списка **Источник данных**.
6. Выберите активный список, из которого требуется брать информацию о записях.
7. Выберите тип визуализации для виджета из выпадающего списка. Доступные типы:
   - **Столбчатая диаграмма** — отображает данные в виде столбцов. На один столбец приходится одно значение атрибута из RQL-запроса.
   - **График** — отображает данные в виде кривой на графике с осями X и Y.
   - **Круговая диаграмма** — отображает данные в виде секторов на круговой диаграмме.
   - **Столбчатая диаграмма с накоплением** — отображает данные в виде сгруппированных столбцов. Столбцы в каждой группе расположены друг поверх друга, формируя единый столбец в рамках группы.
   - **Таблица** — отображает данные в табличном виде.
   - **Итоги** — отображает одно выбранное значение на виджете.
   - **Карта с группировкой** — отображает данные на карте мира с группировкой по координатам.
   - **Карта стран мира** — отображает данные на карте мира с группировкой по странам.
8. Задайте лимит строк для виджетов вручную или с помощью кнопок — и +. Лимит строк определяет максимальное количество строк, которые будут использованы для построения виджета. Допустимый диапазон лимита строк: 1 — 100 000.



Рекомендуется задавать адекватные лимиты или использовать лимиты по умолчанию, чтобы не перегрузить систему. Подробности приведены в разделе Запросы к данным.

По умолчанию для таблиц и круговых диаграмм установлен лимит в 100 строк, для остальных типов визуализации — 1000 строк.



Лимит строк также можно задать в RQL-запросе с помощью оператора `LIMIT`. В этом случае лимит определяется минимальным из значений поля **Лимит строк**

и оператора `LIMIT`. Например:

- Если в поле **Лимит строк** задано значение `100`, а в запросе указано `LIMIT 20`, то будет установлен лимит в 20 строк.
- Если в поле **Лимит строк** задано значение `20`, а в запросе указано `LIMIT 100`, то также будет установлен лимит в 20 строк.



Если задан лимит более 6 строк, то на диаграммах типов **График**, **Столбчатая диаграмма** и **Столбчатая диаграмма с накоплением** будет отображаться только часть подписей, с равномерными пропусками, чтобы избежать наложения.

9. Выберите из выпадающего списка способ задания периода и установите период:
   - **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.
   - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.
10. Введите запрос на языке RQL в поле запроса.

    При необходимости вы можете вставить RQL-запрос из поискового шаблона. Для этого выберите нужный шаблон из выпадающего списка **Шаблон запроса** (⌄). Для быстрого поиска шаблона в выпадающем списке начните вводить его название в строке поиска.

    При составлении RQL-запроса необходимо учитывать особенности каждого типа диаграммы. Подробности приведены в разделе Типы диаграмм.

    При необходимости в секции **Переменные** вы можете задать переменные, которые можно использовать в RQL-запросе. Подробности приведены в разделе Работа с переменными в виджетах.

11. Заполните поля настроек диаграммы.

    Для каждого типа диаграммы доступен свой набор полей. Подробности приведены в разделе Типы диаграмм.

12. Выберите из выпадающего списка способ задания периода и установите период:
    - **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.
    - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.
13. Для регулярного обновления данных на виджете установите переключатель **Включить автообновление** в активное положение и укажите периодичность обновления:

a. Задайте частоту обновления вручную или с помощью кнопок — и +.

b. Выберите из выпадающего списка единицы времени периода.



Не рекомендуется включать автоматическое обновление данных для каждого виджета дашборда, так как это может излишне нагрузить систему. Подробности приведены в разделе Обновление данных.

14. Чтобы предварительно просмотреть, как будут отображаться данные на виджете в соответствии с заданными настройками, нажмите на кнопку в правой верхней части окна создания виджета. Доступны следующие варианты предпросмотра:
    - **Диаграмма** — отображает визуальное представление виджета на дашборде.
    - **Данные** — отображает результаты RQL-запроса в табличной форме.
15. Нажмите на кнопку **Добавить виджет**. Система создаст виджет и отобразит соответствующее уведомление. Новый виджет появится на дашборде.



Вы также можете создать виджет активного списка, открыв панель статистики по полю в разделе **Поиск**.

Типы диаграмм

Выбор типа диаграммы доступен только для источников данных **События** и **Активные списки**. Для каждого типа диаграммы доступен свой набор полей в секции **Сопоставление полей**, которые необходимо заполнить атрибутами из RQL-запроса.

Доступны следующие типы диаграмм:

- Столбчатые диаграммы
- Карта стран мира
- Карта с группировкой
- Круговая диаграмма
- График
- Таблица
- Итоги

Столбчатые диаграммы

Система позволяет создавать столбчатые диаграммы следующих типов:

- **Столбчатая диаграмма** — позволяет отображать значения показателей в виде вертикальных столбцов. Высота столбцов пропорциональна значениям.
- **Столбчатая диаграмма с накоплением** — позволяет сравнивать сгруппированные показатели с другими группами тех же показателей, например, распределение количества событий по IP-адресу источника с их группировкой по типу. Столбцы значений в каждой группе расположены друг поверх друга, формируя единый столбец в рамках группы.

Для столбчатых диаграмм доступны следующие поля настроек:

- **Легенда** — позволяет указать атрибут, по значению которого формируются столбцы виджета.
- **Значение** — позволяет указать атрибут, по значению которого определяется шкала виджета и высота столбцов.
- **Группировка** — позволяет указать атрибут, по которому производится группировка данных в легенде виджета.



При заполнении поля **Группировка** в легенде виджета отображаются только первые 20 групп.



Поле **Группировка** не влияет на группировку данных на диаграмме.



Через легенду виджета можно включать и выключать отображение данных на диаграмме нажатием на цветовое обозначение соответствующего ряда данных.

Примеры виджета:

Для событий

Необходимо построить диаграмму, показывающую распределение количества событий по IP-адресу источника с их группировкой по типу.

Запрос:

```
SELECT sourceIp, type, count(id) as cnt GROUP BY type
```

Здесь:

- в поле **Легенда** указывается `sourceIp` — атрибут, возвращающий IP-адрес источника;
- в поле **Значение** указывается `cnt` — псевдоним атрибута, возвращающего количество событий;
- в поле **Группировка** указывается `type` — атрибут, возвращающий тип события.

Для активных списков

Необходимо построить диаграмму, показывающую распределение количества IP-адресов DDOS-атак по IP-адресам с их группировкой по типам.

Запрос:

```
SELECT ip, attack_type, (count(id) AS cnt GROUP BY Ip, attack_type
```

Здесь:

- в поле **Легенда** указывается `ip` — атрибут, возвращающий IP-адреса, замеченные в атаке;
- в поле **Значение** указывается `cnt` — псевдоним атрибута, возвращающего количество записей;

- в поле **Группировка** указывается `attack_type` — атрибут, возвращающий тип атаки.

Карта стран мира

Виджет **Карта стран мира** отображает количество событий или данных из активных списков на карте мира с группировкой по странам.

Для карты стран мира доступны следующие поля настроек:

- **Страна** — позволяет указать атрибут, возвращающий название страны.

  По атрибуту в поле **Страна** необходимо группировать данные в RQL-запросе.

- **Количество** — позволяет указать атрибут или псевдоним атрибута, который передает количество данных.

  Для карты стран мира нужно группировать данные в RQL-запросе только по полю, значение которого идентифицирует страну. При группировке по другим полям виджет может работать некорректно.

Примеры виджета:

Для событий

Необходимо отобразить количество событий с группировкой по странам.

Запрос:

```
SELECT country, count(*) as cnt GROUP BY country
```

Здесь:

- в поле **Страна** указывается `country` — атрибут, возвращающий название страны;
- в поле **Количество** указывается `cnt` — псевдоним атрибута, возвращающего количество событий.

Для активных списков

Необходимо отобразить количество DDOS-атак с группировкой по странам мира.

Запрос:

```
SELECT country, count(*) as cnt WHERE attack_type IN 'DDOS' GROUP BY country
```

Здесь:

- в поле **Страна** указывается `country` — атрибут, возвращающий название страны.
- в поле **Количество** указывается `cnt` — псевдоним атрибута, возвращающего количество

атак.

Карта с группировкой

Виджет **Карта с группировкой** позволяет отображать данные с группировкой по координатам. Группировка на виджете происходит методом кластеризации — когда координаты группируются по сходству их местоположения. При приближении карты координаты разбиваются на точки, которые соответствуют местоположению их координат. При отдалении — точки, находящиеся рядом, собираются в кластер (единую точку).

Для карты с группировкой также доступна дополнительная секция настроек **Координаты**.

Секция **Координаты** содержит следующие поля:

- **Широта** — позволяет указать атрибут, возвращающий широту.
- **Долгота** — позволяет указать атрибут, возвращающий долготу.



При составлении RQL-запроса для виджета **Карта с группировкой** не нужно использовать оператор `GROUP BY`. Группировка происходит автоматически по указанным атрибутам координат.

Секция **Сопоставления полей** содержит поля настройки таблицы в окне детализации данных по координатам:

- **Название колонки** — позволяет задать название столбца в таблице.
- **Атрибут из запроса** — позволяет указать атрибут, значение которого будет выводиться в столбце.

Для добавления нового столбца в таблицу используйте кнопку **Добавить колонку** (+). Для удаления столбца из таблицы используйте кнопку  .



Окно детализации данных открывается при нажатии на точку координат.

Примеры виджета:

Для событий

Необходимо отобразить все события с группировкой по координатам, у которых есть связанный инцидент. Таблица должна содержать координаты событий и информацию по инцидентам: ID, название и время.

Запрос:

```
SELECT latitude, longitude, incidentId, incidentName, incidentTimestamp WHERE
incidentId is not null
```

Название колонки	Атрибут из запроса
Широта	`latitude`
Долгота	`longitude`
ID инцидента	`incidentId`
Название инцидента	`incidentName`
Время инцидента	`incidentTimestamp` — атрибут, возвращающий время, когда произошел инцидент

Для активных списков

Необходимо отобразить местоположение IP-адресов, которые были замечены в DDOS-атаке. Таблица должна содержать IP-адреса источников атаки, координаты, название стран и количество запросов с IP-адреса.

Запрос:

```
SELECT ip, involved_attacks, country, latitude, longitude WHERE attack_type IN
'DDOS'
```

Название колонки	Атрибут из запроса
IP	`ip` — атрибут, возвращающий IP-адрес источника атаки
Количество запросов	`involved_attacks`
Широта	`latitude`
Долгота	`longitude`
Страна	`country`

Круговая диаграмма

Виджет **Круговая диаграмма** позволяет отобразить данные в виде секторов на круговой диаграмме.

Для круговых диаграмм доступны следующие поля настроек:

- **Легенда** — позволяет указать атрибут, который описывает соответствующий сектор на круговой диаграмме.
- **Значение** — позволяет указать атрибут, по значениям которого формируются сектора на круговой диаграмме.

Примеры виджета:

Для событий

Необходимо построить диаграмму, показывающую распределение количества событий по ID коллекторов.

Запрос:

```
SELECT count(*) as cnt, collectorId GROUP BY collectorId
```

Здесь:

- в поле **Легенда** указывается `collectorId` — атрибут, возвращающий ID коллектора;
- в поле **Значение** указывается `cnt` — псевдоним атрибута, возвращающего количество событий.

Для активных списков

Необходимо построить диаграмму, показывающую распределение количества атак по типам.

Запрос:

```
SELECT attack_type, sum(involved_attacks) AS total_attacks GROUP BY attack_type
```

Здесь:

- в поле **Легенда** указывается `attack_type` — атрибут, возвращающий тип атаки;
- в поле **Значение** указывается `total_attacks` — псевдоним атрибута, возвращающего суммарное количество атак.

График

Виджет **График** позволяет отобразить данные в виде кривой на графике с осями X и Y.

Для графиков доступны следующие поля настроек:

- **Ось X** — позволяет задать атрибут, значения которого будут отображаться на оси X и по значениям которого будет высчитываться положение графика по оси X.
- **Ось Y** — позволяет задать атрибут, значения которого будут отображаться на оси Y и по значениям которого будет высчитываться положение графика по оси Y.
- **Группировка** — позволяет указать атрибут, по которому производится дополнительная группировка. Одна кривая графика делится на несколько кривых, которые сгруппированы по атрибуту, указанному в поле.



Параметр **Группировка** указывается опционально и влияет на отображение легенды виджета.



При заполнении поля **Группировка** в легенде виджета отображаются только первые 20 групп.

Примеры виджета:

Для событий

Необходимо построить график, показывающий зависимость количества событий от ID коллекторов.

Запрос:

```
SELECT collectorId, count(*) AS cnt WHERE collectorId like '%' GROUP BY
collectorId
```

Здесь:

- в поле **Ось X** указывается `collectorId` — атрибут, возвращающий ID коллектора;
- в поле **Ось Y** указывается `cnt` — псевдоним атрибута, возвращающего количество событий;

Для активных списков

Необходимо построить график, показывающий зависимость количества атак с каждого IP-адреса в зависимости от оценки его репутации.

Запрос:

```
SELECT ip, reputation_score, sum(involved_attacks) AS total_attacks GROUP BY
ip, reputation_score
```

Здесь:

- в поле **Ось X** указывается `reputation_score` — атрибут, возвращающий оценку рейтинга;
- в поле **Ось Y** указывается `total_attacks` — псевдоним атрибута, возвращающего суммарное количество атак.

Таблица

Виджет **Таблица** позволяет отображать данные в табличном виде.

Для таблиц доступны следующие поля настроек столбцов таблицы:

- **Название колонки** — позволяет задать название столбца табличной части.
- **Атрибут из запроса** — позволяет задать атрибут, значение которого будет выводиться в столбце.

Для добавления нового столбца в таблицу используйте кнопку **Добавить колонку** (+). Для удаления столбца из таблицы используйте кнопку  .

Примеры составления виджета:

Для событий

Необходимо создать таблицу с двумя столбцами, позволяющую просмотреть соотношение

количества событий для 10 последних коллекторов.

Запрос:

```
SELECT collectorId, count(*) AS cnt WHERE collectorId like '%' GROUP BY
collectorId
```

Название колонки	Атрибут из запроса
Коллектор	`collectorId` — атрибут, возвращающий ID коллектора
Количество событий	`cnt` — псевдоним атрибута, возвращающего количество событий

В поле **Лимит строк** указывается значение `10`, что позволяет вывести в таблицу не более 10 записей.

Лимит строк также можно указать в запросе с помощью оператора `LIMIT`:

```
SELECT collectorId, count(*) AS cnt WHERE collectorId like '%' GROUP BY
collectorId LIMIT 10
```

В этом случае лимит строк в таблице определяется минимальным из значений поля **Лимит строк** и оператора `LIMIT`.

Для активных списков

Необходимо создать таблицу с информацией об IP-адресах из определенной подсети, замеченных в DDOS-атаках. Таблица должна быть отсортирована по времени последнего обнаружения от позднего к раннему.

Запрос:

```
SELECT ip, attack_type, involved_attacks, last_detected WHERE ip IN
'192.0.2.0/24' ORDER BY last_detected DESC
```

Название колонки	Атрибут из запроса
IP	`ip` — атрибут, возвращающий IP источника атаки
Тип атаки	`attack_type`
Количество атак	`involved_attacks`
Время	`last_detected` — атрибут, возвращающий метку времени последнего обнаружения

Итоги

Виджет **Итоги** позволяет отобразить данные в виде единого значения.

Для итогов доступно поле **Значение**, позволяющее указать атрибут, значение которого будет отображаться на виджете.

Примеры виджета:

Для событий

Необходимо вывести количество событий за указанный в окне редактирования виджета период времени.

Запрос:

```
SELECT count(id) AS cnt
```

Здесь:

- в поле **Значение** указывается `cnt` — псевдоним атрибута, возвращающего количество событий.

Для активных списков

Необходимо вывести количество уникальных IP, замеченных в DDOS-атаках.

Запрос:

```
SELECT uniq(ip) AS uniq
```

Здесь:

- в поле **Значение** указывается `uniq` — псевдоним атрибута, возвращающего количество уникальных IP-адресов.

Работа с переменными в виджетах

Для виджетов событий и активных списков в секции **Переменные** окна редактирования виджета вы можете задать переменные, которые можно использовать в RQL-запросе.



- Указанные переменные также применяются к дашборду и отображаются на панели переменных дашборда.
- Если заданная переменная уже существует на панели переменных дашборда, то ее значение на панели будет заменено на значение из виджете.

Доступные операции над переменными в виджетах:

- Создание переменной

- Использование переменной в RQL-запросе

Создание переменной

Чтобы добавить переменную:

1. Откройте требуемый дашборд.
2. Откройте окно редактирования виджета одним из способов:
   a. Выберите опцию **Изменить** в выпадающем меню кнопки действий ⋮ виджета.
   b. Нажмите на кнопку **Добавить виджет** (+) в верхней части раздела **Дашборды**.
3. Нажмите на кнопку **Добавить** в секции **Переменные**. Появится строка настройки переменной.
4. Введите аргумент переменной. Аргумент может содержать только латинские буквы (`A–Z`, `a–z`) и цифры (`0–9`).
5. Выберите тип значения переменной. Доступные типы:
   - **Строка**;
   - **Число**;
   - **Дата**;
   - **Логическое**.
6. Укажите значение переменной. Способ задания значения зависит от его типа:
   - **Строка** и **Число** — введите значение вручную.
   - **Дата** — введите значение вручную или задайте с помощью панели выбора даты и времени.
   - **Логическое** — выберите значение из выпадающего списка: **True** или **False**.

Чтобы удалить переменную, нажмите на кнопку 🗑 в строке ее настроек.

Использование переменной в RQL-запросе

Чтобы использовать указанную переменную в RQL-запросе, необходимо использовать шаблон Handlebars:

- Шаблоны для обращения к переменной типа **Строка**, **Число** или **Логическое**:
  - `{{<variable_argument>}}` — обращение к переменной с экранированием специальных символов в ее значении;
  - `{{{<variable_argument>}}}` — обращение к переменной без экранирования специальных символов в ее значении.
- Шаблоны для обращения к переменной типа **Дата**:
  - `{{sqlDateTime <variable_argument>}}` — обращение к значению типа DateTime64;
  - `{{sqlDateTime <variable_argument> false}}` — обращение к значению типа DateTime.

Здесь:

- `<variable_argument>` — аргумент переменной.

Если аргумент переменной содержит специальные символы или `0`, то при обращении к переменной

необходимо экранировать ее аргумент одинарными кавычками, например: `{{'<variable_argument>'}}`.



Пример использования переменных в RQL-запросе

Предположим, что нам необходимо получить события, отфильтрованные по значениям `evo.streams` и `evo.core` поля `moduleKey`.

Чтобы передавать конкретные значения в RQL-запрос, можно создать две переменные:

- Первая переменная:
  - Аргумент: `moduleKey1`.
  - Тип значения: **Строка**.
  - Значение: `evo.streams`.
- Вторая переменная:
  - Аргумент: `moduleKey2`.
  - Тип значения: **Строка**.
  - Значение: `evo.core`.

Тогда для решения поставленной задачи можно использовать следующий запрос:

```
SELECT id, moduleKey, tenantId WHERE moduleKey IN
("{{moduleKey1}}","{{moduleKey2}}") ORDER BY tenantId
```

Обновление виджета

Чтобы отобразить актуальные данные на виджете, требуется его обновить. Вы можете принудительно обновить виджет вручную или настроить автоматическое обновление.

Обновление виджета вручную

Чтобы обновить данные виджета вручную:

1. Откройте требуемый дашборд.
2. Нажмите на кнопку ⟳ в правом верхнем углу выбранного виджета. Данные на виджете будут обновлены.

Автоматическое обновление виджета



Не рекомендуется включать автоматическое обновление данных для каждого виджета дашборда, так как это может излишне нагрузить систему. Подробности приведены в разделе Обновление данных.

Чтобы настроить регулярное автоматическое обновление виджета:

1. Откройте требуемый дашборд.

2. Нажмите на кнопку действий ⋮ в верхнем правом углу виджета и выберите опцию **Изменить**. Отобразится окно редактирования виджета.
3. Установите переключатель **Включить автообновление** в активное положение и укажите периодичность регулярных обновлений:
   a. Задайте частоту обновления вручную или с помощью кнопок — и +.
   b. Выберите из выпадающего списка единицы времени периода.
4. Нажмите на кнопку **Изменить виджет**. Система сохранит измененные данные виджета. Данные на виджете будут автоматически обновляться по истечении указанного периода времени.

Изменение размеров виджета

Чтобы изменить размеры (высоту и ширину) виджета:

1. Откройте требуемый дашборд.
2. Наведите курсор на правый нижний угол выбранного виджета. Курсор примет вид двунаправленной стрелки.
3. Потяните за правый нижний угол виджета:
   - вправо/влево — для изменения ширины;
   - вверх/вниз — для изменения высоты.

   Размеры виджета будут изменены.

Перемещение виджета

Чтобы переместить виджет на дашборде:

1. Откройте требуемый дашборд.
2. Наведите курсор на область названия выбранного виджета. Курсор примет вид четырехнаправленной стрелки.
3. Перетащите виджет в нужное положение на дашборде. Виджет будет перемещен.

Изменение настроек виджета

Чтобы изменить настройки виджета:

1. Откройте требуемый дашборд.
2. Нажмите на кнопку действий ⋮ в верхнем правом углу виджета и выберите опцию **Изменить**. Отобразится окно редактирования виджета.
3. Внесите изменения в требуемые поля.
4. Нажмите на кнопку **Изменить виджет**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится на дашборде.

Настройка отображения легенды виджета

По умолчанию при добавлении виджета в правой его части отображается легенда, на которой приведены цветовые обозначения и названия рядов данных на виджете.

Легенда не отображается:

- для виджетов оповещений с типами диаграмм **Последние** или **Итоги**;
- для виджетов событий и активных списков с типами диаграмм **Карта с группировкой**, **Таблица** или **Итоги**.

Чтобы скрыть легенду, нажмите на кнопку действий ⋮ в верхнем правом углу виджета и выберите опцию **Скрыть легенду**. Легенда перестанет отображаться на виджете, а диаграмма визуализации данных займет всю ширину виджета.

Чтобы отобразить скрытую легенду, нажмите на кнопку действий ⋮ в верхнем правом углу виджета и выберите опцию **Показать легенду**. В правой части виджета отобразится его легенда, а диаграмма визуализации отмасштабируется по оставшемуся на виджете пространству.

Настройка отображения рядов данных виджета

По умолчанию при добавлении виджета отображаются все его ряды данных.

Чтобы скрыть ряд данных на виджете, нажмите на его цветовое обозначение в легенде. Ряд перестанет отображаться на диаграмме визуализации данных, а его цветовое обозначение в легенде станет серым.

Чтобы отобразить скрытый ряд данных на виджете, нажмите на его цветовое обозначение в легенде. Ряд отобразится на диаграмме визуализации данных, а его обозначение в легенде вернет свой изначальный цвет.

Просмотр деталей ряда данных виджета

Просмотр деталей ряда данных доступен для:

- виджетов оповещений;
- виджетов событий;
- виджетов активных списков.

Просмотр ряда данных в виджетах оповещений

Просмотр ряда данных доступен для виджетов оповещений следующих типов:

- **Активные**;
- **Новые**;
- **Последние**;
- **Распределение по времени**;
- **Распределение по атрибуту**.

Чтобы просмотреть детали ряда данных:

- Для виджета типа **Активные**, **Новые**, **Распределение по времени** или **Распределение по атрибуту**: нажмите на название соответствующего ряда в легенде.

- Для виджета типа **Последние**: нажмите на соответствующую строку в таблице.

Система откроет страницу раздела **Оповещения**, на которой отобразится список оповещений, отфильтрованный в соответствии с выбранным рядом данных.

Просмотр ряда данных в виджетах событий и активных списков

Просмотр ряда данных доступен для виджетов событий и виджетов активных списков следующих типов:

- **Столбчатая диаграмма**;
- **Столбчатая диаграмма с накоплением**;
- **График**;
- **Круговая диаграмма**.

При этом на виджетах должна быть задана группировка только по одному атрибуту (например: `GROUP BY collectorId`).

Чтобы просмотреть детали ряда данных, нажмите на название соответствующего ряда в легенде. Система откроет страницу раздела **Поиск**, на которой отобразится список записей хранилища событий или активного списка, отфильтрованный по атрибуту из поля **Группировка** в настройках виджета.



Переход к разделу **Поиск** из легенды недоступен для полей с типом данных `JSON`.

Создание копии виджета

Чтобы создать копию виджета:

1. Откройте требуемый дашборд.
2. Нажмите на кнопку действий ⋮ в верхнем правом углу виджета и выберите опцию **Создать копию**. Система создаст копию виджета и отобразит соответствующее уведомление. Новый виджет появится на дашборде в конце списка.

Удаление виджета

Чтобы удалить виджет:

1. Откройте требуемый дашборд.
2. Нажмите на кнопку действий ⋮ в верхнем правом углу виджета и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления виджета.
3. Нажмите на кнопку **Удалить**. Система удалит виджет и отобразит соответствующее уведомление. Удаленный виджет исчезнет с дашборда.

Экспорт виджета

Чтобы экспортировать виджет:

1. Откройте требуемый дашборд.

2. Нажмите на кнопку ⬆ в правом верхнем углу выбранного виджета. Отобразится окно экспорта виджета.
3. Нажмите на кнопку **Экспортировать**. Система загрузит данные виджета на устройство пользователя в формате CSV и отобразит соответствующее уведомление.

Рекомендации по работе с дашбордами

В данном разделе приведены рекомендации по работе с дашбордами и виджетами в целях оптимизации и повышения работоспособности системы.

Обновление данных

1. Используйте автообновление только для тех виджетов, где это требуется. Это позволит снизить нагрузку на систему, особенно при наличии большого числа пользователей и открытых дашбордов.



Для виджетов без автообновления нужно выполнять обновление данных вручную при просмотре виджета.

2. Старайтесь указывать разумный интервал для автоматического обновления данных виджета, например, от 5 минут.

Оптимизация дашбордов и виджетов

1. Если виджеты используют сложные RQL-запросы или агрегируют большие объемы данных, старайтесь располагать их на разных дашбордах.
2. Не перегружайте дашборд большим количеством одновременно визуализируемых метрик — это может замедлить отображение данных.



Лучше использовать несколько специализированных дашбордов, чем один универсальный.

Запросы к данным

1. Составляйте избирательные запросы:
    - Фильтруйте данные по ключевым полям, чтобы выводить только релевантные данные.
    - Задавайте лимит на количество выводимых результатов, чтобы уменьшить нагрузку на систему.



Также ограничение на количество выводимых результатов можно задать в поле **Лимит** окна редактирования виджета. Если ограничение задано в RQL-запросе и в поле **Лимит**, то применяется наименьшее из двух значений.

    - Не применяйте сортировку, если запрос возвращает большое количество данных.
    - Избегайте запросов полного сканирования, которые требуют чтения и обработки всех данных таблицы.



Задавайте ограничение диапазона дат в секции **Период времени**, чтобы отслеживать только необходимые данные.

▼ *Пример оптимизации запроса*

Необходимо отобразить десять IP-адресов с самым большим количеством ошибочных авторизаций.

Неоптимизированный запрос может выглядеть следующим образом:

```
SELECT source_ip, user_name, event_name, timestamp
WHERE event_name = 'user_login_failed'
ORDER BY timestamp DESC
```

В этом запросе нет ограничения по количеству записей. Система находит все события с ошибочной авторизацией, загружает их в оперативную память и сортирует по времени. Такие операции могут излишнее нагружать систему.

Оптимизированный запрос может выглядеть следующим образом:

```
SELECT source_ip, count(*) AS failed_attempts, count(DISTINCT user_names) AS
unique_users
WHERE event_name = `user_login_failed`
GROUP BY source_ip
ORDER BY failed_attempts DESC
LIMIT 10
```

Здесь:

- С помощью оператора `WHERE` выбираются только те записи, IP-адреса которых не прошли авторизацию.
- С помощью оператора `LIMIT` задается ограничение выводимых результатов в десять строк. Также ограничение можно задать с помощью поля **Лимит** в окне редактирования виджета.
- Сортировка с помощью оператора `ORDER BY` происходит только по ограниченному числу записей, так как был задан лимит.
- С помощью оператора `GROUP BY` производится группировка всех записей, не прошедших авторизацию, по IP-адресам.

Дополнительно можно добавить фильтр по времени в блоке **Период времени** в окне редактирования виджета. Это сузит круг данных и снизит нагрузку на систему.

2. Используйте переменные, чтобы менять контекст для всех виджетов сразу, не редактируя каждый запрос по отдельности.
3. Сохраняйте оптимизированные запросы в шаблоны поиска, чтобы в будущем переиспользовать их.



Вы можете протестировать RQL-запрос в разделе **Поиск** перед добавлением его в виджет.

Работа с инфраструктурой и ролями

1. При возможности увеличьте выделенные ресурсы для подов с базами данных, к которым часто поступают RQL-запросы от виджетов дашбордов.
2. Установите лимиты на потребление оперативной памяти для ролей, чтобы сложные запросы из виджетов персонального дашборда пользователя не повлияли на работу всей системы.

Мониторинг устойчивости

1. Анализируйте загруженность подов и обращайте внимание на нагрузку Clickhouse. Для мониторинга состояния кластера можно создать пользовательские дашборды в Grafana. Подробности о мониторинге за кластером приведены в разделе Мониторинг кластера.
2. Анализируйте виджеты, если они регулярно загружаются дольше остальных при открытии раздела **Дашборды**.

Работа с оповещениями

Данный раздел описывает процесс работы с оповещениями в системе. Работа осуществляется в разделе **Оповещения** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об оповещениях

Оповещения генерируются на основе логики правил сегментации, которые классифицируют и обрабатывают корреляционные события для последующего создания оповещений на их основе. Правила сегментации применяются к потоку событий в рамках сервиса оповещений на конвейере обработки событий, который отвечает за сегментацию корреляционных событий и формирование оповещений в системе.



Чтобы на основе корреляционных событий могли создаваться оповещения, в правиле корреляции необходимо задать поле `trigger_alert` со значением `true`.

Правила сегментации обеспечивают фильтрацию и группировку корреляционных событий, позволяя их преобразование в оповещения в соответствии с заранее определенной логикой. Эти правила ассоциированы с конфигурацией сервиса оповещений и задают параметры для группировки событий, учитывая такие аспекты, как тип события, источник или другие характеристики. Основная цель правил сегментации — управление потоком корреляционных событий для их последующей обработки и генерации оповещений.

После генерации оповещения в него начинают поступать события, которые удовлетворяют условиям в соответствующем правиле сегментации. Так, например, в одном оповещении могут собираться все корреляционные события, которые были созданы одним и тем же правилом

корреляции. При этом события дополнительно могут группироваться внутри оповещения по определенным критериям, таким как тип события, источник или другие атрибуты, на основе логики, заданной в правиле сегментации.

Интерфейс раздела

Панель инструментов раздела **Оповещения** включает в себя следующие компоненты:

- Кнопка позволяет выполнить массовое изменение выбранных оповещений.
- Кнопка позволяет экспортировать данные выбранных оповещений в формате CSV.
- Красный кружок со счетчиком новых оповещений отображает количество оповещений, поступивших в систему и еще не взятых в обработку и имеющих статус **Новое**.
- Зеленый кружок со счетчиком оповещений в работе отображает количество оповещений, находящихся в обработке и имеющих статус **В работе**.
- Кнопка **Закрыть все** позволяет одновременно закрыть все оповещения со статусом **Новое**.
- Поле **Поиск** предназначено для быстрого поиска оповещений в отображаемом списке по названию.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка оповещений.

В рабочей области раздела отображается таблица имеющихся оповещений. Таблица содержит следующие столбцы:

- **Название** — название оповещения.
- **Уровень угрозы** — уровень угрозы оповещения: **Критический**, **Высокий**, **Средний** или **Низкий**.
- **Теги** — теги оповещения.
- **Статус** — текущий статус оповещения: **Ложное срабатывание**, **В работе**, **Новое** или **Закрыто**.
- **Количество событий** — количество корреляционных событий в сегменте группировки для генерации оповещения.
- **Ответственный** — пользователь, ответственный за оповещение.
- **Дата создания** — дата и время поступления оповещения в систему.
- **Дата изменения** — дата и время последнего изменения оповещения.

При работе с таблицей оповещений доступны следующие операции:

- Поиск оповещений.

  ▼ *Особенности поиска оповещений*

  Поиск оповещений осуществляется:

  - По всем полям первого корреляционного события.
  - По следующим полям оповещения:
    - **ID**;
    - **Название**;

        - **Описание**;
        - **ID тенанта**;
        - **ID правила сегментации**;
        - **Название правила корреляции**;
        - **Описание правила корреляции**.

- Фильтрация списка оповещений.

  ▼ *Особенности фильтрации оповещений*

  В параметрах фильтрации доступны следующие опции:

    - **Время создания** — временной диапазон, в котором было сгенерировано оповещение. Начало и конец диапазона можно ввести вручную или задать с помощью панели выбора даты и времени.
    - **Правило сегментации** — правило сегментации, породившее оповещение. Выбирается из выпадающего списка. Доступен выбор нескольких пунктов.
    - **Теги** — пользовательские теги. Вводятся вручную и применяются с помощью клавиши ENTER.
    - **Статус** — текущий статус оповещения. Выбирается из выпадающего списка. Доступен выбор нескольких пунктов.
    - **Уровень угрозы** — уровень угрозы оповещения. Выбирается из выпадающего списка. Доступен выбор нескольких пунктов.
    - **Ответственный** — пользователь, ответственный за оповещение. Выбирается из выпадающего списка. Доступен выбор нескольких пунктов.

  Чтобы отобразить в таблице оповещения, назначенные текущему пользователю, выберите из выпадающего списка вариант **Мои оповещения**.

- Сортировка оповещений по полям **Название**, **Количество событий**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного оповещения в правой части рабочей области отображается его карточка с основной информацией.

Работа с оповещением

Доступные операции над оповещениями:

- Просмотр основной информации об оповещении
- Просмотр детальной информации об оповещении
- Работа с комментариями
- Изменение оповещения

- Массовое закрытие оповещений
- Экспорт оповещения

Просмотр основной информации об оповещении

Чтобы просмотреть основную информацию об оповещении:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на строку оповещения в списке. Система отобразит в правой части экрана карточку этого оповещения с основной информацией о нем.

В верхней части карточки отображается название оповещения. Справа от названия расположена кнопка  , при нажатии на которую осуществляется экспорт выбранного оповещения в формате CSV.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит сведения, сгруппированные по следующим секциям:

- **Информация об оповещении** — основные сведения об оповещении, содержащие следующие параметры:
  - **ID** — уникальный идентификационный код оповещения, генерируемый автоматически при его поступлении в систему.
  - **Название** — название оповещения.



Название оповещения формируется на основе логики, определенной в поле `alert_name` правила сегментации. Если поле `alert_name` не определено, оповещению будет автоматически присвоено название правила корреляции, которое первым привело к его срабатыванию. Если правило корреляции не ассоциировано с правилом сегментации, название оповещения будет автоматически сформировано из названия правила корреляции.

  - **Описание** — описание оповещения.
  - **ID тенанта** — уникальный идентификационный код тенанта, к которому относится оповещение.
  - **Название правила сегментации** — название правила сегментации, сгенерировавшего оповещение.
  - **ID правила сегментации** — уникальный идентификационный код правила сегментации, сгенерировавшего оповещение.
  - **Поля группировки** — поля, по которым производилась группировка корреляционных событий в рамках процесса сегментации.
  - **Количество событий для группировки** — количество корреляционных событий в сегменте группировки для генерации оповещения.
  - **Ограничение группировки по времени** — временной интервал в секундах, в рамках которого корреляционные события группируются в оповещение.

- **Название правила корреляции** — название правила корреляции, породившего оповещение.
- **Описание правила корреляции** — описание правила корреляции, породившего оповещение.
- **Уровень угрозы** — уровень угрозы оповещения: **Критический**, **Высокий**, **Средний** или **Низкий**.
- **Теги** — теги оповещения.
- **Ответственный** — пользователь, ответственный за оповещение.
- **Статус** — текущий статус оповещения: **Ложное срабатывание**, **В работе**, **Новое** или **Закрыто**.
- **Количество событий** — количество корреляционных событий в сегменте группировки для генерации оповещения.
- **Дата создания** — дата и время поступления оповещения в систему.
- **Дата обновления** — дата и время последнего обновления оповещения.

- **Первое корреляционное событие** — поля первого корреляционного события, на основе которого сгенерировано оповещение.
- **Базовые события** — список базовых (сырых) событий, лежащих в основе первого корреляционного события. Список представлен в виде таблицы со следующими столбцами:
  - **ID** — уникальный идентификационный код базового события.
  - **Timestamp** — дата и время поступления базового события в систему.

  Вы можете сворачивать и разворачивать список базовых событий с помощью стрелки справа от заголовка списка.

  В строке каждого базового события в списке расположена кнопка поиска, при нажатии на которую выполняется поиск соответствующего события в разделе **Поиск**. Чтобы найти все базовые события, которые лежат в основе первого корреляционного события, нажмите на кнопку **Найти все** () справа от заголовка списка.

В нижней части карточки оповещения находится кнопка **Подробнее**. При нажатии на кнопку в рабочей области отображается детальная информация об оповещении, которая включает в себя следующие вкладки:

- **Информация** — отображает основную информацию об оповещении.
- **Расследование** — отображает информацию о событиях и группах из правила корреляции, связанных с оповещением.
- **Интеграция** — отображает информацию о связанных с оповещением интеграциях.

Просмотр детальной информации об оповещении

Чтобы просмотреть детальную информацию об оповещении:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на строку оповещения в списке. Система отобразит в правой части экрана карточку этого оповещения с основной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. В рабочей области отобразится детальная информация об оповещении.

В верхней части экрана детальной информации об оповещении отображается название оповещения, а также содержатся следующие компоненты:

- Кнопка ⬆ позволяет экспортировать выбранное оповещение в формате CSV.
- Переключатель **Комментарии**, позволяющий отобразить в правой части рабочей области панель комментариев, которая содержит список всех комментариев к оповещению в хронологическом порядке.
- Кнопка **Назад**, позволяющая вернуться к карточке выбранного оповещения.

Экран детальной информации об оповещении содержит следующие вкладки:

- **Информация** — отображает основную информацию об оповещении.
- **Расследование** — отображает информацию о событиях и группах из правила корреляции, связанных с оповещением.
- **Интеграция** — отображает информацию о связанных с оповещением интеграциях.

Вкладка Информация

Вкладка **Информация** отображает основные сведения об оповещении, сгруппированные по следующим секциям:

- **Информация об оповещении** — основные сведения об оповещении, содержащие следующие параметры:
  - **ID** — уникальный идентификационный код оповещения, генерируемый автоматически при его поступлении в систему.
  - **Название** — название оповещения.

> Название оповещения формируется на основе логики, определенной в поле `alert_name` правила сегментации. Если поле `alert_name` не определено, оповещению будет автоматически присвоено название правила корреляции, которое первым привело к его срабатыванию. Если правило корреляции не ассоциировано с правилом сегментации, название оповещения будет автоматически сформировано из названия правила корреляции.

  - **Описание** — описание оповещения.
  - **ID тенанта** — уникальный идентификационный код тенанта, к которому относится оповещение.
  - **Название правила сегментации** — название правила сегментации, сгенерировавшего оповещение.
  - **ID правила сегментации** — уникальный идентификационный код правила сегментации, сгенерировавшего оповещение.
  - **Поля группировки** — поля, по которым производилась группировка корреляционных событий в рамках процесса сегментации.
  - **Количество событий для группировки** — количество корреляционных событий в сегменте группировки для генерации оповещения.

- **Ограничение группировки по времени** — временной интервал в секундах, в рамках которого корреляционные события группируются в оповещение.
- **Название правила корреляции** — название правила корреляции, породившего оповещение.
- **Описание правила корреляции** — описание правила корреляции, породившего оповещение.
- **Уровень угрозы** — уровень угрозы оповещения: **Критический**, **Высокий**, **Средний** или **Низкий**.
- **Теги** — теги оповещения.
- **Ответственный** — пользователь, ответственный за оповещение.
- **Статус** — текущий статус оповещения: **Ложное срабатывание**, **В работе**, **Новое** или **Закрыто**.
- **Количество событий** — количество корреляционных событий в сегменте группировки для генерации оповещения.
- **Дата создания** — дата и время поступления оповещения в систему.
- **Дата обновления** — дата и время последнего обновления оповещения.

- **Первое корреляционное событие** — поля первого корреляционного события, на основе которого сгенерировано оповещение.
- **Базовые события** — список базовых (сырых) событий, лежащих в основе первого корреляционного события. Список представлен в виде таблицы со следующими столбцами:
  - **ID** — уникальный идентификационный код базового события.
  - **Timestamp** — дата и время поступления базового события в систему.



Вы можете сворачивать и разворачивать список базовых событий с помощью стрелки справа от заголовка списка.

В строке каждого базового события в списке расположена кнопка поиска  , при нажатии на которую выполняется поиск соответствующего события в разделе **Поиск**. Чтобы найти все базовые события, которые лежат в основе первого корреляционного события, нажмите на кнопку **Найти все** () справа от заголовка списка.

Вкладка Расследование

Вкладка **Расследование** отображает сведения о событиях и группах из правила корреляции, связанных с оповещением.

В верхней части вкладки отображается таблица **Связанные события**, которая содержит события корреляции, связанные с оповещением. Таблица состоит из следующих столбцов:

- **Время** — дата и время возникновения события.
- **Информация о событии** — базовое (сырое) событие, лежащее в основе просматриваемого события.



В таблице **Связанные события** отображаются только первые 1000 корреляционных событий, связанных с оповещением.



Для таблицы событий доступна сортировка по полю **Время**.

Над таблицей находится кнопка **Все события корреляции**, при нажатии на которую выполняется поиск всех событий корреляции из таблицы **Связанные события** в разделе **Поиск**.

Слева от строки каждого события корреляции в таблице расположена кнопка в виде стрелки, позволяющая раскрыть список базовых событий этого события.



На вкладке также можно раскрыть списки базовых событий сразу всех событий корреляции. Для этого необходимо использовать кнопку в виде стрелки слева от области заголовка таблицы.

Чтобы посмотреть детальную информацию по событию, нажмите на его строку в списке. В правой части рабочей области отобразится карточка выбранного события. Карточка содержит следующие сведения о событии:

- Дата и время получения события.
- Служебные поля модели события.
- Регулярные поля модели события.
- Для события корреляции: список базовых событий оповещения. Список отображается в виде таблицы со следующими столбцами:
  - **ID** — уникальный идентификационный код базового события.
  - **Timestamp** — дата и время поступления базового события в систему.

  Вы можете сворачивать и разворачивать список базовых событий с помощью стрелки справа от заголовка списка.

  В строке каждого базового события в списке расположена кнопка поиска, при нажатии на которую выполняется поиск соответствующего события в разделе **Поиск**. Чтобы найти все базовые события, которые лежат в основе первого корреляционного события, нажмите на кнопку **Найти все** () справа от заголовка списка.
- Для базового события: подробная информация об исходном событии в виде VRL-запроса.

  Справа от заголовка блока с информацией об исходном событии расположена кнопка **Скопировать** (), позволяющая скопировать исходное событие в буфер обмена.



Вы можете сворачивать и разворачивать блоки сведений о событии с помощью стрелки справа от заголовков блоков.

В нижней части вкладки в рабочей области отображается **список групп событий**, связанных с оповещением. Список представлен в виде таблицы, содержащей столбцы с полями, по которым проводится группировка в правиле корреляции.

В строке каждой группы в списке расположена кнопка **Перейти**, при нажатии на которую выполняется поиск событий в разделе **Поиск** со значениями полей, соответствующими выбранной группировке.

Вкладка Интеграции

Вкладка **Интеграции** отображает сведения об интеграциях, связанных с оповещением.

Список интеграций представлен в виде таблицы, содержащей столбцы с названием и текущим статусом связанной интеграции.

В строке каждой интеграции в списке расположена кнопка **Отправить**, при нажатии на которую осуществляется отправка оповещения во внешнюю систему согласно настройкам интеграции.

Работа с комментариями

Доступные операции над комментариями:

- Просмотр комментариев
- Добавление комментариев
- Изменение комментариев
- Удаление комментариев

Просмотр комментариев

Чтобы просмотреть комментарии к оповещению:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на строку оповещения в списке. Система отобразит в правой части экрана карточку этого оповещения с основной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. В рабочей области отобразится детальная информация об оповещении.
4. Установите переключатель **Комментарии** в левой верхней части экрана в активное положение. Система отобразит в правой части экрана панель комментариев к оповещению.

На панели комментариев отображены все комментарии, добавленные к оповещению.

В верхней части каждого комментария отображаются:

- идентификатор пользователя, добавившего комментарий;
- дата и время добавления комментария;
- кнопки для изменения и удаления комментария.

Добавление комментариев

Чтобы добавить комментарий к оповещению:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на строку оповещения в списке. Система отобразит в правой части экрана карточку этого оповещения с основной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. В рабочей области отобразится детальная информация об оповещении.

4. Установите переключатель **Комментарии** в левой верхней части экрана в активное положение. Система отобразит в правой части экрана панель комментариев к оповещению.
5. Нажмите на кнопку **Добавить комментарий**. Система отобразит поле ввода нового комментария.
6. Введите комментарий.
7. Нажмите на кнопку **Сохранить**. Система добавит новый комментарий к оповещению и отобразит соответствующее уведомление. Новый комментарий появится в списке комментариев на панели.

Изменение комментариев

Чтобы изменить комментарий к оповещению:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на строку оповещения в списке. Система отобразит в правой части экрана карточку этого оповещения с основной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. В рабочей области отобразится детальная информация об оповещении.
4. Установите переключатель **Комментарии** в левой верхней части экрана в активное положение. Система отобразит в правой части экрана панель комментариев к оповещению.
5. Нажмите на кнопку  в правом верхнем углу комментария, который требуется изменить. Система отобразит поле ввода комментария.
6. Введите новый комментарий.
7. Нажмите **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится на панели комментариев.

Удаление комментариев

Чтобы удалить комментарий к оповещению:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на строку оповещения в списке. Система отобразит в правой части экрана карточку этого оповещения с основной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. В рабочей области отобразится детальная информация об оповещении.
4. Установите переключатель **Комментарии** в левой верхней части экрана в активное положение. Система отобразит в правой части экрана панель комментариев к оповещению.
5. Нажмите на кнопку  в правом верхнем углу комментария, который требуется удалить. Отобразится окно подтверждения удаления комментария.
6. Нажмите на кнопку **Удалить**. Система удалит комментарий и отобразит соответствующее уведомление. Удаленный комментарий будет исключен из списка.

Изменение оповещения

Вы можете изменить оповещение следующими способами:

- через изменение полей в карточке оповещения;
- через массовое изменение оповещений.



- Изменение полей с помощью кнопки редактирования также доступно на вкладке **Информация** на экране детальной информации об оповещении.
- Массовое изменение доступно только для полей **Уровень угрозы**, **Статус** и **Ответственный**.
- Если определенный параметр не требуется изменять, оставьте его поле незаполненным.

Особенности изменения полей оповещения

Способ изменения значения зависит от поля оповещения:

- **Описание** — введите новый текст описания.
- **Уровень угрозы** — выберите из выпадающего списка новый уровень угрозы оповещения: **Критический**, **Высокий**, **Средний** или **Низкий**.
- **Теги** — задайте новый список тегов.
- **Ответственный** — выберите из выпадающего списка пользователя, ответственного за оповещение.



Ответственный после назначения не может быть удален. При необходимости можно выбрать другого пользователя, но нельзя убрать ответственного за оповещение, если он уже был назначен.

- **Статус** — выберите из выпадающего списка новый статус оповещения: **Ложное срабатывание**, **В работе** или **Закрыто**.



Статус **Новое** присваивается оповещению при его создании и не может быть назначен вручную.

Массовое закрытие оповещений

Массовое закрытие позволяет одновременно перевести все оповещения со статусом **Новое** в один из следующих статусов:

- **Закрыто**;
- **Ложное срабатывание**.

Чтобы воспользоваться массовым закрытием оповещений:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на кнопку **Закрыть все** над списком оповещений. Отобразится окно закрытия оповещений.
3. Выберите итоговый статус закрываемых оповещений из выпадающего списка.
4. Нажмите на кнопку **Закрыть**. Система переведет все новые оповещения в выбранный статус и

отобразит соответствующее уведомление.

Экспорт оповещения

Экспорт оповещений происходит в формате CSV со следующими полями:

- ID оповещения;
- все поля сущности оповещения;
- ссылка на оповещение;
- ID корреляционных событий.

Чтобы экспортировать отдельное оповещение:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на строку оповещения в списке. Система отобразит в правой части экрана карточку этого оповещения с основной информацией о нем.
3. Нажмите на кнопку ↑ в верхней части карточки. Отобразится окно экспорта оповещения.
4. Нажмите **Экспортировать**. Система загрузит информацию об оповещении на устройство пользователя и отобразит соответствующее уведомление.



Экспортировать оповещение можно также из экрана его детальной информации. Для этого:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Нажмите на строку оповещения в списке. Система отобразит в правой части экрана карточку этого оповещения с основной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. В рабочей области отобразится детальная информация об оповещении.
4. Нажмите на кнопку ↑ на панели инструментов. Отобразится окно экспорта оповещения.
5. Нажмите **Экспортировать**. Система загрузит информацию об оповещении на устройство пользователя и отобразит соответствующее уведомление.

Вы также можете экспортировать сразу несколько оповещений. Для этого:

1. Перейдите в раздел **Оповещения**. Система отобразит сведения об имеющихся оповещениях.
2. Установите флажки напротив тех оповещений, которые вы хотите экспортировать, в левом столбце таблицы.
3. Нажмите на кнопку ↑ на панели инструментов. Отобразится окно экспорта выбранных оповещений.
4. Нажмите **Экспортировать**. Система загрузит информацию об оповещениях на устройство пользователя и отобразит соответствующее уведомление.



При экспорте списка оповещений выгружаются только те оповещения, которые

отображаются в списке после применения фильтров и поиска.

Работа с отчетами

Данный раздел описывает процесс работы с отчетами и шаблонами в системе. Работа осуществляется в разделе **Отчеты** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об отчетах

Отчеты в системе создаются для сбора, анализа и представления данных, собранных системой.

Пользователи могут создавать, просматривать и управлять отчетами, что включает в себя настройку параметров отчета, его создание на основе предопределенных шаблонов, отправку во внешние системы, а также экспорт в файл. Отчеты могут быть настроены для автоматической публикации по расписанию.

Интерфейс раздела

Интерфейс раздела зависит от открытой вкладки. Доступные вкладки:

- **Сформированные** — вкладка для работы со сформированными отчетами.
- **Шаблоны** — вкладка для работы с шаблонами отчетов.

Интерфейс вкладки Сформированные

Панель инструментов вкладки содержит следующие элементы:

- Поле **Поиск** предназначено для быстрого поиска отчетов по их названию.
- Кнопка **Фильтр** () позволяет открыть в правой части рабочей области панель для настройки фильтрации списка отчетов.

В рабочей области вкладки отображается таблица имеющихся отчетов. Таблица содержит следующие столбцы:

- **Название** — имя отчета, используемое для его идентификации в системе.
- **Теги** — теги, ассоциированные с отчетом.
- **Дата публикации** — дата и время публикации отчета в системе.

При работе с таблицей отчетов доступны следующие операции:

- Поиск отчетов по полю **Название**.
- Фильтрация отчетов по полю **Теги**. Теги вводятся вручную и применяются по клавише ENTER.
- Сортировка отчетов по полям **Название** и **Дата публикации**.

- Настройка отображения таблицы.

При выборе конкретного отчета в правой части рабочей области отображается его карточка с детальной информацией.

Интерфейс вкладки Шаблоны

Панель инструментов вкладки содержит следующие элементы:

- Кнопка **Создать** (+) позволяет создать новый шаблон отчета.
- Кнопка **Создать из схемы** (+) позволяет создать новый шаблон отчета на основе схемы.
- Поле **Поиск** предназначено для быстрого поиска шаблонов по их названию.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка шаблонов.

В рабочей области вкладки отображается таблица имеющихся шаблонов отчетов. Таблица содержит следующие столбцы:

- **Название** — имя шаблона отчета, используемое для его идентификации в системе.
- **Расписание** — индикатор автоматического создания отчетов на основе этого шаблона по расписанию: **Включено** или **Выключено**.
- **Теги** — список связанных с шаблоном тегов, используемых для его поиска и классификации.
- **Дата изменения** — дата и время последнего изменения шаблона отчета.
- **Дата последнего отчета** — дата и время последней публикации отчета на основе текущего шаблона.

При работе с таблицей шаблонов отчетов доступны следующие операции:

- Поиск шаблонов по полю **Название**.
- Фильтрация шаблонов по полям **Теги** и **Расписание**:
  - Теги вводятся вручную и применяются по клавише ENTER.
  - Индикатор расписания выбирается из выпадающего списка.
- Сортировка шаблонов по полям **Название** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного шаблона отчета в правой части рабочей области отображается его карточка с детальной информацией.

Работа с шаблоном отчета

Шаблоны отчетов являются инструментом для определения структуры и периодичности публикации отчетов. Они позволяют выбирать аналитические данные для включения в отчеты и поддерживают функции автоматического создания и отправки отчетов по расписанию.

Доступные операции над шаблоном отчета:

- Создание шаблона отчета

- Просмотр карточки шаблона отчета
- Изменение шаблона
- Создание копии
- Удаление шаблона
- Сохранение шаблона отчета как схемы

Создание шаблона отчета

Создать шаблон отчета можно следующими способами:

- Создать шаблон, добавляя виджеты вручную, или на основе существующего дашборда.
- Использовать предварительно созданную схему.

Создание шаблона отчета вручную

Чтобы создать шаблон отчета вручную:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит список имеющихся шаблонов отчетов.
2. Нажмите на кнопку **Создать** (+) на панели инструментов. Система отобразит структуру шаблона в рабочей области и боковую панель с его текущими настройками.
3. Введите название шаблона отчета.
4. При необходимости введите описание шаблона отчета.
5. При необходимости добавьте теги для классификации и поиска шаблона в системе.
6. При необходимости выберите интеграцию для отправки отчета во внешние системы. Для этого выберите из выпадающего списка **Отправить** одну из доступных почтовых интеграций.
7. При необходимости выберите из выпадающего списка **Шаблон дашборда** дашборд, который определит структуру текущего шаблона отчета.

   Пользователю доступны общие дашборды и созданные им персональные дашборды.

8. Добавьте отдельные виджеты в шаблон с помощью кнопки **Добавить виджет**. Если вы выбрали дашборд как основу для шаблона, вы можете удалить ненужные виджеты.

   Шаблон отчета должен содержать хотя бы один виджет.

9. Настройте виджеты, как описано в разделе Работа с виджетами. Период, за который собраны данные в отчете, определяется настройками виджетов, добавленных в него.

   Изменения настроек не повлияют на исходный дашборд.

   При экспорте отчета система может изменить его внешний вид для лучшего отображения данных.

10. При необходимости задайте аргументы и значения переменных, которые требуется

использовать в RQL-запросах виджетов, в секции **Переменные**.

Чтобы добавить переменную:

a. Нажмите на кнопку **Добавить**. Появится строка настройки переменной.

b. Введите аргумент переменной. Аргумент может содержать только латинские буквы (`A–Z`, `a–z`) и цифры (`0–9`).

c. Выберите тип значения переменной. Доступные типы:

- **Строка**;
- **Число**;
- **Дата**;
- **Логическое**.

d. Укажите значение переменной. Способ задания значения зависит от его типа:

- **Строка** и **Число** — введите значение вручную.
- **Дата** — введите значение вручную или задайте с помощью панели выбора даты и времени.
- **Логическое** — выберите значение из выпадающего списка: **True** или **False**.

Чтобы удалить переменную, нажмите на кнопку в строке ее настроек.

Чтобы применить указанные переменные и перестроить виджеты с учетом их значений, нажмите на кнопку **Применить**.

11. Задайте период хранения. Выберите единицы измерения и их количество, чтобы определить срок, по истечении которого отчет будет удален из системы.
12. Задайте параметры публикации.

    Для публикации отчета по расписанию установите переключатель **Расписание** в активное положение и задайте настройки публикации:

    - **Период** — выберите из выпадающего списка период публикации:
      - **Каждый день** — для публикации отчетов ежедневно.
      - **По дням** — для публикации отчетов в конкретные дни недели. При выборе этого периода необходимо указать дни недели для публикации.
    - **Время** — укажите время, в которое должен быть сформирован отчет.
13. Нажмите **Сохранить**. Система создаст шаблон отчета и отобразит соответствующее уведомление. Новый шаблон появится в разделе **Отчеты** на вкладке **Шаблоны**.

Чтобы закрыть окно создания шаблона и вернуться на вкладку **Шаблоны**, нажмите на кнопку **Назад** в левой верхней части экрана.

Создание шаблона отчета из схемы

Чтобы создать шаблон отчета из схемы:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит список имеющихся шаблонов отчетов.
2. Нажмите на кнопку **Создать из схемы** в левой части панели инструментов. Откроется окно создания шаблона.
3. Выберите схему из выпадающего списка.
4. Сопоставьте переменные виджетов с источниками данных — хранилищем событий или активным списком в зависимости от типа виджетов.
5. Нажмите на кнопку **Продолжить**. Система отобразит структуру шаблона в рабочей области и боковую панель с его текущими настройками.
6. При необходимости внесите изменения в настройки, как при создании шаблона вручную.



Если при настройке вы выберете дашборд в выпадающем списке **Шаблон дашборда**, виджеты текущей схемы будут заменены на виджеты этого дашборда.

7. Нажмите **Сохранить**. Система создаст шаблон отчета и отобразит соответствующее уведомление. Новый шаблон появится в разделе **Отчеты** на вкладке **Шаблоны**.

Чтобы закрыть окно создания шаблона и вернуться на вкладку **Шаблоны**, нажмите на кнопку **Назад** в левой верхней части экрана.

Просмотр карточки шаблона отчета

Чтобы просмотреть карточку шаблона отчета:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.

В верхней части карточки отображается название шаблона отчета. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранного шаблона.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит сведения о шаблоне отчета, сгруппированные по следующим секциям:

- **Информация** — основная информация о шаблоне отчета:
  - **ID** — уникальный идентификатор шаблона отчета.
  - **Название** — имя шаблона отчета.
  - **Описание** — дополнительная информация о шаблоне.
  - **ID тенанта** — уникальный идентификационный код тенанта, к которому относится шаблон.
  - **Теги** — список связанных с шаблоном тегов, используемых для его поиска и классификации.
  - **Расписание** — индикатор автоматического создания отчетов на основе этого шаблона по

расписанию: **Включено** или **Выключено**.

- **Дата последнего отчета** — дата и время создания последнего отчета по текущему шаблону. Поле отображается, если отчет создавался хотя бы один раз.
- **Дата следующего отчета** — дата и время создания следующего отчета по настроенному расписанию. Поле отображается, если настроена публикация отчета по расписанию.
- **Дата создания** — дата и время создания шаблона отчета.
- **Создал** — пользователь, инициировавший создание шаблона отчета.
- **Дата изменения** — дата и время последнего изменения шаблона отчета.
- **Изменил** — пользователь, выполнивший последнее изменение шаблона отчета.

- **Переменные** — список аргументов и значений переменных, заданных в шаблоне. Секция отображается, если задана хотя бы одна переменная.

В нижней части карточки расположена кнопка **Сформировать**, позволяющая создать версию отчета по выбранному шаблону на текущий момент.

Изменение шаблона

Вы можете изменить шаблон отчета следующими способами:

- через меню действий в карточке шаблона отчета;
- через изменение полей в карточке шаблона отчета.



Чтобы закрыть окно изменения шаблона и вернуться к его карточке, нажмите на кнопку **Назад** в левой верхней части экрана.

Создание копии

Чтобы создать копию шаблона:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Создать копию**. Система отобразит окно настроек шаблона.
4. Внесите требуемые изменения в настройки шаблона.
5. Нажмите на кнопку **Создать**. Система создаст шаблон отчета и отобразит соответствующее уведомление. Новый шаблон появится в разделе **Отчеты** на вкладке **Шаблоны**.

Удаление шаблона

Вы можете удалить шаблон отчета через меню действий в его карточке.

Сохранение шаблона отчета как схемы

Чтобы сохранить шаблон отчета как схему:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Создать схему**. Отобразится окно создания схемы.
4. По умолчанию название и описание создаваемой схемы идентичны названию и описанию шаблона. При необходимости измените название и описание схемы.
5. Выберите из выпадающего списка каталог экспертизы, в котором будет создана схема.
6. Для виджетов событий и активных списков автоматически генерируются переменные, которые при создании шаблона отчета из схемы нужно связать с источниками данных. При необходимости измените переменные виджетов.



Чтобы при создании шаблона отчета несколько виджетов получали данные из одного источника, задайте для них одинаковые переменные.

7. Нажмите на кнопку **Далее**. Откроется окно создания RObject-конфигурации схемы с заполненными данными.
8. Задайте идентификатор схемы шаблона отчета в поле `id`.
9. При необходимости внесите изменения в поля схемы.
10. Нажмите на кнопку **Опубликовать версию**. Система создаст схему шаблона отчета и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.



Чтобы схема стала доступной для создания шаблонов отчета, ее необходимо включить.

Вы можете создать черновик схемы шаблона без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Работа со сформированным отчетом

Все отчеты в системе формируются на основе шаблонов отчетов.

Доступные операции над отчетом:

- Создание отчета
- Просмотр карточки отчета
- Просмотр содержимого отчета
- Экспорт отчета
- Удаление отчета

Создание отчета

Создание отчета начинается с выбора и настройки шаблона. Вы можете добавить различные

виджеты и задать параметры отчета. Отчет можно создать вручную или настроить расписание, чтобы отчеты создавались автоматически.

Чтобы создать отчет из существующего шаблона:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.
3. Нажмите **Сформировать** в нижней части карточки. Система создаст по шаблону версию отчета на текущий момент и отобразит соответствующее уведомление. Новый отчет появится в разделе **Отчеты** на вкладке **Сформированные**.

Просмотр карточки отчета

Чтобы просмотреть карточку отчета:

1. Перейдите на вкладку **Сформированные** раздела **Отчеты**. Система отобразит сведения об имеющихся отчетах.
2. Нажмите на строку отчета в списке. Система отобразит в правой части экрана карточку этого отчета с базовой информацией о нем.

В верхней части карточки отображается название отчета. Справа от названия расположена кнопка для удаления отчета.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит сведения об отчете, сгруппированные по следующим секциям:

- **Информация** — основная информация об отчете:
  - **ID** — уникальный идентификатор отчета.
  - **Название** — имя отчета, используемое для идентификации в системе.
  - **ID тенанта** — уникальный идентификационный код тенанта, к которому относится отчет.
  - **Теги** — список тегов, связанных с отчетом.
  - **Шаблон** — шаблон, по которому был создан отчет.
  - **Дата публикации** — дата и время публикации отчета.
  - **Срок хранения** — срок хранения отчета, по истечении которого он будет удален из системы.
  - **Сформировал** — имя пользователя, сформировавшего отчет.
  - **Запуск** — каким образом был сформирован отчет: **Автоматически** или **Вручную**.
- **Переменные** — список аргументов и значений переменных, заданных в шаблоне. Секция отображается, если задана хотя бы одна переменная.

В нижней части карточки расположены кнопки:

- **Смотреть отчет** — позволяет просмотреть содержимое отчета.

- **Экспорт в файл** — позволяет экспортировать отчет.

Просмотр содержимого отчета

Чтобы просмотреть содержимое отчета в формате HTML:

1. Перейдите на вкладку **Сформированные** раздела **Отчеты**. Система отобразит сведения об имеющихся отчетах.
2. Нажмите на строку отчета в списке. Система отобразит в правой части экрана карточку этого отчета с базовой информацией о нем.
3. Нажмите на кнопку **Смотреть отчет** в нижней части карточки шаблона. Система отобразит содержимое отчета на отдельной вкладке браузера в формате HTML.

Чтобы закрыть отчет и вернуться к списку отчетов, нажмите на кнопку **Назад** в левой верхней части экрана.

Экспорт отчета

Чтобы выгрузить отчет:

1. Перейдите на вкладку **Сформированные** раздела **Отчеты**. Система отобразит сведения об имеющихся отчетах.
2. Нажмите на строку отчета в списке. Система отобразит в правой части экрана карточку этого отчета с базовой информацией о нем.
3. Нажмите на кнопку **Экспорт в файл** в нижней части карточки. Система отобразит окно экспорта.
4. Выберите формат экспорта из выпадающего списка:
   - PDF;
   - CSV.

   Для формата CSV отчет выгружается в виде ZIP-архива.
5. Нажмите на кнопку **Экспорт**. Система выгрузит отчет на устройство пользователя в выбранном формате и отобразит соответствующее уведомление.

При экспорте отчетов их внешний вид может измениться для удобства отображения данных:

- Все временные метки преобразуются в формат UTC.
- Шаги делений на виджетах могут измениться в зависимости от разрешения экрана для лучшего отображения данных.
- В файлах PDF таблицы могут содержать до восьми столбцов. Если столбцов больше четырех, лист с таблицей разворачивается горизонтально.

Удаление отчета

Вы можете удалить отчет через кнопку в его карточке.

Работа с разделом Поиск

Раздел **Поиск** позволяет просматривать и анализировать события и записи активных списков. В этом разделе отображаются события из указанных хранилищ или записи указанного активного списка, с возможностью поиска и фильтрации вхождений с помощью пользовательских фильтров и запросов на языке RQL.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.



Если вы обновляли систему с версии ниже 2.6.0 и поиск по нескольким хранилищам не работает, пересоздайте таблицы хранилищ событий согласно инструкции **Включение поиска по нескольким хранилищам событий** в списке особенностей обновления с версии ниже 2.6.0.

О событиях

Событие — это идентифицированный переход системы, сервиса или сети в состояние, которое указывает на возможное нарушение политики информационной безопасности, сбой средств контроля или иную ситуацию, которая может быть значимой для информационной безопасности.

Система позволяет:

- задавать модели событий для сбора данных из нескольких источников;
- собирать события в соответствии с заданными моделями;
- детектировать действия в системе в виде событий аудита;
- обогащать события за счет получения дополнительных данных и установления связей с другими событиями;
- отслеживать и визуализировать события;
- управлять хранением и обработкой событий;
- создавать фиды событий для отправки в R-Vision TIP и иные системы класса Threat Intelligence.

Режимы поиска

В системе доступны два режима поиска:

- **Базовый поиск** — поиск без использования проекций и группировки результатов.
- **Продвинутый поиск** — поиск с использованием проекций и группировки результатов с помощью операторов языка RQL `SELECT` и `GROUP BY` соответственно.

  Пример запроса в режиме продвинутого поиска

  ```
  SELECT count(id) as cnt, collectorId GROUP BY collectorId
  ```

  При выполнении вышеприведенного запроса данные группируются по полю `collectorId` и

вычисляется `cnt` — количество записей в каждой группе. Результат запроса — таблица с двумя столбцами: `cnt` и `collectorId`.

В режиме продвинутого поиска структура результата не определяется моделью события, из-за чего некоторые функции недоступны, а именно:

- график распределения результатов поиска по времени;
- статистика по значениям полей;
- контекстное меню для работы со свойствами в ячейках таблицы;
- настройка списка и порядка полей, отображаемых в таблице результатов поиска;
- группировка полей в карточке события или записи активного списка;
- шаблоны настроек таблицы результатов поиска.

По умолчанию при открытии раздела **Поиск** используется режим базового поиска. Режим поиска для текущего запроса определяется автоматически при нажатии кнопки **Поиск** и отправке запроса:

- Запрос выполняется в режиме продвинутого поиска, если он содержит операторы `GROUP BY`, `RIGHT JOIN`, `FULL JOIN` или `SELECT`, за исключением `SELECT *`.



RQL поддерживает недетерминированные функции — функции, которые могут возвращать разные результаты при каждом вызове, в том числе при одинаковых входных данных. Примеры недетерминированных функций:

- `now()` — текущая дата и время;
- агрегатные функции, такие как `first_value()`, `last_value()` и `anyHeavy()`.

Не используйте эти функции при поиске по нескольким хранилищам, поскольку в этом случае результат их выполнения не определен.

- В остальных случаях запрос выполняется в режиме базового поиска.

Интерфейс раздела

В верхней части раздела отображается:

- если выбрано одно хранилище событий — название текущего хранилища;
- если выбрано несколько хранилищ событий — количество выбранных хранилищ;
- если выбран активный список — название активного списка.

Чтобы выбрать хранилища или активный список, по которым осуществляется поиск, используйте выпадающий список **Источник событий** в верхней части страницы.



Для быстрого поиска хранилища или активного списка в выпадающем списке начните вводить его название.

Вы можете отобразить выбранные хранилища в начале списка с помощью кнопки .

Вы можете осуществлять одновременный поиск по хранилищам из всех баз данных событий, которые зарегистрированы в разделе **Ресурсы → Базы данных событий** веб-интерфейса системы.

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка позволяет экспортировать найденные события или записи.
- Кнопка открывает страницу сравнения событий. Кнопка отображается, если для поиска выбрано одно или несколько хранилищ событий.
- Кнопка открывает панель избранных событий. Кнопка отображается, если для поиска выбрано одно или несколько хранилищ событий.
- Кнопка открывает панель для работы с поисковыми шаблонами.
- Кнопка позволяет скопировать в буфер обмена текущий поисковый запрос, объединенный с условиями фильтра оператором `AND`. Скопированный запрос можно использовать при создании виджета.
- Кнопка позволяет создать виджет на основе заданного поискового запроса. Кнопка отображается, только если для поиска выбрано одно хранилище событий или активный список и найдена хотя бы одна запись.
- Поле запроса предназначено для ввода ключевых слов или RQL-запроса для поиска данных в выбранном хранилище или активном списке.
- Кнопка позволяет выбрать способ задания периода, в рамках которого требуется искать данные.
- Поля периода предназначены для указания временного диапазона поиска. Зависят от выбранного способа задания периода.
- Кнопка **Поиск** позволяет начать поиск данных по заданным параметрам.

При необходимости вы можете остановить процесс поиска, нажав на кнопку **Стоп**. В разделе будут отображены только записи, найденные до прерывания поиска.

Если активен режим базового поиска, под панелью отображается график распределения найденных записей по времени. В правом верхнем углу графика находится кнопка для сворачивания и разворачивания.

Если в запросе содержится оператор `GROUP BY`, слева от таблицы результатов поиска отображается панель с группами найденных событий.

Вид таблицы результатов поиска зависит от открытой вкладки. Доступные вкладки:

- **Данные** — содержит информацию о найденных событиях или записях активного списка.
- **Статистика** — отображает сводные данные по полям найденных событий и записей активного списка с возможностью просмотра рейтинга значений по выбранному полю.

Вкладка доступна, если выполняется базовый поиск по одному хранилищу событий или активному списку.

Интерфейс вкладки Данные

На вкладке **Данные** отображается список событий или записей. Список представлен в виде таблицы, набор столбцов которой формируется в зависимости от текущего режима поиска:

- В режиме базового поиска:
  - Если выбрано одно хранилище событий — столбцы соответствуют полям модели события выбранного хранилища, для которых задано отображение.
  - Если выбраны несколько хранилищ событий — столбцы соответствуют всем полям всех выбранных хранилищ. Если поле отсутствует в модели события, в результатах поиска значение поля в строке этого события будет пустым.

    Если в моделях событий выбранных хранилищ есть поля с одинаковым именем, но разными типами данных, значения этих полей будут приведены к типу `String` и выведены в один столбец результата.
  - Если выбран активный список — столбцы соответствуют полям схемы активного списка.
- В режиме продвинутого поиска столбцы соответствуют полям, заданным с помощью операторов `SELECT` и `GROUP BY`.



По умолчанию в таблице на вкладке **Данные** отображаются столбцы, соответствующие следующим полям:

- для универсальной модели события: `timestamp`, `dvendor`, `dproduct`, `name`, `msg`, `dvc`, `outcome`;
- для универсальной модели 2.0: `timestamp`, `device_product`, `device_hostname`, `description`, `focus`;
- для модели события аудита: `timestamp`, `raw`, `suser`, `eventType`, `moduleKey`, `eventRiskLevel`, `data`, `meta`;
- для пользовательской модели события: `id`, `type`, `timestamp`, `sourceIp`.

При работе с таблицей записей на вкладке **Данные** доступны следующие операции:

- Сортировка по любому столбцу.
- Обновление таблицы.
- Настройка отображения таблицы.

По умолчанию в таблице доступен просмотр не более 500 первых записей.

При выборе конкретной записи в правой части рабочей области отображается ее карточка с детальной информацией.

Интерфейс вкладки Статистика



- Вкладка **Статистика** доступна только в режиме базового поиска по одному хранилищу событий или активному списку.
- Поля с типом данных `array` и `KeyValue` не отображаются в статистике.

На вкладке **Статистика** представлена сводная информация по полям найденных событий или записей активного списка. На вкладке можно просматривать рейтинг значений по каждому из полей, а также общие статистические данные.

Данные по статистике представлены в виде таблицы со следующими столбцами:

- **Поле** — название поля события или записи активного списка.
- **Тип** — тип данных поля.
- **Заполненных значений** — количество заполненных (непустых) значений поля и их процентная доля от общего количества значений.
- **Уникальных значений** — количество уникальных значений поля.
- **Выборка** — количество значений поля, используемых при расчете статистики, и их процентная доля от общего количества.



Поля, которые отсутствуют в хранилище событий, но при этом имеются в универсальной модели или универсальной модели 2.0, не отображаются в таблице статистики для этого хранилища.

Слева от строки каждого поля в таблице расположена кнопка в виде стрелки, позволяющая показать детальную статистику по выбранному полю.



Вы также можете отобразить статистику сразу всех полей. Для этого необходимо использовать кнопку в виде стрелки слева от области заголовка таблицы.

При работе с таблицей полей на вкладке **Статистика** доступны следующие операции:

- Сортировка по любому столбцу.
- Обновление таблицы.
- Настройка отображения таблицы.

Работа с поиском

Доступные операции при работе с поиском:

- Поиск событий и записей
- Создание виджета по поисковому запросу
- Работа с поисковыми шаблонами
- Работа с графиком распределения событий

Поиск событий и записей

Чтобы выполнить поиск по событиям хранилища или записям активного списка:

1. Перейдите в раздел **Поиск**.
2. Выберите хранилища событий или активный список из выпадающего списка **Источник событий** в верхней части раздела. Для быстрого поиска хранилища или активного списка начните вводить

его название.



Хранилища событий и активные списки нельзя выбирать в качестве источников одновременно. Чтобы объединить данные из хранилищ событий и активных списков, вы можете выбрать хранилища событий в качестве источников и обогатить результаты поиска данными из активных списков с помощью операторов `JOIN` и/или `ENRICH`.



Для быстрого поиска хранилища или активного списка в выпадающем списке начните вводить его название.

3. Чтобы искать записи по ключевым словам или RQL-запросу, введите запрос в поле поиска.



- Если количество возвращаемых записей ограничено с помощью ключевого слова `LIMIT` и поиск выполняется по нескольким хранилищам, лимит будет применен к результату поиска по всем выбранным хранилищам.

  Например, выполняется поиск по двум хранилищам, в каждом из которых по 500 событий, удовлетворяющих запросу, а в запросе указано ограничение `LIMIT 500`. В этом случае запрос вернет только 500 записей.

- При установке курсора в поле поиска отображается панель с 5 последними запросами.
- Если в запросе заданы условия поиска по определенным значениям полей с помощью операторов `=` или `IN`, соответствующие поля для найденных записей на вкладке **Данные** будут подсвечиваться.



Если для выбранных хранилищ событий и ролей текущего пользователя заданы политики управления доступа к событиям, их правила будут неявно добавлены в секцию `WHERE` поискового запроса.

4. Чтобы искать записи за период времени, нажмите на кнопку  и выберите из выпадающего списка способ задания периода:

   - **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.
   - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.
   - **Часто используемые** — выберите в разделе один из предлагаемых периодов.



В нижней части выпадающего списка фильтров расположена секция **История поиска** с 5 последними запросами.

5. При необходимости задайте пользовательские фильтры поиска.
6. Нажмите на кнопку **Поиск**. Список записей, соответствующий заданным критериям, отобразится в таблице. Также над таблицей отобразится количество найденных совпадений.

При необходимости вы можете остановить процесс поиска, нажав на кнопку **Стоп**. В разделе будут отображены только записи, найденные до прерывания поиска.

Поиск по нескольким хранилищам выполняется для всех выбранных хранилищ. Если поле, указанное в секции `WHERE`, отсутствует в модели событий какого-либо хранилища, для этого хранилища в качестве значения данного поля используется `NULL`.

Если при выполнении поиска по нескольким хранилищам одно или несколько выбранных хранилищ недоступны, поиск завершится с ошибкой.

- Если в качестве источника данных выбраны несколько хранилищ событий и в запросе фигурируют поля типа `JSON`, то поиск завершится с ошибкой. В таком случае необходимо убрать из запроса поля типа `JSON` или ограничиться поиском только по одному хранилищу.
- Если в качестве источника данных выбраны несколько хранилищ событий, часть из которых использует модели событий, созданные в системе версии ниже 2.4.0, а другая часть — модели, созданные в системе версии 2.4.0 или выше, то поиск завершится с ошибкой. Это связано с переходом на тип данных `DateTime64` в служебных полях моделей событий в версии 2.4.0. В таком случае необходимо ограничиться поиском только по тем хранилищам, все модели которых были созданы либо в версии ниже 2.4.0, либо в версии 2.4.0 или выше.

История запросов поиска хранится в разделе **Инструменты → Диспетчер запросов**.

Особенности задания запроса поиска

- Один и тот же RQL-запрос может приводить к разным результатам поиска, если в искомой базе данных хранится больше записей, чем указано в настройках отображения. Это связано с тем, что по умолчанию результаты поиска возвращаются в произвольном порядке.

  В таких случаях для корректного поиска записей необходимо добавить атрибут `ORDER BY` и явно указать сортировку по требуемому полю.

  Например, для корректного отображения данных с сортировкой по времени требуется явно указать сортировку по полю `timestamp` в RQL-запросе:

  - `ORDER BY timestamp ASC` или `ORDER BY timestamp` — сортировка записей от старых к новым.
  - `ORDER BY timestamp DESC` — сортировка записей от новых к старым.
- При поиске по полю `timestamp` необходимо задавать дату и время в формате `YYYY-MM-DD hh:mm:ss`, где:
  - `YYYY-MM-DD` — дата в формате "год-месяц-день".
  - `hh:mm:ss` — время в формате "час:минута:секунда".

Поиск по полю `timestamp`

```
timestamp = "2025-09-29 10:41:29"
```



Условие для поля `timestamp` также можно задать с помощью пользовательского фильтра или через автоматическую фильтрацию с помощью кнопки ⊞.

- Система позволяет искать данные по подсетям. Формат записи подсети: `<netmask>/<prefix>`. Здесь:
  - `<subnet>` — маска подсети.
  - `<prefix>` — сетевой префикс. Допустимые значения зависят от версии IP-протокола:
    - для IPv4: 0—32;
    - для IPv6: 0—128.

  Поиск по подсетям можно выполнить следующими способами:

  - С помощью оператора `IN`:

    Поиск по IPv4-подсетям с оператором `IN`

    ```
    sourceIp IN ':192.0.2.0/24'
    ```

    Поиск по IPv6-подсетям с оператором `IN`

    ```
    sourceIp IN '::ffff:192.0.2.0/64'
    ```

  - С помощью функции `isIPAddressInRange`:

    Поиск по IPv4-подсетям с функцией `isIPAddressInRange`

    ```
    isIPAddressInRange(IPv4NumToString(sourceIp),'192.0.2.0/24')
    ```

    Поиск по IPv6-подсетям с функцией `isIPAddressInRange`

    ```
    isIPAddressInRange(IPv6NumToString(sourceIp),'::ffff:192.0.2.0/64')
    ```

- Если в запросе поиска необходимо обратиться к полю, название которого совпадает с ключевым словом ClickHouse, например, `SELECT`, `FROM`, `WHERE`, то название поля необходимо заключать в обратные апострофы.

Пример запроса к полю с ключевым словом в названии

```
SELECT `FROM` FROM table
```

Создание виджета по поисковому запросу

Создание виджета по поисковому запросу доступно, только если для поиска выбрано **одно хранилище событий** или **активный список** и найдена хотя бы одна запись.

Чтобы создать виджет на основе поискового запроса:

1. Нажмите на кнопку на панели инструментов. Откроется окно создания виджета.

   При открытии окна в его правой части можно предварительно посмотреть вид диаграммы на виджете.

   > Предварительный вид диаграммы отображается на основе настроек по умолчанию. Чтобы просмотреть актуальный вид диаграммы после изменения настроек виджета, нажмите на кнопку в правой верхней части окна.

2. Выберите из выпадающего списка дашборд, в котором нужно разместить виджет.
3. Введите название виджета.
4. При необходимости введите описание виджета.
5. Выберите тип визуализации для виджета:
   - **Столбчатая диаграмма**;
   - **Карта стран мира**;
   - **Карта с группировкой**;
   - **График**;
   - **Круговая диаграмма**;
   - **Столбчатая диаграмма с накоплением**;
   - **Таблица**;
   - **Итоги**.

   > По умолчанию установлен тип **Таблица**.

6. Задайте лимит строк для виджетов вручную или с помощью кнопок — и +. Лимит строк определяет максимальное количество строк, которые будут использованы для построения виджета. Допустимый диапазон лимита строк: 1—1000.

   По умолчанию для круговых диаграмм установлен лимит в 100 строк, для столбчатых диаграмм — 1000 строк.

   > Если задан лимит более 6 строк, то на столбчатой диаграмме будет отображаться только часть подписей, с равномерными пропусками, чтобы избежать наложения.

7. При необходимости измените запрос на языке RQL, заданный по умолчанию.

   > Запрос формируется на основе исходного поискового запроса и примененных

фильтров, которые добавляются в секцию `WHERE`.

8. Укажите параметры в секции **Сопоставление полей**. Параметры зависят от выбранного типа диаграммы:
   - Для столбчатой диаграммы: **Легенда**, **Значение**, **Группировка**.
   - Для карты стран мира: **Страна**, **Количество**.
   - Для карты с группировкой: столбцы и соответствующие им атрибуты из запроса.

     Для карты с группировкой также доступна секция полей **Координаты**, в которой необходимо указать атрибуты из запроса, возвращающие широту и долготу.

   - Для графика: **Ось X**, **Ось Y**, **Группировка**.
   - Для круговой диаграммы: **Легенда**, **Значение**.
   - Для столбчатой диаграммы с накоплением: **Легенда**, **Значение**, **Группировка**.
   - Для таблицы: столбцы и соответствующие им атрибуты из запроса — по умолчанию заполнены идентично их порядку и отображению согласно настройкам таблицы.
   - Для итогов: **Значение**.

     Подробности о разных типах диаграмм виджетов приведены в разделе Типы диаграмм.

     При использовании группировки на виджетах отображаются только первые 20 групп.

9. Выберите из выпадающего списка способ задания периода и установите период:
   - **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.
   - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.

   По умолчанию настройки периода заданы согласно параметрам поиска.

10. Для регулярного обновления данных на виджете установите переключатель **Включить автообновление** в активное положение и укажите периодичность обновления:
    a. Задайте частоту обновления вручную или с помощью кнопок — и +.
    b. Выберите из выпадающего списка единицы времени периода.

    Не рекомендуется включать автоматическое обновление данных для каждого виджета дашборда, так как это может излишне нагрузить систему. Подробности приведены в разделе Обновление данных.

11. Нажмите на кнопку **Создать**. Система создаст виджет и отобразит соответствующее

уведомление. Чтобы открыть дашборд с добавленным виджетом, нажмите на кнопку **Перейти к виджету**.

Работа с поисковыми шаблонами

Для удобства поиска можно сохранить текущий запрос в виде шаблона.

Доступные операции над поисковыми шаблонами:

- Создание шаблона
- Поиск шаблона
- Применение шаблона
- Обновление шаблона
- Просмотр конфигурации шаблона

Создание шаблона

Чтобы создать шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Нажмите на кнопку **Создать шаблон**. Отобразится окно сохранения шаблона.
4. Введите название шаблона.
5. При необходимости введите описание шаблона.
6. Выберите каталог раздела **Экспертиза**, в который будет помещен шаблон.
7. Чтобы сохранить в шаблоне текущий заданный период, установите флажок **Сохранить временной период**.
8. Чтобы сохранить в шаблоне имеющиеся пользовательские фильтры, установите флажок **Сохранить фильтры**.
9. Нажмите на кнопку **Создать**. Система создаст шаблон и отобразит соответствующее уведомление. Новый шаблон появится на панели шаблонов.

Поиск шаблона

Чтобы найти нужный шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Введите название требуемого шаблона в поле поиска. На панели отобразятся шаблоны, в названии которых встречается введенный запрос.

Применение шаблона

Чтобы применить шаблон:

1. Перейдите в раздел **Поиск**.

2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Выберите шаблон из списка. Система применит шаблон для поиска событий или записей активных списков.

Обновление шаблона

Чтобы обновить существующий шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Нажмите на кнопку ⇄ в правой части строки нужного шаблона. Отобразится окно обновления шаблона.
4. Внесите необходимые изменения в настройки шаблона.
5. Нажмите на кнопку **Обновить** и подтвердите обновление. Шаблон будет обновлен. В нем будет сохранен поисковый запрос, введенный в поле запроса на момент редактирования. Фильтры и временной период будут сохранены, если установлены флажки **Сохранить временной период** и **Сохранить фильтры**.

Просмотр конфигурации шаблона

Чтобы просмотреть RObject-конфигурацию шаблона:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Нажмите на кнопку ↗ в правой части строки нужного шаблона. Система откроет на новой вкладке браузера раздел **Экспертиза** и отобразит окно просмотра RObject-конфигурации выбранного шаблона поиска.

Работа с графиком распределения событий



График распределения событий доступен только в режиме базового поиска.

График распределения найденных событий или записей по времени отображается после выполнения поискового запроса.

Результаты на графике расположены от более старых к более новым. Для построения графика используются следующие временные значения:

- При поиске в хранилище событий — временные метки событий (`timestamp`).
- При поиске в активных списках — дата создания записей.

Результаты разбиваются на периоды, длительность которых зависит от общего периода поиска. При наведении курсора на определенный период отображаются даты и время его начала и окончания, а также количество событий или записей в этом периоде.

Доступные операции над графиком:

- Отображение и скрытие графика

- Уточнение периода поиска с помощью графика

Отображение и скрытие графика

Панель с графиком по умолчанию свернута. Чтобы отобразить график, нажмите на кнопку **Показать график** (⌄). Над таблицей найденных записей раскроется панель с графиком распределения по времени.

Чтобы скрыть график, нажмите на кнопку **Скрыть график** (⌃). Панель с графиком распределения найденных событий по времени свернется.

Уточнение периода поиска с помощью графика

Вы можете уточнить временной период поискового запроса с помощью графика. Для этого:

1. Наведите курсор на график. Курсор примет форму крестика.
2. Нажмите на нужный период или, удерживая курсор, выделите несколько периодов. Система скорректирует поисковый запрос с учетом выбранного временного диапазона и отобразит уточненные данные в таблице и на графике.

Работа с группами событий или записей



Панель групп доступна только в режиме продвинутого поиска, если в запросе есть оператор `GROUP BY`.

Панель с группами найденных событий или записей отображается после выполнения поискового запроса. Количество групп отображается в заголовке панели.

Панель содержит следующие компоненты:

- Кнопка ⫼ позволяет выбрать поле для сортировки списка групп.
- Кнопка ⤒ позволяет экспортировать данные групп
- Поле поиска позволяет отфильтровать список по значениям полей.

По умолчанию список групп сортируется по первому полю в результате RQL-запроса:

- для текстового поля — по возрастанию;
- для числового поля — по убыванию.

Доступные операции над группами событий или записей:

- Экспорт данных групп
- Просмотр событий или записей группы

Экспорт данных групп

Чтобы экспортировать группу событий или записей активного списка, нажмите на кнопку ⤒ на панели групп. Система загрузит данные всех групп на устройство пользователя одним файлом в формате CSV и отобразит соответствующее уведомление.

Просмотр событий или записей группы

Чтобы отобразить события или записи из конкретной группы, нажмите на карточку или строку группы. В списке событий или записей отобразятся только те, которые относятся к этой группе.

Работа со списком событий или записей активного списка

Список найденных событий или записей отображается в виде таблицы на вкладке **Данные**.

Доступные операции над списком событий или записей:

- Настройка количества отображаемых событий
- Пользовательская фильтрация списка событий
- Автоматическая фильтрация списка по значению поля
- Настройка таблицы

Настройка количества отображаемых событий

По умолчанию отображаются не более 500 первых найденных записей.

Чтобы изменить количество отображаемых записей, выберите значение из выпадающего списка в нижней части страницы. Доступные варианты: 500, 1000, 5000, 10 000, 25 000.

Пользовательская фильтрация списка событий

Доступные операции над пользовательскими фильтрами:

- Добавление фильтра
- Применение фильтра
- Изменение фильтра
- Отключение фильтра
- Включение фильтра
- Инвертирование фильтра
- Удаление фильтра



- Если в секции `WHERE` RQL-запроса заданы условия фильтрации, при формировании запроса в БД они объединяются оператором `AND` с фильтрами, настроенными через графический интерфейс.
- Если в настройках профиля изменен часовой пояс, фильтры по времени перестроятся с учетом этого изменения.

  Например, если изначально был выбран часовой пояс **GMT+3 Moscow** и настроена фильтрация по времени `00:00:00`, то после изменения часового пояса на **GMT+5 Yekaterinburg** фильтрация данных начнет выполняться по времени `02:00:00`. В таком случае, если требуется, чтобы фильтрация происходила по времени `00:00:00` в текущем часовом поясе, необходимо задать фильтр заново.

Добавление фильтра

Чтобы добавить пользовательский фильтр списка:

1. Нажмите на кнопку **Добавить фильтр**. Отобразится окно добавления фильтра.
2. Укажите поле, по которому будут фильтроваться записи.



Выбор полей ограничен моделями выбранных хранилищ событий или схемой активного списка. Для полей типа `Array` создание фильтра недоступно.

Чтобы обратиться к вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

В режиме продвинутого поиска можно задать условия фильтрации полей типа `Array`, а также полей, не входящих в модель (например, псевдонимов), в секции `WHERE` RQL-запроса.

Для полей автоматически устанавливается тип: для вложенных полей `JSON` — тип `String`, для остальных полей — тип согласно модели событий. Поменять тип невозможно.

3. Выберите из выпадающего списка оператор, который будет применен к значению поля. Список доступных операторов зависит от типа поля:
   - Для всех типов:
     - `=` — равно.



Если в фильтре задано условие поиска по определенному значению поля с помощью оператора `=`, соответствующее поле для найденных записей на вкладке **Данные** будет подсвечиваться.

     - `!=` — не равно.
   - Для всех типов, кроме `JSON`:
     - `In` — совпадает с перечисленными значениями. Значения указываются через запятую.



Если в фильтре задано условие поиска по определенному значению поля с помощью оператора `In`, соответствующее поле для найденных записей на вкладке **Данные** будет подсвечиваться.

   - Для строковых типов и `JSON`:
     - `Like` — соответствует заданному шаблону.



Шаблон может включать обычные символы и следующие метасимволы:

- `%` — обозначает любое количество произвольных символов или отсутствие символов.
- `_` — обозначает один произвольный символ.
- `\` — используется для экранирования символов `%`, `_` и `\`.

- Для числовых форматов и дат:
  - `>` — больше.
  - `<` — меньше.
  - `>=` — больше или равно.
  - `<=` — меньше или равно.
  - `Between` — находится в заданном диапазоне.

4. Укажите значение поля.



Указывайте значение `JSON` без символов новой строки и пробелов, например:

```
{"is_active":true,"version":"2"}
```

Для типа данных `UInt` не поддерживаются отрицательные числа. Чтобы получить корректные результаты поиска, указывайте значение больше либо равно 0.

5. Если нужно инвертировать фильтр, установите переключатель **Инвертировать (NOT)** в активное положение.
6. Если нужно указать название фильтра, установите переключатель **Настроить заголовок** в активное положение и введите название.
7. Нажмите на кнопку **Добавить**. Созданный фильтр появится рядом с кнопкой **Добавить фильтр**.



Если вы не указывали название фильтра, система создаст его автоматически на основе данных о выбранном поле, операторе и значении.

Если задан хотя бы один фильтр, система отобразит кнопки:

- **Сбросить все** — удаление всех имеющихся фильтров.
- — выпадающее меню команд фильтрации:
  - **Отключить все** — отключение всех фильтров.
  - **Включить все** — включение всех фильтров.
  - **Инвертировать (NOT) все** — инвертирование всех фильтров.
  - **Удалить все** — удаление всех фильтров.



Чтобы уточнить период поиска, используйте график распределения по времени.

Применение фильтра

Чтобы применить настроенный фильтр списка, нажмите на кнопку **Поиск**. Будут найдены события, удовлетворяющие условиям фильтра.



- Если в качестве источника данных выбраны несколько хранилищ событий, при применении фильтра поиск осуществляется по всем этим хранилищам.
- Если в хранилище отсутствует поле, по которому применяется фильтр, для этого

хранилища в качестве значения данного поля используется `NULL`.

Изменение фильтра

Чтобы изменить заданный пользовательский фильтр списка:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Изменить** в выпадающем меню. Откроется окно изменения фильтра.
3. Внесите изменения в требуемые поля.
4. Нажмите на кнопку **Сохранить**. Измененные настройки фильтра будут сохранены.

Отключение фильтра

По умолчанию пользовательский фильтр создается включенным и учитывается при фильтрации списка.

Чтобы отключить фильтр:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Отключить** в выпадающем меню. Фильтр будет отключен и не будет использоваться для фильтрации списка событий.



Вы также можете отключить сразу все настроенные фильтры. Для этого выберите опцию **Отключить все** в выпадающем меню кнопки  .

Включение фильтра

Чтобы включить пользовательский фильтр:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Включить** в выпадающем меню. Фильтр будет применен к списку событий.



Вы также можете включить сразу все настроенные фильтры. Для этого выберите опцию **Включить все** в выпадающем меню кнопки  .

Инвертирование фильтра

Чтобы инвертировать условие пользовательского фильтра:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Инвертировать (NOT)** в выпадающем меню. Условие фильтра будет заменено на противоположное.



Вы также можете инвертировать условия сразу всех настроенных фильтров. Для

этого выберите опцию **Инвертировать (NOT) все** в выпадающем меню кнопки .

Удаление фильтра

Чтобы удалить пользовательский фильтр:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Удалить** в выпадающем меню. Фильтр будет сброшен без возможности восстановления.



Вы также можете удалить сразу все настроенные фильтры. Для этого нажмите на кнопку **Сбросить все** или выберите опцию **Удалить все** в выпадающем меню кнопки .

Автоматическая фильтрация списка по значению поля



Автоматическая фильтрация доступна только в режиме базового поиска для полей всех типов, кроме `Array` и `JSON`.

Для вложенных полей `JSON` автоматическая фильтрация недоступна, если в ключе содержится индекс массива.

Вы можете настроить автоматическую фильтрацию списка несколькими способами:

- Настройка автоматической фильтрации в списке
- Настройка автоматической фильтрации в карточке
- Настройка автоматической фильтрации по статистике поля

Настройка автоматической фильтрации в списке

Чтобы настроить автоматическую фильтрацию списка по значению поля:

1. Наведите курсор на значение требуемого поля какого-либо события или записи в списке. Рядом с полем отобразятся кнопки фильтрации:
   - — искать события или записи с этим значением.

     Если применить данный фильтр по значению поля, соответствующие поля для найденных записей на вкладке **Данные** будут подсвечиваться.

   - — исключить события или записи с этим значением. Исключение задается через автоматическое создание инвертированного фильтра по выбранному значению поля.
2. Нажмите на одну из кнопок фильтрации для автоматического создания фильтра. Заданный фильтр отобразится в списке фильтров слева от кнопки **Добавить фильтр**.

Настройка автоматической фильтрации в карточке

Чтобы настроить автоматическую фильтрацию списка по значению поля из карточки события:

1. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
2. Наведите курсор на требуемое поле в карточке. Система отобразит кнопки фильтрации в строке этого поля:
   - ⊞ — искать события или записи с этим значением.

     Если применить данный фильтр по значению поля, соответствующие поля для найденных записей на вкладке **Данные** будут подсвечиваться.

   - ⊟ — исключить события или записи с этим значением. Исключение задается через автоматическое создание инвертированного фильтра по выбранному значению поля.
3. Нажмите на одну из кнопок фильтрации для автоматического создания фильтра. Заданный фильтр отобразится в списке фильтров слева от кнопки **Добавить фильтр**.

Настройка автоматической фильтрации по статистике поля

В режиме базового поиска по одному хранилищу событий или активному списку на панели статистики поля доступна фильтрация списка по значениям выбранного поля, а также по информации о датах его значений.

Чтобы настроить автоматическую фильтрацию списка по данным статистики поля:

1. Нажмите на кнопку действий ⋮ в заголовке столбца нужного поля и выберите опцию **Статистика**. В правой части рабочей области отобразится панель с детальной статистикой по столбцу выбранного поля.
2. Наведите курсор на строку требуемого поля на панели статистики. Система отобразит кнопки фильтрации в строке этого поля:
   - ⊞ — искать события или записи с этим значением.

     Если применить данный фильтр по значению поля, соответствующие поля для найденных записей на вкладке **Данные** будут подсвечиваться.

   - ⊟ — исключить события или записи с этим значением. Исключение задается через автоматическое создание инвертированного фильтра по выбранному значению поля.
3. Нажмите на одну из кнопок фильтрации для автоматического создания фильтра. Заданный фильтр отобразится в списке фильтров слева от кнопки **Добавить фильтр**.

Настройка таблицы

Доступные операции по настройке таблицы:

- Выбор отображаемых полей таблицы
- Работа с шаблонами настроек таблицы



- На панели настроек отображаются шаблоны, применяемые к таблице текущей вкладки.
- Шаблоны для вкладок **Данные** и **Статистика** настраиваются и сохраняются

отдельно друг от друга.

- Каждый набор шаблонов привязан к конкретному хранилищу событий или активному списку.

Выбор отображаемых полей таблицы



Настройка отображаемых полей таблицы доступна только в режиме базового поиска. В режиме продвинутого поиска отображаемые поля определяются выполняемым RQL-запросом.

Чтобы выбрать отображаемые поля таблицы, используйте панель настроек либо карточку события или записи.

Чтобы настроить отображение полей с помощью карточки события или записи:

1. На вкладке **Данные** нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
2. Наведите курсор на требуемое поле в карточке. В строке поля отобразятся кнопки действий над ним.
3. Нажмите на кнопку ⋮ и задайте параметры отображения в выпадающем меню:
   - **Скрыть колонку** — скрытие столбца поля из таблицы;
   - **Отобразить колонку** — отображение столбца поля в таблице.



Если не выбрано ни одно поле для отображения, в таблице отображаются следующие поля:

- `timestamp` — дата и время получения события коллектором.
- `summary` — все заполненные поля найденного события в формате `<field> : <value>`.

  Здесь:

  - `<field>` — название поля;
  - `<value>` — значение поля.

Работа с шаблонами настроек таблицы



Шаблоны настроек таблицы доступны только в режиме базового поиска.

Для удобства можно сохранить часто используемые настройки таблицы в виде шаблона.

Доступные операции над шаблонами настроек таблицы:

- Создание шаблона настроек таблицы
- Применение шаблона настроек таблицы
- Поиск шаблона настроек таблицы

- Изменение шаблона настроек таблицы
- Удаление шаблона настроек таблицы

Создание шаблона настроек таблицы

Чтобы создать шаблон настроек таблицы:

1. Нажмите на кнопку **Сохранить как шаблон** в нижней части панели настроек. Отобразится окно сохранения шаблона настроек.
2. Введите название шаблона настроек.
3. Если необходимо сделать шаблон доступным для всех пользователей системы, установите флажок **Сохранить для всех пользователей**.
4. Нажмите на кнопку **Сохранить**. Система сохранит шаблон и отобразит соответствующее уведомление. Новый шаблон появится в выпадающем списке **Шаблон**.

Применение шаблона настроек таблицы

Чтобы применить шаблон настроек таблицы, выберите его из выпадающего списка **Шаблон**. Настройки выбранного шаблона будут применены к таблице событий.

Поиск шаблона настроек таблицы

Чтобы найти шаблон настроек таблицы, введите его название в поле **Шаблон**. В выпадающем списке отобразятся шаблоны, в названии которых встречается введенный запрос.

Изменение шаблона настроек таблицы

Изменение шаблона настроек происходит путем замены его полей на текущие настройки таблицы.

Чтобы изменить шаблон:

1. Выберите из выпадающего списка **Шаблон** требуемый шаблон.
2. Внесите необходимые изменения в настройки.
3. В выпадающем списке **Шаблон** нажмите на кнопку **Заменить шаблон** (⇄) в строке нужного шаблона. Отобразится окно подтверждения замены настроек шаблона.
4. Нажмите на кнопку **Заменить**. Система сохранит внесенные изменения в шаблоне и отобразит соответствующее уведомление. Настройки шаблона будут изменены.

Удаление шаблона настроек таблицы

Чтобы удалить шаблон настроек таблицы:

1. В выпадающем списке **Шаблон** нажмите на кнопку **Удалить шаблон** (🗑) в строке нужного шаблона. Отобразится окно подтверждения удаления.
2. Нажмите на кнопку **Удалить**. Система удалит шаблон и отобразит соответствующее уведомление. Удаленный шаблон исчезнет из списка.

Работа со статистикой полей

- Статистика полей доступна только в режиме базового поиска по одному хранилищу событий или активному списку.
- Поля с типом данных `Array`, `JSON` и `KeyValue` не отображаются в статистике.
- Для расчета статистики учитываются первые 100 000 записей, удовлетворяющие запросу поиска в базе данных. Из-за этого данные в статистике могут быть неактуальными, если запросу поиска удовлетворяет более 100 000 записей.

  При этом, ввиду того что по умолчанию результаты поиска возвращаются в произвольном порядке, в описываемой ситуации один и тот же RQL-запрос может приводить к разным результатам поиска, а следовательно, и к разным показателям статистики.

  Таким образом, чтобы получить более корректные и актуальные данные статистики, рекомендуется сортировать данные по определенному полю через оператор `ORDER BY` и ограничивать выборку данных через оператор `LIMIT`.

- Если поиск был выполнен с указанием времени **За последний период**, то при повторном поиске результаты статистики могут измениться из-за сдвига временного диапазона поиска.

На вкладке **Статистика** представлена сводная статистика по полям всех найденных событий или записей активного списка. Здесь можно просматривать рейтинг значений по полю, а также детализированные статистические данные, включая тип данных, количество значений, процент заполненных и уникальных значений.

Поля, которые отсутствуют в хранилище событий, но при этом имеются в универсальной модели или универсальной модели 2.0, не отображаются в таблице статистики для этого хранилища.

Таблица на вкладке **Статистика** содержит следующие столбцы:

- **Поле** — название поля события или записи активного списка.
- **Тип** — тип данных поля.
- **Заполненных значений** — отображаются как процент вхождений, содержащих непустые значения, так и их абсолютное количество.
- **Уникальных значений** — количество уникальных значений поля.
- **Выборка** — процент уникальных значений от общего количества, а также их абсолютное количество.

Доступные операции при работе со статистикой:

- Просмотр детальной статистики по полю
- Создание виджета со статистикой по полю

Просмотр детальной статистики по полю

Чтобы отобразить детальную статистику по полю:

1. Перейдите на вкладку **Статистика**.
2. Нажмите на стрелку ⌄ в строке нужного поля. В раскрывшейся панели система отобразит детализированную статистику, сгруппированную по следующим секциям:
   - **Общие сведения**:
     - **Тип поля** — тип данных поля.
     - **Заполненных значений** — процентная доля заполненных значений (не пустых) от общего количества значений поля.
     - **Уникальных значений** — количество уникальных (неповторяющихся) значений поля.
     - **Выборка** — процентное соотношение количества уникальных значений к общему числу значений поля.
   - **Рейтинг значений**:
     - Частота встречаемости значения, указана в процентах.
     - Количество повторений значения в событиях или записях, приведено в скобках.
     - Кнопки для фильтрации событий по значению и копирования значения в буфер обмена.

     В рейтинге отображаются до 10 самых частых или редких значений, в зависимости от направления сортировки.

     При необходимости вы можете загрузить следующие 10 значений при их наличии, нажав на кнопку **Загрузить еще**.

     Справа от заголовка секции также расположены следующие кнопки:

     -  или  позволяют отфильтровать значения в рейтинге по возрастанию или убыванию.
     - **Создать виджет** () позволяет создать виджет на основе статистики по полю.

     Секция **Рейтинг значений** отображается для полей следующих типов данных: `Bool`, `Enum`, `Float`, `IPv4`, `IPv6`, `LCString`, `Mac`, `String`, `UInt`, `UUID`.
   - **Даты**:
     - **Первая дата** — минимальное значение даты из всех найденных событий.
     - **Последняя дата** — максимальное значение даты из всех найденных событий.

     Секция **Даты** отображается для полей с типом данных `DateTime` или `DateTime64`.

Детальную статистику по полю можно также просмотреть на вкладке **Данные**:

- Через меню действий:

1. Наведите курсор на какое-либо значение требуемого поля в списке. Рядом с полем отобразятся кнопки действий.
2. Нажмите на кнопку действий ⋮ в заголовке столбца нужного поля и выберите опцию **Статистика**. В правой части рабочей области отобразится панель с детальной статистикой по столбцу выбранного поля.

- Через карточку события или записи:
  1. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
  2. Наведите курсор на требуемое поле в карточке. В строке поля отобразятся кнопки действий.
  3. Нажмите на кнопку ⋮ и выберите опцию **Статистика**. Вместо карточки события или записи в правой части рабочей области отобразится панель с детальной статистикой по столбцу выбранного поля.



При просмотре детальной статистики на вкладке **Данные** кнопка **Создать виджет** () отображается в нижней части панели статистики.



При наведении курсора на строку любого поля на панели статистики, в строке отображаются следующие кнопки:

-  — кнопка копирования данных из этого поля в буфер обмена.
-  и  — кнопки фильтрации списка событий по данным из этого поля.

Создание виджета со статистикой по полю

Чтобы создать виджет со статистикой по полю:

1. На панели статистики нажмите на кнопку **Создать виджет** (). Откроется окно создания виджета.



Если у вас открыт просмотр детальной статистики по всем полям, то перед созданием виджета необходимо дождаться прогрузки статистики.

При открытии окна в его правой части можно предварительно посмотреть вид диаграммы на виджете.



Предварительный вид диаграммы отображается на основе настроек по умолчанию. Чтобы просмотреть актуальный вид диаграммы после изменения настроек виджета, нажмите на кнопку  в правой верхней части окна.

2. Выберите из выпадающего списка дашборд, в котором нужно разместить виджет.
3. Введите название виджета.
4. При необходимости введите описание виджета.
5. Выберите тип визуализации для виджета:
   - **Столбчатая диаграмма**;

- **Карта стран мира**;
- **Карта с группировкой**;
- **График**;
- **Круговая диаграмма**;
- **Столбчатая диаграмма с накоплением**;
- **Таблица**;
- **Итоги**.



По умолчанию установлен тип **Столбчатая диаграмма**.

6. Задайте лимит строк для виджетов вручную или с помощью кнопок — и +. Лимит строк определяет максимальное количество строк, которые будут использованы для построения виджета. Допустимый диапазон лимита строк: 1—1000.

   По умолчанию для круговых диаграмм установлен лимит в 100 строк, для столбчатых диаграмм — 1000 строк.



Если задан лимит более 6 строк, то на столбчатой диаграмме будет отображаться только часть подписей, с равномерными пропусками, чтобы избежать наложения.

7. При необходимости измените запрос на языке RQL, заданный по умолчанию.



Запрос формируется на основе исходного поискового запроса и примененных фильтров, которые добавляются в секцию `WHERE`. Поле, по которому построена статистика, добавляется в секцию `GROUP BY`. В запрос автоматически добавляется функция `count()` для подсчета количества записей.

8. Укажите параметры в секции **Сопоставление полей**. Параметры зависят от выбранного типа диаграммы:
   - Для столбчатой диаграммы:
     - **Легенда** — по умолчанию установлено название поля, по которому построена статистика.
     - **Значение** — по умолчанию установлено значение `count`.
     - **Группировка**.
   - Для карты стран мира: **Страна**, **Количество**.
   - Для карты с группировкой: столбцы и соответствующие им атрибуты из запроса.



Для карты с группировкой также доступна секция полей **Координаты**, в которой необходимо указать атрибуты из запроса, возвращающие широту и долготу.

   - Для графика: **Ось X**, **Ось Y**, **Группировка**.
   - Для круговой диаграммы:

        - **Легенда** — по умолчанию установлено название поля, по которому построена статистика.
        - **Значение** — по умолчанию установлено значение `count`.
    - Для столбчатой диаграммы с накоплением: **Легенда**, **Значение**, **Группировка**.
    - Для таблицы: столбцы и соответствующие им атрибуты из запроса.
    - Для итогов: **Значение** — по умолчанию установлено значение `count`.



Подробности о разных типах диаграмм виджетов приведены в разделе Типы диаграмм.



При использовании группировки на виджетах отображаются только первые 20 групп.

9. Выберите из выпадающего списка способ задания периода и установите период:
    - **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.
    - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.

    По умолчанию настройки периода заданы согласно параметрам поиска.
10. Для регулярного обновления данных на виджете установите переключатель **Включить автообновление** в активное положение и укажите периодичность обновления:
    a. Задайте частоту обновления вручную или с помощью кнопок — и +.
    b. Выберите из выпадающего списка единицы времени периода.



Не рекомендуется включать автоматическое обновление данных для каждого виджета дашборда, так как это может излишне нагрузить систему. Подробности приведены в разделе Обновление данных.

11. Нажмите на кнопку **Создать**. Система создаст виджет и отобразит соответствующее уведомление. Чтобы открыть дашборд с добавленным виджетом, нажмите на кнопку **Перейти к виджету**.

Работа с событием или записью активного списка

Доступные операции над событиями или записями активного списка:

- Просмотр события или записи
- Копирование события или записи
- Копирование данных поля
- Экспорт событий и записей

Просмотр события или записи

Чтобы просмотреть найденное событие или запись активного списка:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.

В верхней части карточки отображается идентификатор записи. Справа от идентификатора расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранной записи.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Состав и порядок полей карточки зависит от режима поиска и типа сущности:

- Карточка события в режиме базового поиска содержит следующие вкладки:
  - **Информация** — поля модели, а также информация об исходном событии или базовых событиях.
  - **Данные JSON** — данные полей с типом данных `JSON`.
- Карточка записи активного списка в режиме базового поиска содержит группы полей:
  - Закрепленные поля, отсортированные по времени закрепления.
  - Все остальные поля, упорядоченные согласно схеме активного списка.
- В режиме продвинутого поиска поля карточки не сгруппированы и отображаются в порядке, указанном в секции `SELECT` RQL-запроса.

В режиме базового поиска при наведении курсора на строку поля отображаются следующие кнопки:

- ⊞ и ⊟ — кнопки фильтрации списка по данным из этого поля — для всех полей, кроме полей типа `JSON`.
- ⋮ — кнопка, открывающая меню следующих дополнительных опций над полем:
  - копирование данных из этого поля в буфер обмена;
  - скрытие и отображение столбца этого поля в списке — для всех полей, кроме полей типа `JSON`;
  - просмотр статистики столбца этого поля — для всех полей, кроме полей типа `Array`, `JSON` и `KeyValue`.



Просмотр статистики доступен только в режиме базового поиска по одному хранилищу событий или активному списку.

В режиме базового поиска можно настраивать отображение полей в карточке:

- Чтобы закрепить поле, наведите курсор на его строку и нажмите на кнопку 📌, появившуюся слева от названия поля.

  Чтобы открепить поле, нажмите на кнопку 📌 слева от его названия.



Список закрепленных полей сохраняется индивидуально для текущего пользователя системы в рамках конкретного хранилища событий или активного списка.

- Поле **Поиск по полям** позволяет искать поля по названию и значению. Закрепленные поля не скрываются при поиске.
- Переключатель **Скрыть пустые поля события** (для события) или **Скрыть пустые поля записи** (для записи активного списка) позволяет скрыть поля, не имеющие значений. Закрепленные поля не скрываются при установленном переключателе.

Вкладка Информация

Вкладка **Информация** содержит следующие группы полей:

- Идентификатор и название хранилища, в котором находится событие.
- Названия и значения всех полей в модели событий хранилища, в котором находится событие.



Если выполняется поиск по нескольким хранилищам, то в карточке события отображаются только поля модели событий хранилища, в котором оно находится.

Поля расположены в следующем порядке:

  - Закрепленные поля, отсортированные по времени закрепления.
  - Остальные поля модели события в соответствии с порядком в ней, кроме поля `raw`.
  - Поле `_storageId`, содержащее идентификатор хранилища, в котором находится событие. Отображается только при поиске по нескольким хранилищам.
  - Разобранные данные полей с типом `JSON` всех уровней вложенности. Поля отображаются с названиями вида `<Название поля модели>.<Ключ 1>.<Ключ 2>⋯.<Ключ N>`. Например, `data.source.ip`.

- **Исходное событие** (для базового события) — исходное событие, лежащее в основе просматриваемого события, то есть содержимое поля `raw` в структурированном виде.

  Кнопка **Скопировать** () позволяет скопировать данные в буфер обмена.

- **Базовые события** (для события корреляции) — список базовых событий в виде таблицы со следующими столбцами:

  - **ID**;
  - **Timestamp**.

  Кнопка **Найти все** () позволяет отфильтровать события по полю `id`, используя все значения из списка.

  Кнопки  и  в каждой строке списка позволяют отфильтровать события по полю `id` со значением текущего события.

Вкладка Данные JSON

Вкладка **Данные JSON** содержит данные каждого поля с типом JSON в виде интерактивного дерева элементов.

Кнопка **Скопировать** (⧉) позволяет скопировать данные поля в буфер обмена.

Копирование события или записи

Чтобы скопировать информацию в буфер обмена:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
3. Нажмите на кнопку ⧉ в верхней части карточки. Система сохранит данные в буфер обмена и отобразит соответствующее уведомление.

Копирование данных поля

Вы можете копировать данные полей непосредственно из списка или из карточки. Статистику по конкретному полю можно скопировать из панели статистики соответствующего поля.

Копирование данных из списка

Чтобы скопировать данные поля из списка:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Наведите курсор на значение требуемого поля какого-либо события или записи в списке. Рядом с полем отобразятся кнопки действий над ним.
3. Нажмите на кнопку ⋮ и выберите опцию **Копировать**. Данные выбранного поля будут скопированы в буфер обмена.

Копирование данных из карточки

Чтобы скопировать данные поля из карточки:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
3. Скопируйте данные нужного поля:
   - Чтобы скопировать данные исходного события или данные JSON, нажмите на кнопку **Скопировать** (⧉) у соответствующего поля.
   - Чтобы скопировать данные конкретного поля события:
     a. Наведите курсор на требуемое поле в карточке. В строке поля отобразятся кнопки действий над ним.

b. Нажмите на кнопку ⋮ и выберите опцию **Копировать**.

Данные выбранного поля будут скопированы в буфер обмена.

Копирование данных статистики поля

В режиме базового поиска по одному хранилищу событий или активному списку на панели статистики доступно копирование значений выбранного поля, а также временных меток соответствующих значений.

Чтобы скопировать данные с панели статистики поля:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на кнопку действия ⋮ в заголовке столбца нужного поля и выберите опцию **Статистика**. В правой части рабочей области отобразится панель с детальной статистикой по столбцу выбранного поля.
3. Наведите курсор на строку требуемого поля на панели статистики. Система отобразит кнопку копирования данных в строке этого поля.
4. Нажмите на кнопку . Данные выбранной строки будут скопированы в буфер обмена.

Экспорт событий и записей



При экспорте найденных данных система автоматически выполняет повторный запрос на поиск с теми же параметрами, даже если поиск уже был выполнен вручную. Это может привести к ситуациям, когда в экспорт попадают новые данные, найденные за промежуток времени между ручным поиском и экспортом.

Чтобы избежать таких ситуаций и добиться совпадения экспортированных данных и данных в таблице раздела, необходимо указывать конкретный временной период поиска. Например, с `2025-05-15, 12:00:00` по `2025-05-15, 12:05:00`.

Чтобы экспортировать группу событий или записей активного списка:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на кнопку на панели инструментов. Система отобразит окно экспорта.
3. Выберите из выпадающего списка формат экспорта: CSV или JSON.
4. Если установлен флажок **Исключить пустые поля**, из результатов экспорта исключаются поля без значений. Для включения всех полей в результат экспорта снимите этот флажок.
5. Нажмите на кнопку **Экспортировать**. Система загрузит все вхождения, представленные в текущей выдаче, на устройство пользователя одним файлом в выбранном формате и отобразит соответствующее уведомление.

Сравнение событий

Сравнение различных событий между собой позволяет проводить более глубокий анализ и исследование поступающих событий с целью выявления подозрительных отклонений.

Доступные операции по сравнению событий:

- Добавление события в сравнение
- Просмотр сравнения
- Настройка сравнения полей событий
- Копирование событий в сравнении
- Удаление события из сравнения

Добавление события в сравнение

Чтобы добавить событие в сравнение:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на строку события в списке. Система отобразит в правой части экрана карточку этого события с подробной информацией о нем.
3. Выберите опцию **Добавить в сравнение** в выпадающем меню кнопки действий ⋮ в верхней части карточки. Система добавит событие в список сравнения и отобразит соответствующее уведомление.



В сравнение можно добавлять не более десяти событий.

Просмотр сравнения

Чтобы просмотреть добавленные в сравнение события:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.



Кнопка доступна только при наличии событий, добавленных в сравнение. Вы также можете перейти на страницу просмотра сравнения с помощью кнопки **Перейти в список сравнения**, отображаемой в уведомлении при добавлении события в сравнение.

Чтобы вернуться в раздел **Поиск**, нажмите на кнопку **Назад** в левой верхней части экрана.

Панель инструментов страницы сравнения событий включает в себя следующие компоненты:

- Кнопка копирует данные всех событий в сравнении в буфер обмена.
- Кнопка удаляет все события из сравнения.
- Переключатель **Показать отличия** позволяет отобразить только те поля, значения которых отличаются в каком-либо из сравниваемых событий.
- Переключатель **Скрыть пустые поля** позволяет скрыть поля, не имеющие значений ни в одном из сравниваемых событий.

- Переключатель **Сравнить с эталоном** позволяет выбрать эталонное событие для сравнения и подсветить отличия в полях других событий.

  Кнопка позволяет настроить способ сравнения событий с эталоном. Кнопка доступна только при активном переключателе **Сравнить с эталоном**.

- Выпадающий список **Поля** предназначен для выбора конкретных полей событий для сравнения.

В рабочей области расположена таблица сравнения событий. Первый столбец таблицы содержит ключи полей событий. В остальных столбцах отображаются данные этих полей для каждого из сравниваемых событий.

В заголовке столбца каждого события расположена кнопка **Удалить из сравнения** () для удаления события из сравнения.

Настройка сравнения полей событий

Чтобы настроить сравнение полей событий:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.

   Кнопка доступна только при наличии событий, добавленных в сравнение.

3. При необходимости выберите из выпадающего списка **Поля** на панели инструментов поля событий, которые требуется сравнить. Если поля не выбраны, сравнение проводится для всех полей событий.
4. При необходимости задайте настройки отображения полей событий с помощью переключателей на панели инструментов:
   - **Показать отличия** — включите опцию, если требуется отобразить только те поля событий, в которых имеются отличия.
   - **Скрыть пустые поля** — включите опцию, если требуется скрыть все поля событий, не имеющих значений ни в одном из событий.
   - **Сравнить с эталоном** — включите опцию, если требуется сравнить события с эталоном и подсветить в них соответствующие отличия.

     По умолчанию в качестве эталона выбирается первое событие в списке сравниваемых событий. При необходимости вы можете назначить эталоном другое событие, нажав на кнопку в его заголовке.

     Чтобы настроить способ сравнения событий с эталоном, нажмите на кнопку на панели инструментов и выберите вариант сравнения:

     - **По символам** — значения полей сравниваются посимвольно.
     - **По словам** — значения полей сравниваются по последовательностям символов. В качестве разделителей последовательностей выступают специальные символы.

- **По полям** — значения полей сравниваются целиком.

Подсветка отличий происходит следующим образом:

- Красным цветом подсвечиваются сведения, присутствующие в эталоне, но отсутствующие в сравниваемом событии.
- Зеленым цветом подсвечиваются сведения, отсутствующие в эталоне, но присутствующие в сравниваемом событии.
- Сведения, совпадающие и в эталоне, и в сравниваемом событии, не подсвечиваются.

Копирование событий в сравнении

Чтобы скопировать данные всех сравниваемых событий в буфер обмена:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.



Кнопка доступна только при наличии событий, добавленных в сравнение.

3. Нажмите на кнопку на панели инструментов страницы сравнения событий. Система сохранит данные событий из сравнения в буфер обмена и отобразит соответствующее уведомление.

Удаление события из сравнения

Чтобы удалить событие из сравнения:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.



Кнопка доступна только при наличии событий, добавленных в сравнение.

3. Нажмите на кнопку в заголовке нужного события. Система удалит выбранное событие из сравнения и отобразит соответствующее уведомление. Событие перестанет отображаться на странице сравнения.



Вы также можете удалить событие из сравнения из его карточки. Для этого:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
3. Нажмите на кнопку в верхней части карточки события. Кнопка станет неактивной, система удалит выбранное событие из сравнения и отобразит соответствующее уведомление. Событие перестанет отображаться на странице

сравнения.

Вы также можете удалить сразу все события из сравнения. Для этого:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.

   Кнопка доступна только при наличии событий, добавленных в сравнение.
3. Нажмите на кнопку на панели инструментов страницы сравнения событий. Система отобразит окно подтверждения действия.
4. Нажмите на кнопку **Удалить**. Система удалит все события из сравнения и отобразит соответствующее уведомление.

Работа с избранными событиями

При расследовании инцидентов может возникнуть необходимость быстрого доступа к важным событиям. Для этого можно использовать панель избранных событий.

Панель избранных событий доступна только при поиске по хранилищам событий. Если для поиска выбран активный список, панель избранных событий недоступна.

Доступные операции над избранными событиями:

- Добавление события в список избранных
- Просмотр списка избранных событий
- Просмотр избранных событий в поиске
- Удаление событий из списка избранных

Добавление события в список избранных

Чтобы добавить событие в список избранных:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на строку события в списке. Система отобразит в правой части экрана карточку этого события с подробной информацией о нем.
3. Выберите опцию **Добавить в избранное** в выпадающем меню кнопки действий в верхней части карточки. Система добавит событие в список избранных и отобразит соответствующее уведомление.

Список избранных событий сохраняется индивидуально для текущего пользователя системы в рамках конкретного хранилища.

Просмотр списка избранных событий

Чтобы просмотреть список избранных событий:

1. Перейдите в раздел **Поиск**.
2. Выберите хранилище событий из выпадающего списка в верхней части раздела.
3. Нажмите на кнопку ☆ на панели инструментов раздела. Система отобразит панель избранных событий для текущего хранилища.

На панели отображается таблица избранных событий текущего хранилища. Таблица содержит следующие столбцы:

- **Время** — временная метка события. Извлекается из поля события `timestamp`.
- **Информация о событии** — краткие сведения о событии: значения первых нескольких заполненных полей в соответствии с порядком их хранения в событии.

Вы можете просмотреть все заполненные поля события с помощью кнопки ⌄ и свернуть их с помощью кнопки ⌃ в левом столбце строки события.



Вы можете раскрыть содержимое заполненных полей сразу всех избранных событий с помощью кнопки ⌄ и свернуть его с помощью кнопки ⌃ в заголовке левого столбца таблицы.

В нижней части таблицы отображаются следующие компоненты:

- Поле **Всего записей** показывает количество избранных событий.
- Кнопка **Посмотреть в поиске** (🔍) позволяет просмотреть избранные события в таблице на вкладке **Данные** раздела **Поиск**.
- Кнопка **Очистить избранное** (🗑) позволяет удалить все события из списка избранных.

Просмотр избранных событий в поиске

Чтобы просмотреть список избранных событий в таблице раздела **Поиск**:

1. Перейдите в раздел **Поиск**.
2. Выберите хранилище событий из выпадающего списка в верхней части раздела.
3. Нажмите на кнопку ☆ на панели инструментов раздела. Система отобразит панель избранных событий для текущего хранилища.
4. Нажмите на кнопку **Посмотреть в поиске** (🔍) в нижней части панели. Система осуществит поиск по текущему хранилищу и отобразит избранные события в таблице на вкладке **Данные** раздела **Поиск**.

Удаление событий из списка избранных

Чтобы удалить событие из списка избранных:

1. Перейдите в раздел **Поиск**.
2. Выберите хранилище событий из выпадающего списка в верхней части раздела.

3. Нажмите на кнопку ☆ на панели инструментов раздела. Система отобразит панель избранных событий для текущего хранилища.
4. Нажмите на кнопку 🗑 в правом столбце строки события, которое вы хотите удалить. Система удалит событие из списка избранных и отобразит соответствующее уведомление.



Вы также можете удалить событие из списка избранных из его карточки. Для этого:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на строку события в списке. Система отобразит в правой части экрана карточку этого события с подробной информацией о нем.
3. Выберите опцию **Удалить из избранного** в выпадающем меню кнопки действий ⋮ в верхней части карточки. Система удалит событие из списка избранных и отобразит соответствующее уведомление.

Вы также можете удалить сразу все события из списка избранных. Для этого:

1. Перейдите в раздел **Поиск**.
2. Выберите хранилище событий из выпадающего списка в верхней части раздела.
3. Нажмите на кнопку ☆ на панели инструментов раздела. Система отобразит панель избранных событий для текущего хранилища.
4. Нажмите на кнопку **Очистить избранное** (🗑) в нижней части панели.
5. Нажмите на кнопку **Очистить**. Система удалит все события из списка избранных и отобразит соответствующее уведомление.

Добавление исключений в черный и белый списки

Чтобы добавить исключение для коррелятора на основе события:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях.
2. Нажмите на строку события в списке. Система отобразит в правой части экрана карточку этого события с подробной информацией о нем.
3. Выберите опцию **Добавить в исключения** в выпадающем меню кнопки действий ⋮ в верхней части карточки. Откроется окно добавления исключения.



Опция доступна только для корреляционных событий.

4. Установите флажки у полей, которые нужно включить в условие исключения и нажмите на кнопку **Далее**.
5. Выберите список исключений, в который нужно добавить запись: черный или белый список.
6. При необходимости укажите дату и время, когда запись должна быть удалена из списка.



Чтобы запись оставалась в списке исключений бессрочно, не заполняйте поле **TTL**.

7. Поле ввода условия на языке VRL автоматически заполняется условиями для выбранных полей с оператором `=` и заданными значениями. Условия для отдельных полей объединяются оператором `AND`. При необходимости измените условия.

   В условии недоступно использование активных списков, таблиц обогащения, глобальных функций и переменных.

8. Нажмите на кнопку **Добавить**. Система проверит условие и добавит исключение в выбранный активный список.

Ресурсы

В данном разделе описаны компоненты системы, которые обеспечивают настройку и управление различными функциями системы. Компоненты сгруппированы по их функциональным областям и включают коллекторы, модели данных, базы данных и интеграции с внешними системами.

Содержание раздела:

- Работа с коллекторами
  - Работа с конвейерами
    - Работа с точками входа
    - Работа с VRL-трансформациями
    - Работа с фильтрами
    - Работа с маршрутизаторами
    - Работа с коррелляторами
    - Работа с нормализаторами
    - Работа с агрегаторами
    - Работа с сервисами оповещений
    - Работа со сбором метрик
    - Работа с конечными точками
  - Работа с шинами
  - Работа с обогащениями
- Работа с моделями событий
  - Универсальная модель события
  - Универсальная модель события 2.0
  - Типы данных
  - Типы операций в событии
- Работа с базами данных событий
- Работа с хранилищами событий
- Работа с активными списками
- Работа с интеграциями
  - Интеграция с R-Vision SOAR
  - Почтовые интеграции
- Работа с драйверами БД
- Работа с секретами

Работа с коллекторами

Данный раздел описывает процесс работы с коллекторами в системе. Работа осуществляется в разделе **Ресурсы** → **Коллекторы** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О коллекторах

Коллектор — это компонент системы, который обеспечивает сбор событий из источников, их обработку и дальнейшую отправку на хранение и/или во внешние системы. События поступают в коллектор через установленные на нем конвейеры, посредством которых они могут быть обработаны.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый коллектор.
- Кнопка ⤓ позволяет импортировать коллекторы в систему.
- Кнопка ⤒ позволяет экспортировать выбранные коллекторы из системы в формате JSON.
- Поле **Поиск** предназначено для быстрого поиска коллекторов в отображаемом списке по названию.
- Выпадающий список **Пространство** позволяет настроить отображение списка коллекторов в соответствии с выбранным подключением к пространству.
- Выпадающий список **Статус** позволяет настроить отображение списка коллекторов в соответствии с выбранным статусом.

В рабочей области отображается таблица имеющихся коллекторов. Таблица содержит следующие столбцы:

- **Название** — название коллектора, используемое для его идентификации в системе.
- **Статус** — текущий статус коллектора: **Включен**, **Выключен** или **Ошибка**.
- **Пространство** — название подключения к пространству, в котором расположен коллектор.
- **Конвейеров** — количество конвейеров в коллекторе.
- **Дата создания** — дата и время создания коллектора.
- **Дата изменения** — дата и время последнего изменения коллектора.
- **Описание** — описание коллектора.

При работе с таблицей коллекторов доступны следующие операции:

- Поиск коллекторов по полю **Название**.
- Фильтрация коллекторов по полям **Пространство** и **Статус**. Пространство и статус выбираются

из выпадающего списка.

- Сортировка коллекторов по полям **Название**, **Пространство**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного коллектора в правой части рабочей области отображается его карточка с детальной информацией.

Работа с коллектором

Доступные операции над коллекторами:

- Создание коллектора
- Просмотр коллектора
- Включение коллектора
- Выключение коллектора
- Изменение коллектора
- Удаление коллектора
- Импорт коллектора
- Экспорт коллектора

Создание коллектора

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором будет создан коллектор. По умолчанию выбрано подключение к центральному пространству.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.



- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов на каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.

7. Для работы в режиме распределенной корреляции установите переключатель **Распределенный коррелятор** в активное положение.



Изменение этой настройки потребует перезапуска коллектора.



Если система работает на базе кластера Kubernetes с одним узлом, включение режима распределенной корреляции недоступно.

8. Укажите количество ядер процессора (**CPU**) и объем оперативной памяти (**RAM**), выделяемый для работы распределенного коррелятора. Это можно сделать непосредственно в поле или с помощью кнопок + и —.



Операции полей `join` и `group` декларативного правила вычисляются на коррелляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. В составе распределенного коррелятора может быть только один коррелятор, но несколько предпроцессоров.

   a. Укажите ресурсы для коррелятора:

      i. Выберите узел для размещения коррелятора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы коррелятора. Количество доступных на узле ресурсов отображается под полями ввода.

   b. Укажите ресурсы для предпроцессора:

      i. Выберите узел для размещения предпроцессора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы предпроцессора. Количество доступных на узле ресурсов отображается под полями ввода.

9. Нажмите на кнопку **Добавить предпроцессор**, если необходимо добавить дополнительный предпроцессор.

10. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:

   - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
   - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
   - **Info** — записывается общая информация о работе и состоянии коллектора.
   - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
   - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

11. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее

уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Просмотр коллектора

Чтобы просмотреть коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.

В верхней части карточки отображается название коллектора. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранного коллектора.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка коллектора содержит следующие вкладки:

- **Информация** — общая информация о коллекторе и настройках логирования.
- **Конвейеры** — информация о конвейерах в коллекторе.
- **Шины** — информация о шинах в коллекторе.
- **Обогащение** — информация о таблицах обогащения, добавленных в коллектор.

Вкладка Информация

На вкладке **Информация** отображаются основные сведения о коллекторе, сгруппированные по следующим секциям:

- **Метрики коллектора**:
  - **CPU** — количество ядер и уровень загруженности процессора сервиса.
  - **RAM** — объем и уровень загруженности оперативной памяти сервиса.
  - **HDD** — объем и уровень загруженности жесткого диска сервиса.
- **Метрики распределенного коррелятора** — метрики по каждому узлу распределенного коррелятора, включая данные по каждому предпроцессору и коррелятору:
  - **CPU** — количество ядер и уровень загруженности процессора на узле.
  - **RAM** — объем и уровень загруженности оперативной памяти на узле.
  - **HDD** — объем и уровень загруженности жесткого диска на узле.
- **Информация**:
  - **ID** — уникальный идентификационный код коллектора, генерируемый автоматически при его создании.
  - **Название** — имя коллектора, используемое для его идентификации в системе.
  - **Пространство** — название подключения к пространству, в котором расположен коллектор.

Чтобы получить подробные сведения о подключении, нажмите на его название. Система откроет раздел **Настройки → Пространства и сервисы → Пространства** и отобразит на экране подробную карточку этого подключения.

- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится коллектор.
- **Статус** — текущий статус коллектора: **Включен**, **Выключен** или **Ошибка**.
- **IP-адрес** — IP-адрес, с которым связан коллектор.
- **Имя хоста** — имя хоста, с которым связан коллектор.
- **Описание** — краткое описание функциональности коллектора.
- **Конвейеров** — количество конвейеров, связанных с коллектором.
- **Дата создания** — дата и время создания коллектора.
- **Создал** — пользователь, инициировавший создание коллектора.
- **Дата изменения** — дата и время последнего изменения коллектора.
- **Изменил** — пользователь, выполнивший последнее изменение коллектора.

- **Просмотр логов** — информация о логировании в коллекторе.

  Содержит поле **Уровень логирования** — уровень логирования коллектора: **Error**, **Warn**, **Info**, **Debug**, **Trace** или **По умолчанию**.

  В секции также расположена кнопка **Просмотреть логи**, позволяющая открыть окно для просмотра логов коллектора.

- **EPS по точкам входа за последние 30 минут** — график с количеством событий в секунду (EPS), полученных во всех точках входа в коллекторе за последние 30 минут.
- **Входной EPS корреляторов за последние 30 минут** — график с количеством событий в секунду (EPS), полученных на входе всех коррелляторов в коллекторе за последние 30 минут.
- **Усредненный EPS по точкам входа за последние 30 минут** — график с усредненным количеством событий в секунду (EPS) во всех точках входа в коллекторе за последние 30 минут.
- **Усредненный входной EPS корреляторов за последние 30 минут** — график с усредненным количеством событий в секунду (EPS) на входе всех коррелляторов в коллекторе за последние 30 минут.

- Если при получении данных по одной из метрик коллектора или распределенного коррелятора произошла ошибка, для нее отобразится текст "N/A".
- Если при получении данных по одному из графиков произошла ошибка, для него отобразится текст "Не удалось запросить данные для графика".

▼ *Особенности расчета EPS на графиках в коллекторе*

Временной шаг между точками на каждом графике составляет 15 секунд.

Расчет показателей EPS в каждой точке осуществляется по следующей формуле: $(K_1 – K_2)/T$.

Здесь:

- **K_1** — количество событий в момент времени текущей точки.
- **K_2** — количество событий в момент времени текущей точки минус **T**.
- **T** — временной диапазон расчета.

Таким образом, графики отличаются между собой источником событий, участвующих в расчетах, и значением **T**:

График	Источники событий	Временной диапазон T
EPS по точкам входа за последние 30 минут	Все точки входа в коллекторе	15 секунд
Входной EPS корреляторов за последние 30 минут	Входы всех корреляторов в коллекторе	15 секунд
Усредненный EPS по точкам входа за последние 30 минут	Все точки входа в коллекторе	30 минут
Усредненный входной EPS корреляторов за последние 30 минут	Входы всех корреляторов в коллекторе	30 минут



- Подписи по оси X на каждом графике отображаются с шагом в 5 минут.
- Для каждого графика отображаются две линии:
  - **Синяя** — сведения по текущему коллектору.
  - **Красная** — суммарные сведения по всем коллекторам в системе.



Чтобы обновить графики, перезагрузите страницу.

В верхней части карточки также расположен переключатель статуса коллектора.

Вкладка Конвейеры

На вкладке **Конвейеры** отображается список конвейеров, добавленных в коллектор.

По умолчанию для каждого конвейера отображается только его название и цветовой индикатор статуса конфигурации:

- зеленый — конфигурация установлена;
- серый — конфигурация не установлена;
- красный — ошибка установки.

Также в строке конвейера имеется кнопка действий ⋮. При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного конвейера.

Чтобы получить более подробные сведения о конвейере, нажмите на кнопку ⌄ в правой части его строки. Для конвейера отобразится карточка со следующей информацией:

- **ID** — уникальный идентификационный код конвейера, генерируемый автоматически при его

создании.

- **Описание** — краткое описание функциональности конвейера.
- **Статус конфигурации** — текущий статус конфигурации конвейера: **Установлена**, **Не установлена** или **Ошибка установки**.
- **Дата создания** — дата и время создания конвейера.
- **Создал** — пользователь, инициировавший создание конвейера.
- **Дата изменения** — дата и время последнего изменения конвейера.
- **Изменил** — пользователь, выполнивший последнее изменение конвейера.
- **EPS за последние 30 минут** — график с количеством полученных событий в секунду (EPS) за последние 30 минут с шагом в 5 минут.
- **Адреса точек входа** — адреса всех точек входа, добавленных на конвейер.

  Для каждой точки входа в правой части строки расположена кнопка  . При нажатии на эту кнопку система скопирует адрес точки входа в буфер обмена и отобразит соответствующее уведомление.

В карточке конвейера также расположены кнопки для настройки и установки или отключения его конфигурации.

В верхней части вкладки содержится поле для быстрого поиска конвейеров по названию.

В нижней части вкладки расположены кнопки для добавления и импорта конвейеров.

Вкладка Шины

На вкладке **Шины** отображается список шин в коллекторе.

По умолчанию для каждой шины отображается только ее название. Также в строке шины имеется кнопка действий ⋮. При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранной шины.

Чтобы получить более подробные сведения о шине, нажмите на кнопку ⌄ в правой части ее строки. Для шины отобразится карточка со следующей информацией:

- **ID** — уникальный идентификационный код шины, генерируемый автоматически при ее создании.
- **Описание** — краткое описание функциональности шины.
- **Дата создания** — дата и время создания шины.
- **Создал** — пользователь, инициировавший создание шины.
- **Дата изменения** — дата и время последнего изменения шины.
- **Изменил** — пользователь, выполнивший последнее изменение шины.

В верхней части вкладки содержится поле для быстрого поиска шин по названию.

В нижней части вкладки расположена кнопка для добавления шин.

Вкладка Обогащение

На вкладке **Обогащение** отображается список обогащений в коллекторе.

По умолчанию для каждого обогащения отображается только его название. Также в строке обогащения имеется кнопка действий ⋮. При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного обогащения.

Чтобы получить более подробные сведения об обогащении, нажмите на кнопку ⌄ в правой части его строки. Для обогащения отобразится карточка со следующей информацией:

- **ID обогащения** — уникальный идентификационный код компонента обогащения, генерируемый автоматически при добавлении его в коллектор.
- **Описание** — краткое описание функциональности обогащения.
- **ID таблицы обогащения** — уникальный идентификационный код таблицы обогащения, которая была использована при создании обогащения.
- **Таблица обогащения** — название таблицы обогащения, которая была использована при создании обогащения.
- **Версия таблицы обогащения** — версия таблицы обогащения, указанная в поле `version` в настройках таблицы.
- **Дата создания** — дата и время создания обогащения.
- **Создал** — пользователь, инициировавший создание обогащения.
- **Дата изменения** — дата и время последнего изменения обогащения.
- **Изменил** — пользователь, выполнивший последнее изменение обогащения.

В верхней части вкладки содержится поле для быстрого поиска обогащений по названию.

В нижней части вкладки расположена кнопка для добавления обогащений.

Включение коллектора

Для того чтобы коллектор мог начать работу с событиями, поступающими в систему, его нужно включить.



Вместе с коллектором также выключаются и все связанные с ним конвейеры.

Чтобы включить коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. Установите переключатель состояния коллектора в верхней части карточки в активное положение. Отобразится окно подтверждения включения коллектора.
5. Нажмите на кнопку **Включить**. Система включит коллектор и отобразит соответствующее

уведомление.

Если при включении коллектора возникли ошибки, система отобразит сообщение об этом. Для просмотра списка ошибок нажмите на кнопку **Показать**.

Выключение коллектора

Чтобы выключить работающий коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. Установите переключатель в верхней части коллектора в неактивное положение. Отобразится окно подтверждения выключения коллектора.
5. Нажмите на кнопку **Выключить**. Система выключит коллектор и отобразит соответствующее уведомление.

Изменение коллектора

Вы можете изменить коллектор следующими способами:

- через меню действий в карточке коллектора;
- через изменение полей в карточке коллектора.



- Изменить подключение к пространству можно, только если коллектор выключен.
- Если было изменено подключение к пространству, то при сохранении изменений отобразится окно подтверждения. После подтверждения все конфигурации конвейеров в коллекторе отключатся и перейдут в режим черновика. Чтобы возобновить их работу, необходимо заново вручную установить конфигурации конвейеров.

Удаление коллектора

Вы можете удалить коллектор через меню действий в его карточке.



- Удаление недоступно для коллекторов, используемых в других сущностях системы.
- Вместе с коллектором также удаляются и все добавленные в него конвейеры, шины и обогащения.

Импорт коллектора

Для загрузки конфигурации коллектора в систему применяется функционал импорта, поддерживающий обработку файлов в формате JSON.

Перед тем как импортировать коллектор:

1. Создайте или импортируйте в систему хранилище событий.
2. Импортируйте в систему все элементы экспертизы, которые содержит импортируемый коллектор.

Чтобы импортировать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку ⤓ на панели инструментов. Система отобразит окно импорта коллектора.
3. Нажмите **Выбрать файл** (⊕) и выберите JSON-файл для импорта. Файл отобразится в окне импорта коллектора.
4. Выберите подключение к пространству, в котором будет создан коллектор. По умолчанию выбрано подключение к центральному пространству.

   Если в JSON-файле коллектора указан параметр `spaceId` с ID пространства, он автоматически будет заменен на ID выбранного пространства.

5. Нажмите на кнопку **Импортировать**. Коллектор будет загружен в систему. Система отобразит окно с итогами импорта.

   Если в нормализаторе конвейера был включен аудит источников, то при импорте конвейера применяется политика аудита, установленная в системе по умолчанию.

Экспорт коллектора

Чтобы экспортировать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Экспортировать**. Отобразится окно экспорта данных.
4. При необходимости установите флажок **Экспортировать без изменяемых параметров**, чтобы экспортировать конвейеры с пустыми значениями изменяемых параметров.

   ▼ *Параметры конвейера, которые можно экспортировать с пустыми значениями*

   - `databaseDriver` — драйвер базы данных;
   - `credentials` — данные базовой аутентификации;
   - `token` — токен для аутентификации;
   - `storageId` — ID конечной точки хранилища событий;
   - `eventStorageId` — ID сервиса оповещений;

- `eventBusId` — ID шины;
- `monitoringPolicyId` — ID политики аудита.

5. Нажмите на кнопку **Экспортировать**. Система загрузит данные коллектора на устройство пользователя в формате JSON и отобразит соответствующее уведомление.

Вы также можете экспортировать сразу несколько коллекторов. Для этого:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Установите флажки напротив тех коллекторов, которые вы хотите экспортировать, в левом столбце таблицы.
3. Нажмите на кнопку ↑ на панели инструментов. Система запросит подтверждение выполнения экспорта.
4. Нажмите на кнопку **Экспортировать**. Система загрузит данные коллекторов на устройство пользователя в формате JSON и отобразит соответствующее уведомление.



Также можно экспортировать конфигурацию коллектора из объектного хранилища NATS ObjectStore.

Работа с логами коллектора

Логирование коллекторов обеспечивает запись событий и состояний системы сбора данных, что позволяет администраторам и разработчикам отслеживать и анализировать работу системы. Уровень логирования настраивается вручную при создании коллектора или изменении его параметров, что позволяет гибко управлять объемом и детализацией регистрируемых данных. Доступные уровни логирования включают в себя Error, Warn, Info, Debug и Trace, каждый из которых предоставляет различный уровень информации о работе коллектора.

Функционал просмотра логов доступен через пользовательский интерфейс на вкладке **Информация** карточки коллектора. Здесь отображается текущий уровень логирования и предоставляется возможность просмотра самих логов.

Доступные операции над логами коллектора:

- Просмотр логов коллектора
- Фильтрация списка логов коллектора
- Копирование логов коллектора
- Изменение уровня логирования



Логи коллектора очищаются при его выключении.

Просмотр логов коллектора

Чтобы просмотреть логи коллектора:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.

2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. В секции **Просмотр логов** система отображает текущий уровень логирования коллектора. Нажмите на кнопку **Просмотреть логи**. Система отобразит окно со списком логов коллектора.



В списке логов доступен просмотр только последних 100 000 записей.

В левом верхнем углу окна отображается ID просматриваемого коллектора.

Доступны два режима вывода логов:

- **Режим мониторинга** — вывод логов в реальном времени.

  Если в течение заданного интервала после проверки какой-либо точки входа не появились новые логи, система выдаст сообщение уровня Info о неактивности источника. Если источник остается неактивным, система будет повторно логировать его с тем же интервалом, обновляя длительность простоя в сообщении.



Для точек входа типов Database, HTTP Client и WMI интервал указывается в настройках в поле **Интервал запроса, секунд**.

Для остальных точек входа интервал равен 10 секундам.

- **Режим поиска по временному периоду** — вывод логов в рамках заданного периода. Максимальный период запроса — 30 дней.

Переключатель **Мониторинг** позволяет управлять режимом вывода логов.



Время событий отображается в логах в формате `YYYY-MM-DDThh:mm:ss.sssZ` независимо от настроек интерфейса.



- Логи коллектора можно также просмотреть:
  - при нажатии на кнопку **Просмотреть логи** () на странице конфигурации конвейера;
  - при нажатии на кнопку **Посмотреть логи** в карточке сервиса;
  - в кластере Kubernetes.
- Если ошибки в коллекторе не удается устранить с помощью анализа его логов, рекомендуется проанализировать логи пода `collector-manager`, управляющего работой всех коллекторов в системе.

Фильтрация списка логов коллектора



Просмотр логов за конкретный временной интервал недоступен в режиме мониторинга.

Чтобы отфильтровать список логов коллектора в режиме просмотра логов:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. В секции **Просмотр логов** система отображает текущий уровень логирования коллектора. Нажмите на кнопку **Просмотреть логи**. Система отобразит окно со списком логов коллектора.
5. Установите переключатель **Мониторинг** в неактивное положение. Настройки фильтрации станут доступными для редактирования.
6. Настройте период поиска логов. Для этого нажмите на кнопку и выберите из выпадающего списка способ задания периода:
   - **Последний период** — задайте значение вручную или с помощью кнопок — и +, после чего выберите единицы времени из выпадающего списка. Поиск логов будет выполняться в указанном диапазоне вплоть до нынешнего момента.
   - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск логов.
   - **Часто используемые** — выберите в разделе один из предлагаемых периодов.
7. Нажмите на кнопку **Найти**. Список логов, соответствующий заданному периоду, отобразится в окне.

Копирование логов коллектора

Чтобы скопировать записи журнала логов коллектора:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. В секции **Просмотр логов** система отображает текущий уровень логирования коллектора. Нажмите на кнопку **Просмотреть логи**. Система отобразит окно со списком логов коллектора.
5. Нажмите на кнопку **Скопировать** в правой нижней части экрана. Система скопирует записи журнала в буфер обмена и отобразит соответствующее уведомление.

Изменение уровня логирования

Изменение уровня логирования выполняется через редактирование настроек коллектора.

Чтобы изменить уровень логирования коллектора:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.

3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек коллектора.
4. Выберите значение из выпадающего списка **Уровень логирования**. Доступны следующие значения:
    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке коллектора.

Работа с конвейерами

Данный раздел описывает процесс работы с конвейерами обработки событий в системе. Работа осуществляется на вкладке **Конвейеры** в карточке коллектора.

О конвейерах

Конвейер — компонент коллектора, представляющий собой упорядоченную последовательность взаимосвязанных элементов, выполняющих различные этапы обработки событий. В процессе работы конвейера происходит прием событий, их нормализация, перенаправление, обогащение данными, фильтрация, агрегация, корреляционный анализ и отправка как отдельных, так и сгруппированных или коррелированных событий для хранения и/или передачи во внешние системы.

Список конвейеров коллектора отображается на вкладке **Конвейеры** в его карточке. Над списком расположено поле для поиска конвейеров по названию.

Доступные операции над конвейером:

- Добавление конвейера
- Просмотр конвейера
- Настройка конфигурации конвейера
- Установка конфигурации конвейера
- Изменение конвейера
- Создание копии конвейера
- Удаление конвейера
- Импорт конвейера
- Экспорт конвейера

Добавление конвейера

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.



Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Просмотр конвейера

Чтобы отобразить информацию о конвейере:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку ⌄ в строке конвейера. Отобразится карточка конвейера.

В верхней части карточки отображается название конвейера. Слева от названия расположен цветовой индикатор статуса конфигурации конвейера:

- зеленый — конфигурация установлена;
- серый — конфигурация не установлена.

Справа от названия конвейера расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранного конвейера.

Карточка конвейера содержит следующую информацию:

- **ID** — уникальный идентификационный код конвейера, генерируемый автоматически при его создании.
- **Описание** — краткое описание функциональности конвейера.

- **Статус конфигурации** — текущий статус конфигурации конвейера: **Установлена** или **Не установлена**.
- **Дата создания** — дата и время создания конвейера.
- **Создал** — пользователь, инициировавший создание конвейера.
- **Дата изменения** — дата и время последнего изменения конвейера.
- **Изменил** — пользователь, выполнивший последнее изменение конвейера.
- **EPS за последние 30 минут** — график с количеством зарегистрированных событий в секунду (EPS) за последние 30 минут с шагом в 5 минут.



Если при получении данных по графику произошла ошибка, для него отобразится текст "Не удалось запросить данные для графика".

- **Адреса точек входа** — адреса всех точек входа, добавленных на конвейер.

  Для каждой точки входа в правой части строки расположена кнопка  . При нажатии на эту кнопку система скопирует адрес точки входа в буфер обмена и отобразит соответствующее уведомление.

В карточке конвейера также расположены кнопки для настройки и установки или отключения его конфигурации.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Для базовой обработки и хранения данных в системе достаточно настроить точку входа и конечную точку. Дополнительные элементы конвейера, такие как фильтр, нормализатор и коррелятор, позволяют настроить более сложную логику обработки данных, проходящих через конвейер событий.

Эти элементы можно разместить между точкой входа и конечной точкой в любой необходимой последовательности и комбинации. При этом можно использовать несколько элементов одного типа для более детализированной настройки обработки.

Для управления порядком и логикой прохождения данных через различные этапы конвейера можно применять маршрутизатор. Он позволяет перенаправлять и распределять поток данных в соответствии с заранее заданными условиями.

Чтобы настроить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку  в строке конвейера. Отобразится карточка конвейера.

5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации, над которой расположены кнопки установки или отключения конфигурации конвейера.



- Диаграмму конфигурации также можно открыть, нажав в строке конвейера в карточке коллектора на кнопку действий ⋮ и выбрав опцию **Конфигурация конвейера**.
- По умолчанию справа от переключателя состояния конвейера система показывает уведомление о трех выявленных ошибках из-за отсутствия элементов на конвейере. Для просмотра всех ошибок нажмите **Показать**.

6. Настройте конфигурацию конвейера:
   a. Создайте и настройте элементы конвейера.
   b. Свяжите элементы конвейера в определенной последовательности согласно желаемой логике получения, обработки и передачи данных.
7. Установите конфигурацию конвейера.



Если при настройке конвейера были допущены ошибки, система отобразит сообщение об этом справа от кнопки установки или отключения конфигурации конвейера. Для просмотра списка ошибок нажмите на кнопку **Показать**.

Интерфейс конфигурации конвейера

Редактирование конфигурации происходит на визуальной диаграмме через добавление элементов и связей между ними. Каждый элемент конвейера отвечает за конкретный этап обработки событий.

На диаграмме отображаются типы и названия элементов, а также дополнительная информация, которая зависит от типа элемента:

- **Точка входа**:
  - тип точки;
  - адрес подключения (при наличии);
  - порт подключения (при наличии);
  - тип протокола (при наличии);
  - уровни угрозы (для точки входа типа Audit);
  - статус аудита: включен или выключен.
- **Коррелятор**: количество правил корреляции.
- **Нормализатор**:
  - количество правил нормализации;
  - статус аудита: включен или выключен.
- **Агрегатор**: количество правил агрегации.
- **Сервис оповещений**:
  - название хранилища;

    - название интеграции или количество интеграций, если их несколько;
    - количество правил сегментации (включая правило по умолчанию).
- **Шина**: тип соединения с шиной (**Получение** или **Отправка**).
- **Конечная точка**: тип точки, а также, в зависимости от типа:
    - для конечной точки **Хранилище событий**: название хранилища;
    - для конечной точки **Внешняя система**: тип внешней системы и ее адрес (при наличии).

Если информация сокращена, при наведении курсора на элемент отображается всплывающее сообщение с полным текстом.

Каждый элемент имеет хотя бы один вход или выход, при этом вход может быть только один, а выходов может быть несколько. На диаграмме конфигурации входы и выходы элемента обозначены метками на его боковых сторонах. Вход соответствует метке на левой стороне элемента, а выходы — метке на правой стороне.

Элементы конвейера соединены между собой направляющими линиями, которые указывают порядок и направление обработки событий. Элементы можно располагать между точкой входа и конечной точкой или шиной в любой желаемой последовательности и комбинации. Например, можно использовать несколько элементов одного типа.

Доступны два режима работы с конфигурацией конвейера:

- **Актуальная версия** — доступен только просмотр конфигурации конвейера.
- **Черновик** — доступно редактирование конфигурации конвейера.



Редактирование конфигурации в режиме черновика доступно даже при установленной конфигурации.

Чтобы установить режим работы с конфигурацией конвейера, выберите его из выпадающего списка **Версия**, который расположен в правой части панели инструментов.

В режиме черновика на панели инструментов доступны следующие кнопки:

- **Добавить элемент** — позволяет добавить новый элемент конвейера.
- **Сбросить изменения** () — позволяет сбросить изменения в черновике до сохраненной ранее конфигурации. Если изменений в черновике не было, кнопка не активна.
- **Отцентровать конвейер** () — позволяет автоматически отмасштабировать и отцентровать диаграмму без упорядочивания ее элементов.
- **Упорядочить элементы** () — позволяет автоматически упорядочить элементы, а также отмасштабировать и отцентровать диаграмму по ее элементам.
- **Просмотреть логи** () — позволяет просмотреть логи коллектора.
- **Установить конфигурацию** — позволяет установить настроенную в черновике конфигурацию конвейера.
- **Отключить конфигурацию** — позволяет отключить текущую установленную конфигурацию конвейера.

В режиме актуальной версии на панели инструментов доступны следующие кнопки:

- **Посмотреть логи** () — позволяет просмотреть логи коллектора.
- **Сбросить метрики** () — позволяет сбросить текущие метрики у элементов конвейера.
- **Отключить конфигурацию** — позволяет отключить текущую установленную конфигурацию конвейера.



- Чтобы приблизить или отдалить элементы на диаграмме, используйте колесико мыши.
- Чтобы перемещаться по диаграмме, зажмите кнопку мыши и перемещайте курсор в нужном направлении.

Также в режиме актуальной версии в нижней части каждого элемента конвейера отображаются следующие метрики:

- **Получено событий** () — количество событий, полученных элементом за последние 30 минут;
- **Отправлено событий** () — количество событий, отправленных элементом за последние 30 минут;
- **Ошибки** () — количество ошибок на элементе за последние 30 минут.

Дополнительные метрики можно посмотреть на вкладке **Метрики** окна просмотра элемента конвейера.



Между соседними элементами конвейера возможно небольшое расхождение метрик по отправленным и полученным событиям. Это обусловлено следующими особенностями:

- **Во время обработки событий** — расхождение происходит из-за особенностей регистрации метрик на элементах. При уменьшении потока событий метрики синхронизируются и данное расхождение устраняется.
- **После отключения конфигурации** — расхождение происходит из-за того, что на момент отключения конфигурации один элемент уже зафиксировал отправку события далее по цепочке, а другой еще не успел зафиксировать его принятие. Данное расхождение сохраняется на протяжении всей последующей работы конвейера, пока метрики не будут сброшены.

Добавление элемента конвейера

Каждый элемент конвейера соответствует определенному этапу конвейера обработки событий. Добавление новых элементов конвейера доступно только в режиме черновика.

Доступные типы элементов:

- **Точка входа** — имеет один выход и не имеет входов. Подключается к источнику данных и на выходе выдает поток исходных событий, поступающих из источника. Выбор типа точки входа зависит от формата получаемых данных.
- **VRL-трансформация** — имеет один вход и один выход. На входе принимает события и проводит их преобразование. На выходе выдает преобразованные события. Инструкция для

трансформации событий указывается с использованием языка VRL.

Рекомендуется для несложных модификаций в структуре событий, например:

- для использования одной или нескольких функций, упрощающих дальнейшую работу с событиями;
- для добавления, изменения или удаления поля в структуре.

- **Маршрутизатор** — имеет один вход и несколько выходов: по одному выходу на каждый фильтр (маршрут), задаваемый на языке VRL. На входе проверяет каждое событие по условиям маршрутов и направляет его на выходы тех маршрутов, где условие фильтрации выполняется.
- **Фильтр** — имеет один вход и один выход. Содержит условие фильтрации, задаваемое на языке VRL. На входе принимает событие, проверяет его по условию фильтрации и направляет на выход только в том случае, если условие выполняется. В противном случае событие отсеивается и не участвует в дальнейшей обработке.
- **Шина** — работает в рамках одного коллектора. В зависимости от типа соединения на конфигурации конвейера, имеет либо только один вход без выходов (получение), либо только один выход без входов (отправка). Принимает события на входе соединения типа **Получение** и пересылает их на выход соединения типа **Отправка**.

  Шины позволяют настроить обмен событиями между несколькими конвейерами.

- **Коррелятор** — имеет один вход и один выход. На входе принимает поток событий и применяет к ним правила корреляции, определяя взаимосвязанные события, которые соответствуют заданным критериям. На выходе выдает корреляционные события.
- **Нормализатор** — имеет один вход и один выход. На входе принимает поток событий и применяет к ним правила нормализации, осуществляя разбор и преобразование полученных полей события в нужную структуру. На выходе выдает нормализованные события.

  Рекомендуется для проведения сложных и объемных преобразований, применения одних и тех же преобразований в нескольких конвейерах и возможности изменения правил преобразований без перезапуска конвейера.

  Использование нормализаторов необходимо для приведения событий к структуре, ожидаемой конечной точкой. В противном случае возможны потери в потоке событий, если структура событий не соответствует требуемой структуре.

- **Сервис оповещений** — имеет один вход и не имеет выходов. Содержит настроенные связи между правилами сегментации и правилами корреляции. На входе принимает поток корреляционных событий и на основе заданных связей определяет, к какому оповещению должно быть отправлено то или иное корреляционное событие.

  Сервис оповещений недоступен для конвейеров в вынесенных коллекторах.

- **Конечная точка** — имеет один вход и не имеет выходов. На входе принимает событие и, в зависимости от заданных настроек, пересылает его в хранилище событий или во внешнюю систему. Выбор типа конечной точки зависит от пункта назначения, в который пересылаются события.

Если структура события отличается от структуры, ожидаемой выбранным типом конечной точки, то данное событие отбрасывается. Чтобы минимизировать потери в потоке событий, необходимо использовать нормализаторы для приведения событий к требуемой структуре.

- **Агрегатор** — имеет один вход и один выход. На входе принимает поток событий и применяет к ним правила агрегации, объединяя события в группы. На выходе выдает агрегированные события, которые представляют собой совокупности характеристик исходных событий, удовлетворяющих условиям агрегации.
- **Сбор метрик** — имеет один выход и не имеет входов. Собирает метрики по выбранным элементам конвейеров в рамках коллектора и на выходе выдает поток событий, содержащий собранные метрики.

Чтобы добавить элемент конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
2. Нажмите **Добавить элемент** над полем диаграммы.
3. Введите необходимые параметры и нажмите на кнопку **Добавить**. Новый элемент отобразится на диаграмме конфигурации.

Добавленные элементы не учитываются по умолчанию в текущей установленной конфигурации. Чтобы подгрузить изменения в установленную конфигурацию, ее требуется установить заново.

Настройка связей между элементами конвейера

Настройка связей между элементами конвейера доступна только в режиме черновика.

Чтобы связать элементы, наведите курсор на метку, расположенную на боковой стороне элемента. Курсор примет вид крестика. Перетащите крестик к метке на другом элементе. Связь будет создана, порядок обработки событий будет определен логикой связи.

Один вход может быть соединен с несколькими выходами, а один выход — с несколькими входами. При этом недопустимы циклические связи.

Чтобы удалить связь между элементами, выделите ее и нажмите клавишу DELETE. В Windows также можно использовать клавишу BACKSPACE.

Настроенные связи не учитываются по умолчанию в текущей установленной конфигурации. Чтобы подгрузить изменения в установленную конфигурацию, ее требуется установить заново.

Просмотр элемента конвейера

Просмотр элементов конвейера доступен только в режиме актуальной версии конфигурации.

Чтобы просмотреть элемент конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Актуальная**.
2. Нажмите на кнопку **Просмотр** () справа от названия элемента. Отобразится окно просмотра

элемента.

В окне доступны следующие вкладки:

- **Информация** — отображает заданные настройки элемента конвейера.
- **Метрики** — отображает сведения о полученных и отправленных событиях, об ошибках, а также о загруженности элемента в виде соответствующих графиков. Отображаются сведения за последние 30 минут. Шаг графика — 15 секунд.



Между соседними элементами конвейера возможно небольшое расхождение метрик по отправленным и полученным событиям. Это обусловлено следующими особенностями:

- **Во время обработки событий** — расхождение происходит из-за особенностей регистрации метрик на элементах. При уменьшении потока событий метрики синхронизируются и данное расхождение устраняется.
- **После отключения конфигурации** — расхождение происходит из-за того, что на момент отключения конфигурации один элемент уже зафиксировал отправку события далее по цепочке, а другой еще не успел зафиксировать его принятие. Данное расхождение сохраняется на протяжении всей последующей работы конвейера, пока метрики не будут сброшены.

Доступны следующие графики метрик:

- **Получено событий**;
- **Отправлено событий**;
- **Получено байт**;
- **Отправлено байт**;
- **Количество ошибок**;
- **Утилизация компонента**.



Метрика утилизации характеризует степень загруженности элемента. Она принимает значения от 0 до 1:

- чем ближе утилизация к 1, тем меньше время простоя элемента;
- если 0, то элемент не используется;
- если 1, то элемент никогда не простаивает.

Для элемента типа **Конечная точка** дополнительно доступны графики **Буферизировано событий** и **Буферизировано байт**, отображающие статистику буферизации.



Для нормализаторов и коррелляторов на вкладке **Информация** доступна кнопка **Изменить**, которая позволяет обновить заданные настройки.

Изменение элемента конвейера

Изменение элементов конвейера доступно только в режиме черновика.

Чтобы изменить элемент конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
2. Откройте окно настроек элемента одним из следующих способов:
   - Нажмите на кнопку действий ⋮ справа от названия элемента и выберите опцию **Изменить**.
   - Дважды нажмите на элемент на диаграмме конфигурации конвейера.
3. Измените требуемые настройки элемента.
4. Нажмите на кнопку **Сохранить**. Система обновит информацию об элементе.



Изменение нормализаторов и корреляторов доступно также и в режиме актуальной версии конфигурации конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Актуальная**.
2. Нажмите на кнопку **Просмотр** (👁) справа от названия элемента. Отобразится окно просмотра элемента.
3. Удостоверьтесь, что в окне просмотра элемента открыта вкладка **Информация**.
4. Нажмите на кнопку **Изменить**. Отобразится окно настройки элемента.
5. Измените требуемые настройки элемента.
6. Нажмите на кнопку **Сохранить**. Система обновит информацию об элементе.

Измененные настройки элементов не учитываются по умолчанию в текущей установленной конфигурации. Чтобы подгрузить изменения в установленную конфигурацию, ее требуется установить заново.

Удаление элемента конвейера

Удаление элементов конвейера доступно только в режиме черновика.

Чтобы удалить элемент конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
2. Нажмите на кнопку действий ⋮ справа от названия элемента и выберите опцию **Удалить**. Отобразится окно подтверждения удаления.
3. Нажмите на кнопку **Удалить**. Элемент конвейера будет удален.

Удаленные элементы не учитываются по умолчанию в текущей установленной конфигурации. Чтобы подгрузить изменения в установленную конфигурацию, ее требуется установить заново.

Валидация конвейера

При настройке конвейера система проверяет конфигурацию на наличие ошибок. Сообщение о количестве допущенных ошибок отображается справа от кнопки установки или отключения конфигурации конвейера. Чтобы просмотреть список ошибок, нажмите на кнопку **Показать**.

Могут возникать два типа ошибок: ошибки конфигурации конвейера в целом и ошибки отдельных элементов. Ошибки конвейера включают в себя:

- Отсутствие хотя бы одного источника событий: точки входа, шины на получение событий или сбора метрик.



Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

- Отсутствие хотя бы одной конечной точки, шины на отправку событий или сервиса оповещений.



Для отладки конвейера вы можете использовать конечную точку Blackhole, которая не сохраняет и не отправляет события.

Ошибки элементов связаны с некорректной настройкой конкретных элементов. Чтобы перейти к проблемному элементу, под описанием ошибки нажмите на кнопку **К элементу**. Ошибки элементов включают:

- Незаполненные обязательные поля элемента.



При возникновении данной ошибки вы можете открыть соответствующий элемент конвейера в режиме редактирования, нажав на кнопку **К элементу**. Все незаполненные обязательные поля будут подсвечены.

- Отсутствие связей на входе элемента.
- Отсутствие связей на выходе элемента.
- Невалидный код VRL.
- Неуникальное значение порта в точке входа.
- Правило, содержащее отсутствующий активный список.
- Правило, содержащее отсутствующую таблицу обогащения.
- Циклическая связь элементов. Ошибка возникает, если элемент связан сам с собой через другие элементы.
- Отсутствие настроенного потока событий на шине.
- Аудит источников включен на нескольких последовательных нормализаторах.
- Отсутствие необходимых ресурсов. Ошибка возникает, если в конфигурации элементе импортированного конвейера указаны ресурсы, которые отсутствуют в системе. Это могут быть:
  - Драйвер БД в точке входа Database;
  - Хранилище событий в конечной точке;
  - Секреты в точке входа или конечной точке.

Сброс метрик

Чтобы сбросить метрики элементов конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Текущая**.
2. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения сброса метрик.
3. Нажмите на кнопку **Сбросить**. Система сбросит текущие метрики у элементов конвейера и графики метрик по компонентам до 0 и отобразит соответствующее уведомление.



При сбросе метрик не обнуляются данные на графиках **EPS в точках входа за последние 30 минут** в карточке коллектора и **EPS за последние 30 минут** в карточке конвейера. На графиках отражается общий показатель.

Установка конфигурации конвейера

Установка конфигурации конвейера доступна только в режиме черновика.



Для того чтобы установить конфигурацию конвейера в коллекторе, убедитесь, что связанный с ним коллектор уже включен.

Чтобы установить конфигурацию конвейера в коллекторе:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации, над которой расположены кнопки установки или отключения конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
7. Нажмите на кнопку **Установить конфигурацию** над диаграммой конфигурации. Отобразится окно подтверждения установки конфигурации.
8. Нажмите на кнопку **Установить**. Система установит конфигурацию конвейера и отобразит соответствующее уведомление. События начнут поступать в систему и обрабатываться согласно настроенной в черновике конфигурации конвейера.

Отключение активной конфигурации доступно из любого режима работы с конвейером.

Чтобы отключить активную конфигурацию конвейера, нажмите на кнопку **Отключить конфигурацию** над диаграммой конфигурации и подтвердите отключение. Система отключит конвейер и отобразит соответствующее уведомление. События перестанут поступать в систему и нормализовываться.

Чтобы выйти из диаграммы конфигурации и вернуться на вкладку **Конвейеры** в карточке коллектора, нажмите на кнопку **Назад** в левой верхней части экрана.



Если диаграмма конфигурации была открыта через вкладку **Конвейеры** в карточке элемента экспертизы, то при нажатии на кнопку **Назад** осуществится переход обратно в эту вкладку.

Включить и выключить конвейер также можно в карточке конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку ⌄ в строке конвейера. Отобразится карточка конвейера.
5. Включите или выключите конвейер:
   - Для включения конвейера используйте кнопку **Установить конфигурацию** в нижней части карточки.
   - Для выключения конвейера используйте кнопку **Отключить конфигурацию** в нижней части карточки.

Изменение конвейера



Перед тем как редактировать конфигурацию конвейера, убедитесь, что его конфигурация не установлена в коллекторе.

Чтобы изменить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку действий ⋮ в строке конвейера и выберите опцию **Изменить**. Система отобразит окно редактирования конвейера.
5. Внесите изменения в требуемые поля.
6. Нажмите на кнопку **Сохранить**. Измененные данные конвейера будут сохранены.

Создание копии конвейера

Чтобы создать копию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.

2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку действий ⋮ в строке конвейера и выберите опцию **Создать копию**. Система отобразит окно создания копии конвейера.
5. Измените требуемые поля конвейера.
6. Нажмите на кнопку **Создать**. Копия конвейера появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Удаление конвейера



Удаление недоступно для конвейеров, используемых в других сущностях системы.

Чтобы удалить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку действий ⋮ в строке конвейера и выберите опцию **Удалить**. Система отобразит окно подтверждения действия.
5. Нажмите на кнопку **Удалить**. Система удалит выбранный конвейер и отобразит соответствующее уведомление. Удаленный конвейер исчезнет из списка конвейеров на вкладке **Конвейеры** в карточке коллектора.



Вместе с конвейером также удаляются и все добавленные в него элементы.

Импорт конвейера

Чтобы импортировать конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку **Импортировать** в нижней части карточки. Отобразится окно импорта конвейера.
5. Нажмите на кнопку **Выбрать файл** и выберите файл конвейера для импорта.
6. Нажмите на кнопку **Загрузить**. Загруженный конвейер отобразится в списке конвейеров на

вкладке **Конвейеры** в карточке коллектора.



Если в нормализаторе конвейера был включен аудит источников, то при импорте конвейера применяется политика аудита, установленная в системе по умолчанию.

Экспорт конвейера

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку действий ⋮ в строке конвейера и выберите опцию **Экспортировать**. Система отобразит окно подтверждения действия.
5. При необходимости включите опцию **Экспортировать без изменяемых параметров**.

   ▼ *Параметры конвейера, которые можно экспортировать с пустыми значениями*

   - `databaseDriver` — драйвер базы данных;
   - `credentials` — данные базовой аутентификации;
   - `token` — токен для аутентификации;
   - `storageId` — ID конечной точки хранилища событий;
   - `eventStorageId` — ID сервиса оповещений;
   - `eventBusId` — ID шины;
   - `monitoringPolicyId` — ID политики аудита.

6. Нажмите на кнопку **Экспортировать**. Данные конвейера будут экспортированы в файл формата JSON.

Работа с точками входа

Данный раздел описывает процесс работы с точками входа на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О точках входа

Точка входа — это элемент конвейера, который отвечает за доставку в систему исходных событий. Выбор типа точки входа зависит от формата данных, получаемых из источника события. На диаграмме конфигурации конвейера точка входа имеет один выход и не имеет входов.

В системе доступны следующие типы точек входа:

- AMQP
- Audit
- Database

- Demo Logs
- R-Vision Endpoint
- FTP
- Global Bus
- HTTP Client
- HTTP Server
- Kafka
- Logstash
- MongoDB
- NATS JetStream
- Netflow
- SFTP
- SMB
- Socket
- Syslog
- Vector
- WMI



Некоторые точки входа, такие как Syslog, позволяют получать события только по портам из определенного диапазона, например, 30000—32767. Однако, если нет возможности передавать события из внешней системы на порты из этого диапазона, вы можете настроить проксирование трафика через Ingress-контроллер. В таком случае информация о событиях будет перенаправляться из порта, на который она поступает, на порт, который будет использоваться в конвейере обработки событий.

Чтобы настроить проксирование событий, обратитесь к разделу Настройка получения событий по произвольному порту.

Доступные операции над точкой входа:

- Добавление точки входа
- Изменение точки входа
- Удаление точки входа
- Настройка точки входа

Добавление точки входа

Чтобы добавить точку входа на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации

конвейера выбран вариант **Черновик**.

3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
4. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
5. Введите название точки входа.
6. Выберите тип точки входа и задайте ее параметры. Набор параметров точки входа зависит от выбранного типа.



Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

7. Если требуется отслеживать состояние источника данных, выберите политику аудита источников. В противном случае установите переключатель **Аудит** в неактивное положение.

   По умолчанию выбрана системная **Политика аудита источников по точке входа**.



Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.

8. Если заданные настройки точки входа необходимо сохранить в качестве шаблона для дальнейшего использования, установите флажок **Сохранить как шаблон**.
9. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Изменение точки входа

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление точки входа

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Настройка точки входа

Набор параметров точки входа отличается в зависимости от ее типа. В системе доступны следующие типы точек входа:

- AMQP
- Audit
- Database
- Demo Logs
- R-Vision Endpoint
- FTP

- Global Bus
- HTTP Client
- HTTP Server
- Kafka
- Logstash
- MongoDB
- NATS JetStream
- Netflow
- SFTP
- SMB
- Socket
- Syslog
- Vector
- WMI

AMQP

При создании точки входа типа AMQP требуется указать значения следующих параметров:

1. **Адрес подключения**: выбор секрета типа **Строка подключения**, хранящего URI-адрес сервера AMQP в формате: `amqp://<user>:<password>@<host>:<port>/<vhost>?timeout=<seconds>`.

   Например: `amqp://user:password@127.0.0.1:5672/%2f?timeout=10`.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

2. **Идентификатор потребителя** (опционально): идентификатор группы потребителей, который будет использоваться для получения событий.

   Например: `event-consumer-group`.

3. **Ключ exchange** (опционально): ввод ключа exchange.
4. **Ключ смещения AMQP** (опционально): ввод ключа смещения AMQP.
5. **Название очереди** (опционально): ввод названия очереди.
6. **Ключ маршрутизации AMQP** (опционально): ввод ключа маршрутизации AMQP.
7. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Audit

Точка входа Audit собирает события сервиса аудита и доставляет их на конвейер для дальнейшей обработки и анализа.

Сервис аудита предназначен для сбора, анализа и хранения данных о действиях пользователей и событиях в системе. Сервис аудита позволяет фиксировать такие действия, как входы в систему, создание, удаление и изменение ресурсов системы и конфигурации ее отдельных компонентов.

С помощью точки входа Audit можно отслеживать действия пользователей в системе для предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности. Например, вы можете настроить конфигурацию конвейера на выполнение следующих операций:

- Создание оповещений о критических событиях
- Отслеживание действий суперадминистраторов
- Отслеживание действий над объектами

При создании точки входа типа Audit требуется указать значения следующих параметров:

1. **Уровни угрозы**: выбор уровня угрозы для фильтрации потока входящих событий аудита. Доступные варианты:
   - **Отладочный** — уровень угрозы 1 (`DEBUG`);
   - **Низкий** — уровень угрозы 2 (`LOW`);
   - **Средний** — уровень угрозы 3 (`MEDIUM`);
   - **Высокий** — уровень угрозы 4 (`HIGH)`;
   - **Критический** — уровень угрозы 5 (`CRITICAL`);
   - **Фатальный** — уровень угрозы 6 (`FATAL`);
   - **Аварийный** — уровень угрозы 7 (`ACCIDENT`).
2. **Максимум сообщений в обработке**: максимальное количество сообщений NATS, отправленных на точку входа, но еще не подтвержденных системой. Должно быть целым числом без знака, в диапазоне от `0` до `100000`. По умолчанию установлено ограничение в 1000 сообщений.

Database



Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. При размещении точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

При создании точки входа типа Database требуется указать значения следующих параметров:

1. **Драйвер базы данных**: выбор драйвера БД. По умолчанию доступны предустановленные драйвера следующих баз данных:

   - ClickHouse;
   - PostgreSQL;
   - MS SQL;
   - MySQL;
   - Oracle DB.

   Чтобы просмотреть выбранный драйвер на странице раздела Драйверы БД, нажмите на кнопку **Открыть**.

2. **Адрес подключения**: выбор секрета типа **Строка подключения**, хранящего URI-адрес для подключения к базе данных. Формат URI должен соответствовать требованиям выбранной базы данных:

   - **PostgreSQL**: `jdbc:postgresql://[host][:<port>]/[database][?user=<user>&password=<password>&<parameters>]`

     Здесь:

     - `host` — адрес подключения к БД в формате `<хост>[:<порт>]`. Например: `localhost:5433`.
     - `database` — имя БД.
     - `user` — учетные данные пользователя БД.
     - `password` — пароль пользователя БД.
     - `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `&`.

       Учетные данные, адрес подключения и имя БД можно задавать в виде параметров.

     Полный список параметров приведен в документации PostgreSQL.

     Примеры строки подключения:

     - `jdbc:postgresql://localhost/testdb?user=user&password=secret&ssl=true`
     - `jdbc:postgresql://localhost:5432/testdb?user=user&password=password`

   - **ClickHouse**: `jdbc:clickhouse://[user@][host][/database][?parameters]`

     Здесь:

     - `user` — учетные данные пользователя БД в формате `<имя_пользователя>[:<пароль>]`. Например: `user:password`.
     - `host` — адрес подключения к БД в формате `<хост>[:<порт>]`. Например: `localhost:5433`.
     - `database` — имя БД.

- `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `&`.

  Учетные данные, адрес подключения и имя БД можно задавать в виде параметров.

  Параметры ClickHouse идентичны параметрам PostgreSQL. Полный список параметров приведен в документации PostgreSQL.

  При работе с БД ClickHouse в качестве точки входа следует соблюдать следующие рекомендации:

  - Не следует обращаться к базе данных `system`, поскольку она недоступна для чтения.
  - Рекомендуется использовать сжатие передаваемых данных алгоритмом `gzip`. Для этого используйте параметр `compress_algorithm=gzip` в строке подключения.

  Примеры строки подключения:

- `jdbc:clickhouse://user:password@localhost/database&compress_algorithm=gzip`
- `jdbc:clickhouse://localhost/database?user=user&password=password&compress_algorithm=gzip`

◦ **MS SQL**: `jdbc:sqlserver://[host][;parameters]`

Здесь:

- `host` — адрес подключения к БД в формате `<хост>[\<имя_экземпляра>][:<порт>]`. Например: `localhost\instance1:1433`.
- `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `;`.

  Учетные данные пользователя и имя БД можно задавать в виде параметров.

  Полный список параметров приведен в документации драйвера JDBC для SQL Server.

Примеры строки подключения:

- `jdbc:sqlserver://localhost;user=user;password=password`
- `jdbc:sqlserver://localhost:1433;database=database`

◦ **MySQL**: `jdbc:mysql://[user@][host][/database][?parameters]`

Здесь:

- `user` — учетные данные пользователя БД в формате `<имя_пользователя>[:<пароль>]`. Например: `user:password`.
- `host` — адрес подключения к БД в формате `<хост>[:<порт>]`. Например: `localhost:3306`.

- `database` — имя БД.
- `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `&`.



Учетные данные, адрес подключения и имя БД можно задавать в виде параметров.

Полный список параметров приведен в документации MySQL.

Примеры строки подключения:

- `jdbc:mysql://user:password@localhost:3306/database`
- `jdbc:mysql://localhost:3306/database?user=user&password=password`

◦ **Oracle** **DB**: `jdbc:oracle:thin:@[[protocol:]//]host[/service][:server_mode][/instance][?parameters]`

Здесь:

- `protocol` — протокол передачи данных. Например: `tcp`.
- `host` — адрес подключения к БД в формате `<хост>:[<порт>]`. Например: `localhost:1521`.
- `service` — имя сервиса для подключения к экземпляру БД.
- `server_mode` — режим работы сервера БД. Доступные значения:
  - `dedicated` — выделенный сервер;
  - `shared` — общий сервер;
  - `pooled` — сервер, добавленный в пул.
- `instance` — имя экземпляра БД.
- `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `&`.

Полный список параметров приведен в документации Oracle.

Примеры строки подключения:

- `jdbc:oracle:thin:@tcp//localhost:1521/service`
- `jdbc:oracle:thin:@tcp//localhost:1521/service/instance1`

Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

3. **SQL-запрос**: ввод SQL-запроса для работы с базой данных.

Пример SQL-запроса

```
SELECT id, raw, collectorId FROM `default`.EventStorage_1072 limit 10;
```



При написании SQL-запросов, которые объединяют несколько таблиц, избегайте

одинаковых имен столбцов, чтобы предотвратить ошибки. Для этого:

- Вместо `SELECT *` перечисляйте конкретные столбцы.
- Используйте псевдонимы для столбцов с одинаковыми именами.

Например, вместо запроса вида

```
SELECT
    *
FROM
    Database1.TableStart AS tableStart
JOIN Database1.TableStop AS tableStop
ON
    tableStart.id = tableStop.id
WHERE
    tableStart.id > ?;
```

используйте запрос

```
SELECT
    tableStart.id AS tableStartId,
    /* псевдоним для столбца id в первой таблице */
    tableStart.AdminId,
    tableStart.StartTime,
    tableStop.id,
    tableStop.EndTime,
    tableStop.IsSuccessful
    /* перечислены конкретные столбцы */
FROM
    Database1.TableStart AS tableStart
JOIN Database1.TableStop AS tableStop
ON
    tableStart.id = tableStop.id
WHERE
    tableStart.id > ?;
```

4. **Поле идентификатора**: добавление полей-идентификаторов для определения строки начала чтения при первом подключении к БД. Список полей представлен в виде таблицы со столбцом для указания ключа поля идентификатора и столбцом для задания его стартового значения.



Система хранит значение последней обработанной записи и при следующем выполнении запроса модифицирует условие выборки, инициируя чтение данных не с указанной строки в **Поле идентификатора**, а с записи, следующей непосредственно за последней обработанной.



Необходимо гарантировать, что значения всех полей, определенных в качестве идентификаторов в строке строго согласуются между собой, отражая

уникальные данные одной записи в базе данных.

5. **Интервал запроса, секунд** (опционально): интервал между запросами в секундах.



Интервал между инициациями SQL-запросов не охватывает время исполнения запросов, что может фактически продлить интервал между запусками.

6. **Максимальное количество строк** (опционально): максимальное количество строк для чтения в ходе выполнения SQL-запроса.

Пример настройки точки входа Database

Представим, что в подключаемой базе данных имеется таблица событий `events`, из которой необходимо извлечь данные по столбцам `id`, `timestamp` и `data`.

После определения типа базы данных и конфигурации строки подключения, следующий шаг заключается в настройке идентификационных полей для отслеживания и выборки данных. В качестве таких полей можно использовать `id` и `timestamp`. Добавим их ключи в левый столбец таблицы **Поле идентификатора**.



Необходимо гарантировать, что значения всех полей, определенных в качестве идентификаторов в строке — таких как `id`, `timestamp` и любые другие участвующие поля — строго согласуются между собой, отражая уникальные данные одной записи в базе данных. Эти поля в совокупности формируют составной ключ, обеспечивая точную идентификацию записи в БД.

Предположим, что первые 50 записей не должны попасть в выборку. Учитывая, что идентификаторы (`id`) в базе данных присваиваются последовательно, установим для поля `id` в **Поле идентификатора** значение "51". Таким образом, извлечение данных начнется непосредственно с записи, имеющей `id=51`.

Если предположить, что запись с `id=51` соответствует событию, произошедшему в "2024-01-01 00:00:00", это значение также указывается в качестве критерия выборки в **Поле идентификатора** для `timestamp`. Использование `id` в сочетании с `timestamp` в качестве начальных условий выборки формирует составной ключ, обеспечивающий уникальность каждой записи в базе данных.

Определение и использование составных ключей, основанных на нескольких идентификаторах, таких как `id` и `timestamp`, повышает точность выборки данных, минимизируя возможность дублирования или пропуска записей при изменении условий выборки или в случае неоднозначности идентификации записей только по одному критерию.

В целях снижения нагрузки на коллектор можно ограничить считывание данных из БД порциями по 100 записей, используя в SQL-запросе оператор `LIMIT`. Например:

```
SELECT id, timestamp, data FROM events WHERE id > ? ORDER BY id ASC LIMIT 100;
```

Здесь `?` — это параметр, который позволяет динамически подставлять значение идентификатора (`id`) последней обработанной записи в базе данных. Система хранит `id` последней обработанной записи и при последующем выполнении запроса адаптирует условие выборки, запуская процесс чтения

данных не с фиксированной строки (в данном случае, не с 51), а с записи, следующей непосредственно за последней обработанной.

Предположим, новые данные поступают в рассматриваемую базу данных с периодичностью в 60 минут. Для оптимизации процесса извлечения данных рекомендуется настроить периодичность выполнения SQL-запросов, соответствующую этому интервалу. Оптимальным решением будет установка интервала в 3600 секунд (один час), что позволит системе регулярно обновлять данные без излишних запросов.



Указанный интервал между инициациями SQL-запросов не охватывает время исполнения запросов, что может фактически продлить интервал между запусками.

Таким образом, вышеприведенные настройки позволят считать из таблицы `events` в подключенной БД все события начиная с `id=51` и `timestamp=`2024-01-01 00:00:00`` порциями по 100 записей. Порции записей будут направляться в точку входа раз в 3600 секунд.

Demo Logs

Точка входа Demo Logs предназначена для генерации случайных событий в целях демонстрации или тестирования. Формат, в котором будут генерироваться события, определяется пользователем.

При создании точки входа типа Demo Logs требуется указать значения следующих параметров:

1. **Количество событий** (опционально): количество событий для вывода. По умолчанию точка входа выводит логи непрерывно. Формат: целое число больше 0.
2. **Формат генерируемых событий**: выбор формата, в котором будут генерироваться события. Варианты:
   - **apache_common** — общий журнал Apache;
   - **apache_error** — журнал ошибок Apache;
   - **bsd_syslog** — формат BSD syslog (RFC 3164);
   - **json** — логи HTTP-сервера в формате JSON;
   - **shuffle** — выбор событий из заданного списка;
   - **syslog** — формат syslog (RFC 5424).
3. **Интервал в секундах** (опционально): время задержки между пакетами событий вывода. Формат ввода: число с точностью до трех знаков после запятой, начиная с 0. По умолчанию события отправляются каждую секунду.
4. **Список событий**: отображается, если выбран формат shuffle.

   Чтобы добавить событие, нажмите на кнопку **Добавить**. Отобразится строка добавления события.

   Чтобы удалить событие, нажмите на кнопку в строке события.

   Примеры событий

   ```
   {"dvendor":"Microsoft","dhost":"host1","duser":"user1","sourceIp":"192.0.2.0"}
   ```

```
{"dvendor":"Linux","dhost":"host1","duser":"user1","sourceIp":"198.51.100.0"}
```

Экранировать символы в списке событий не требуется.

При выборе формата shuffle события передаются в виде строки. Чтобы обрабатывать эти события, преобразуйте их в нужный формат, например, с помощью функции `.raw = parse_json!(.raw)`. Чтобы привести метку времени к ожидаемому типу и формату данных (`2024-07-15T09:05:25.153815269Z`), используйте функцию parse_timestamp:

```
.timestamp = parse_timestamp!(.timestamp, format: "%FT%T%.9fZ")
```

R-Vision Endpoint

Точка входа типа R-Vision Endpoint предназначена для получения событий с агентов, установленных на конечных устройствах.

При создании точки входа типа R-Vision Endpoint требуется указать значения следующих параметров:

1. **Домен**: домен NATS JetStream для подключения к шлюзу. Домен задается в конфигурации NATS-сервера в формате `gw-<gateway_id>`, где `<gateway_id>` — идентификатор шлюза. Подключение выполняется к NATS-серверу, заданному по умолчанию: `nats://nats-space:4222`.

   Один шлюз может быть связан только с одной точкой входа R-Vision Endpoint. Таким образом, указанный домен NATS JetStream должен быть уникален и не может использоваться в других точках входа, в том числе в точках входа типа NATS JetStream.

2. **Максимум сообщений в обработке**: максимальное количество сообщений NATS, отправленных на точку входа, но еще не подтвержденных системой. Должно быть целым числом без знака, в диапазоне от `0` до `100000`. По умолчанию установлено ограничение в 1000 сообщений.

FTP

Точка входа типа FTP позволяет собирать данные текстовых файлов с удаленных ресурсов с помощью протокола FTP.

Чтение данных происходит построчно. Для корректного чтения файла его последняя строка должна быть пустой, то есть последним байтом должен быть символ `\n`. В противном случае последняя строка данных в файле будет проигнорирована.

FTP (File Transfer Protocol) — протокол, позволяющий передавать файлы по сети без шифрования. FTP применяется для передачи **неконфиденциальных данных**, например, для загрузки общедоступных файлов с веб-сайта.

При создании точки входа типа FTP требуется указать значения следующих параметров:

1. **Адрес подключения**: полный адрес для подключения к серверу в виде URI, например, `127.0.0.1:9898/logs` или `somehost:9898/logs`.

   По умолчанию используется порт 21.

2. **Тип аутентификации** (опционально): выбор секрета типа **Учетные данные**, хранящего имя пользователя и пароль для авторизации.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

3. **Маска файла**: путь или маска для поиска текстовых файлов на сервере, например, `/var/log/**/*.log`.
4. **Интервал запроса, секунд** (опционально): интервал между запросами в секундах. Интервал включает в себя время выполнения запроса данных. Интервал по умолчанию: 15 секунд.

   Интервал запроса рекомендуется устанавливать выше значения таймаута запроса.

5. **Таймаут запроса, секунд** (опционально): максимальное время выполнения запроса в секундах. Если запрос будет выполняться дольше указанного времени, его выполнение будет прекращено. Таймаут по умолчанию: 5 секунд.
6. **Многострочная обработка** (опционально): объединение нескольких строк лога в одно событие. При включении многострочной обработки необходимо:

   a. Указать шаблон начала события в виде регулярного выражения. Данный шаблон определяет начало нового события, к которому будут добавляться агрегированные строки.

   Примеры регулярных выражений для начала события приведены в разделе Примеры настройки многострочной обработки.

   b. Выбрать способ агрегации строк лога. Доступные варианты:

   - **По шаблону + 1** — к событию добавляются последовательные строки, удовлетворяющие шаблону, а также первая строка, для которой шаблон не сработал.

     Агрегация **По шаблону + 1** полезна, когда запись лога состоит из нескольких строк, которые заканчиваются маркером продолжения, например, обратной косой чертой.

     Данный способ выбран по умолчанию.

   - **По шаблону** — к событию добавляются последовательные строки, удовлетворяющие шаблону.

     Агрегация **По шаблону** полезна, когда строки-продолжения обладают общим признаком, например, ведущими отступами в виде пробелов или табуляции.

   - **Не по шаблону** — к событию добавляются последовательные строки, не удовлетворяющие шаблону.

     Агрегация **Не по шаблону** полезна, когда начало записи лога определяется маркером или

признаком, например, типом сообщения. В таких случаях строки без этого признака относятся к предыдущей записи.

- **Не по шаблону + 1** — к событию добавляются последовательные строки, не удовлетворяющие шаблону, а также первая строка, для которой шаблон сработал.

  Агрегация **Не по шаблону + 1** полезна, когда сообщение заканчивается маркером (например, точкой с запятой), после которого начинается новое сообщение.



Примеры логов для каждого способа агрегации приведены в разделе Примеры настройки многострочной обработки.

c. Указать шаблон условия агрегации в виде регулярного выражения.



Шаблон условия может не совпадать с шаблоном начала события.



Примеры регулярных выражений для условия агрегации приведены в разделе Примеры настройки многострочной обработки.

d. Ввести таймаут агрегации в секундах, по истечении которого событие отправляется, даже если агрегация не завершена. Таймаут по умолчанию: 1 секунда.

7. **Кодировка** (опционально): кодировка данных, полученных от источника. Позволяет задать кодировку, если требуется специальная обработка текстовых данных.

   Доступные кодировки приведены в разделе Кодировки в точках входа.

8. **Цифровой отпечаток**: количество строк, которые будут считываться с начала файла для создания его уникального идентификатора.

   Если файл содержит меньше строк, чем указано в поле, он будет пропущен.

9. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Global Bus

Точка входа типа Global Bus предназначена для получения событий из глобальной шины.



Данная точка входа необходима для работы с коллекторами вне центрального пространства, но также может использоваться и с коллекторами в центральном пространстве.

> Необходимость использования глобальной шины для отправки и получения событий из вынесенных коллекторов обусловлена тем, что запись событий в хранилище доступна только на конвейере коллектора, находящегося в центральном пространстве.
>
> Пример обработки событий, собранных в вынесенном коллекторе, рассмотрен в разделе Сбор событий из вынесенного коллектора.

При создании точки входа типа Global Bus требуется указать значения следующих параметров:

1. **Коллектор**: выбор коллектора, из которого необходимо получать события.
2. **Максимум сообщений в обработке**: максимальное количество сообщений NATS, отправленных на точку входа, но еще не подтвержденных системой. Должно быть целым числом без знака, в диапазоне от `0` до `100000`. По умолчанию установлено ограничение в 1000 сообщений.

HTTP Client

Точка входа HTTP Client выполняет роль клиента во взаимодействии типа "клиент — сервер". Она позволяет подключаться к внешнему серверу по HTTP-протоколу для активного сбора данных с этого сервера. Сбор данных происходит за счет регулярного выполнения HTTP-запросов, посылаемых от точки входа (клиента) к серверу.

При создании точки входа типа HTTP Client требуется указать значения следующих параметров:

1. **Адрес подключения**: полный адрес для подключения к HTTP-серверу в виде URI, например, `http://127.0.0.1:9898/logs` или `http://somehost:9898/logs`.
2. **Тип аутентификации** (опционально): выбор типа аутентификации на HTTP-сервере:
   - **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.
   - **Токен**: выберите секрет типа **Токен**, хранящий токен для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.
3. **HTTP-заголовки** (опционально): заголовки HTTP-запроса. Заголовки позволяют уточнить HTTP-запрос и определяют, какие данные требуется передавать в запросе.

   ▼ *Об HTTP-заголовках*

   HTTP-заголовок — это строка в HTTP-запросе, состоящая из пары "имя: значение". Он используется для уточнения запроса или описания содержимого сообщения. Формат заголовка определяется стандартом RFC 822.

   HTTP-заголовки делятся на следующие категории:

   - Основные заголовки (General Headers) — применяются во всех сообщениях клиента и сервера.

- Заголовки запроса (Request Headers) — используются только в запросах клиента.
- Заголовки ответа (Response Headers) — используются только в ответах от сервера.
- Заголовки сущности (Entity Headers) — применяются к содержимому сообщения.

Полный список HTTP-заголовков можно найти в MDN Web Docs.



Пример HTTP-заголовка

Для считывания данных с сервера порциями по 512 байт, в HTTP-запросе необходимо указать заголовок `Content-Length`. Значение этого заголовка должно соответствовать максимальному размеру поступающих данных в байтах, в данном случае это будет `512`.

Чтобы задать заголовок:

a. Нажмите на кнопку **Добавить**. Отобразится строка добавления заголовка.

b. Введите заголовок и его значение. Для одного заголовка можно задать несколько значений, разделенных запятой.



Точка входа HTTP Client поддерживает пагинацию — постраничное получение данных, когда данные поступают в систему последовательными порциями. Чтобы запрашивать данные с пагинацией, в качестве указателя (курсора) на следующую порцию данных следует использовать поле идентификатора.

Укажите ключ поля идентификатора в двойных фигурных скобках как значение заголовка, например:

- Поле идентификатора: `OffsetAmount`.
- Заголовок: `Offset`.
- Значение заголовка: `{{OffsetAmount}}`.

Чтобы удалить заголовок, нажмите на кнопку в строке нужного заголовка.

4. **Параметры запроса** (опционально): дополнительные параметры, применяемые к строке HTTP-запроса. Параметры представляют собой пары "ключ=значение" и позволяют конкретизировать ресурс, получаемый из HTTP-запроса.



Пример HTTP-параметра

Допустим, HTTP-ресурс содержит множество событий, из которых необходимо выбрать только события с уровнем угрозы **Критический**. Для указания критического уровня угрозы используется пара `severity=critical`. Таким образом, чтобы получить нужные данные, в HTTP-запросе необходимо передать параметр `severity` со значением `critical`.

Чтобы задать параметр:

a. Нажмите на кнопку **Добавить**. Отобразится строка добавления параметра.

b. Введите параметр и его значение. Для одного параметра можно задать несколько значений, разделенных запятой.



Точка входа HTTP Client поддерживает пагинацию — постраничное получение данных, когда данные поступают в систему последовательными порциями. Чтобы запрашивать данные с пагинацией, в качестве указателя (курсора) на следующую порцию данных следует использовать поле идентификатора.

Укажите ключ поля идентификатора в двойных фигурных скобках как значение параметра, например:

- Поле идентификатора: `OffsetAmount`.
- Параметр: `Offset`.
- Значение параметра: `{{OffsetAmount}}`.

Чтобы удалить параметр, нажмите на кнопку  в строке нужного параметра.

5. **Метод запроса** (опционально): выбор метода HTTP-запроса:
   - **DELETE** — удаляет данные с сервера.
   - **GET** — запрашивает данные с сервера. Метод предназначен только для получения данных без дополнительных взаимодействий с сервером.
   - **HEAD** — запрашивает данные с сервера без тела HTTP-ответа.
   - **PATCH** — используется для частичного изменения ресурса.
   - **POST** — отправляет данные на сервер. Метод используется в тех случаях, когда для получения данных требуется дополнительное взаимодействие с сервером (например, передача данных для авторизации на сервере).
   - **PUT** — заменяет данные на сервере данными из HTTP-запроса.
6. **Тело запроса** (опционально): тело отправляемого запроса в формате JSON.



Настройка тела запроса доступна, если выбран метод запроса **PATCH**, **POST** или **PUT**.

Точка входа HTTP Client поддерживает пагинацию — постраничное получение данных, когда данные поступают в систему последовательными порциями. Чтобы запрашивать данные с пагинацией, в качестве указателя (курсора) на следующую порцию данных следует использовать поле идентификатора.

Укажите ключ поля идентификатора в двойных фигурных скобках как значение параметра тела запроса, например:

- Поле идентификатора: `OffsetAmount`.
- Параметр: `Offset`.
- Значение параметра в теле запроса: `"Offset": {{OffsetAmount}}`.

7. **Поле идентификатора** (опционально): список полей-идентификаторов для определения строки

начала чтения при запросе к внешнему серверу. Список представлен в виде таблицы со следующими столбцами:

- **Ключ** — ключ поля идентификатора.
- **Значение по умолчанию** — значение идентификатора при первом запросе к внешнему серверу. Допустимые значения — число или строка длиной до 255 символов.
- **Шаблон** — путь к параметру ответа на запрос, содержащему значение идентификатора. С этого значения будет продолжено чтение при следующем запросе. Пример: `response.body.pageInfo.offset`.



Поддерживаются все конструкции синтаксиса языка шаблонов Handlebars.

8. **Интервал запроса, секунд** (опционально): интервал между HTTP-запросами в секундах. Интервал включает в себя время выполнения HTTP-запроса.



Интервал запроса рекомендуется устанавливать выше значения таймаута запроса.

9. **Таймаут запроса, секунд** (опционально): максимальное время выполнения HTTP-запроса в секундах. Если запрос будет выполняться дольше указанного времени, его выполнение будет прекращено.
10. **Адрес HTTP-прокси** (опционально): полный адрес для подключения к HTTP-прокси в виде URI. Применяется, если HTTP-сервер использует прокси.
11. **Хосты без прокси** (опционально): список хостов, для которых не используется прокси-сервер.

    Чтобы добавить хост без прокси:

    1. Нажмите на кнопку **Добавить**. Отобразится строка добавления хоста.
    2. Введите название хоста.



Для названий хостов допускается использовать:

- доменные имена — например, `example.com`;
- шаблоны доменов — например, `.example.com`;
- IP-адреса — например, `127.0.0.1`;
- CIDR-блоки — например, `192.0.2.1/16`.

    Чтобы удалить хост без прокси, нажмите на кнопку  в его строке.

12. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

    a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

    b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Пример настройки точки входа HTTP Client

Пусть требуется получать события с внешнего сервера. Для этого можно настроить точку входа следующим образом:

1. Введите адрес подключения и выберите тип аутентификации.
2. Настройте заголовки отправляемых HTTP-запросов. Например, для снижения нагрузки на сервер можно ограничить считывание данных порциями по 512 байт. Для этого в поле **HTTP-заголовки** добавьте заголовок `Content-Length` и установите для него значение 512.
3. Настройте параметры отправляемых HTTP-запросов. Предположим, что на сервере хранятся события с различным уровнем угрозы, которые определяются параметром `severity`. Чтобы получать только события с уровнем угрозы **Критический** (`critical`), в поле **Параметры запроса** добавьте параметр `severity` с соответствующим значением.
4. Выберите метод запроса. Так как для доступа к серверу требуется передать данные для авторизации, для повышения безопасности их передачи следует использовать метод POST. В отличие от метода GET, POST кодирует данные и передает их в теле HTTP-запроса, обеспечивая их конфиденциальность. Выберите метод POST в поле **Метод запроса**.
5. Настройте интервал между отправкой запросов. Предположим, что новые данные поступают на сервер с периодичностью 60 минут. Рекомендуется настроить выполнение HTTP-запросов с таким же интервалом. Оптимальным будет интервал в 3600 секунд (один час), что позволит системе регулярно обновлять данные без излишних запросов. Введите значение 3600 в поле **Интервал запроса, секунд**.

   Указанный интервал между HTTP-запросами включает время их выполнения.

6. Поскольку время выполнения HTTP-запросов может превысить указанный интервал, необходимо задать таймаут, ограничивающий максимальное время выполнения запросов. Оптимальным решением будет установка таймаута в 3000 секунд (50 минут), что гарантирует завершение текущего запроса до инициации следующего. Введите значение 3000 в поле **Таймаут запроса, секунд**.

Таким образом, вышеприведенные настройки позволят считывать с сервера события с критическим уровнем угрозы порциями по 512 байт. Порции записей будут направляться в точку входа каждые 3600 секунд.

Пример настройки точки входа HTTP Client с использованием полей идентификаторов

Пусть требуется получать события с внешнего сервера, однако, в отличие от предыдущего примера, требуется получать события постранично (использовать пагинацию) и предотвращать дублирование получаемых событий. Для этого можно использовать поле идентификатора события, с которого следует начать чтение в следующем запросе. Алгоритм настройки точки входа в этом случая будет следующим:

1. Введите адрес подключения и выберите тип аутентификации.
2. Настройте идентификатор события, с которого следует начать чтение:
   a. Добавьте поле в **Поле идентификатора**.
   b. В поле **Ключ** укажите имя идентификатора: например, `OffsetEventID`.
   c. В поле **Значение по умолчанию** введите идентификатор события во внешней системе, с которого начнется чтение событий при первом запросе.
   d. В поле **Шаблон** введите путь к параметру ответа на запрос, содержащему значение идентификатора. С этого значения будет продолжено чтение при следующем запросе. Например, `response.body.pageInfo.offset`.
3. Настройте параметры отправляемых HTTP-запросов:
   a. Чтобы получать только события с уровнем угрозы **Критический** (`critical`), в поле **Параметры запроса** добавьте параметр `severity` с соответствующим значением.
   b. Чтобы ограничить количество событий, получаемых в ответе на один запрос, в поле **Параметры запроса** добавьте параметр `limit` с соответствующим значением: например, 100.
   c. Чтобы использовать идентификатор события, с которого будет начато чтение при следующем запросе, в поле **Параметры запроса** добавьте параметр `offset` со значением `{{OffsetEventID}}`.
4. В качестве метода запроса выберите POST.
5. Настройте интервал и таймаут аналогично предыдущему примеру:
   - **Интервал запроса, секунд**: 3600.
   - **Таймаут запроса, секунд**: 3000.

Таким образом, вышеприведенные настройки позволят считывать с сервера события с критическим уровнем угрозы каждые 3600 секунд. Каждый запрос будет возвращать до 100 событий без повторения идентификаторов.

HTTP Server

Точка входа HTTP Server выполняет роль сервера во взаимодействии типа "клиент — сервер". Она позволяет внешним источникам (клиентам) подключаться к ней для пассивного сбора данных с помощью HTTP-запросов. Обработка HTTP-запросов от клиентов приводит к генерации событий, которые поступают в точку входа (на сервер).

При создании точки входа типа HTTP Server требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

   Если необходимо подключиться к узлу кластера Kubernetes по порту, отличному от порта точки входа, установите переключатель **Отдельный порт для узла кластера** в активное положение. В появившемся поле укажите порт узла кластера. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

2. **Кодировка**: выбор формата кодировки событий. Доступные форматы:
   - **Text** — выбран по умолчанию;

- **JSON**;
- **Newline-delimited JSON**;
- **Binary**.

3. **Путь** (опционально): URL-путь, используемый клиентом для записи событий при использовании метода POST. Например: `/logs` или `/logs/application`.

   Путь будет добавлен в метаданные поступающих событий.

   По умолчанию точка входа обрабатывает HTTP-запросы, поступающие по любому URL-пути, соответствующему заданному шаблону. Например, при настройке пути в виде `/logs`, система также обработает запросы, поступающие по вложенным путям, таким как `/logs/application`. Чтобы разрешить получение HTTP-запросов только с указанного пути, установите флажок **Абсолютный путь**. В этом случае все события, поступающие на путь, отличный от заданного, будут отбрасываться.

4. **HTTP-заголовки** (опционально): заголовки поступающих HTTP-запросов. Заголовки позволяют уточнить, какие дополнительные данные требуется получить от клиентов. Указанные заголовки будут добавлены в метаданные поступающих событий. Допустимо использование подстановочного символа *, например, `X-*`.

   ▼ *Об HTTP-заголовках*

   HTTP-заголовок — это строка в HTTP-запросе, состоящая из пары "имя: значение". Он используется для уточнения запроса или описания содержимого сообщения. Формат заголовка определяется стандартом RFC 822.

   HTTP-заголовки делятся на следующие категории:

   - Основные заголовки (General Headers) — применяются во всех сообщениях клиента и сервера.
   - Заголовки запроса (Request Headers) — используются только в запросах клиента.
   - Заголовки ответа (Response Headers) — используются только в ответах от сервера.
   - Заголовки сущности (Entity Headers) — применяются к содержимому сообщения.

   Полный список HTTP-заголовков можно найти в MDN Web Docs.

   Пример

   Чтобы определить, от какого агента поступило событие, необходимо указать заголовок `User-Agent`.

   Чтобы задать заголовок, нажмите на кнопку **Добавить**. Отобразится строка добавления заголовка. Введите заголовок.

   Чтобы удалить заголовок, нажмите на кнопку  в строке нужного заголовка.

5. **Параметры HTTP-запроса** (опционально): дополнительные параметры, применяемые к строке поступающих HTTP-запросов. Как и HTTP-заголовки, параметры позволяют конкретизировать данные, получаемые от клиентов. Указанные параметры будут добавлены в метаданные поступающих событий. Допустимо использование подстановочного символа *, например, `X-*`.



Пример

Предположим, что от клиентов поступают события с различным уровнем угрозы. Чтобы определить уровень угрозы каждого события, необходимо указать соответствующий параметр, например, `severity`.

Чтобы задать параметр, нажмите на кнопку **Добавить**. Отобразится строка добавления параметра. Введите параметр.

Чтобы удалить параметр, нажмите на кнопку  в строке нужного параметра.

6. **Метод запроса** (опционально): выбор метода поступающих HTTP-запросов:
   - **DELETE** — удаляет данные с сервера.
   - **GET** — запрашивает данные с сервера с полным HTTP-ответом.
   - **HEAD** — запрашивает данные с сервера без тела HTTP-ответа.
   - **PATCH** — используется для частичного изменения ресурса.
   - **POST** — отправляет данные на сервер.
   - **PUT** — заменяет данные на сервере данными из HTTP-запроса.
7. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Пример настройки точки входа HTTP Server

Предположим, требуется получать события от внешних клиентов. После определения порта точки входа, кодировки событий и пути записи событий, следующим шагом будет настройка заголовков и параметров в поступающих HTTP-запросах.

Допустим, необходимо определить, от какого агента поступило каждое событие. Для этого добавим заголовок `User-Agent`. В метаданные событий будет добавлено одноименное поле с информацией об агенте-источнике.

Предположим, что от клиентов поступают события с различным уровнем угрозы, которые определяются параметром `severity`. Чтобы определить уровень угрозы каждого события, добавим этот параметр. В метаданные событий будет добавлено одноименное поле с информацией об уровне угрозы.

Так как в данном примере предполагается только получение событий, в качестве метода поступающих HTTP-запросов выберем GET.

Таким образом, эти настройки позволят получать от клиентов события с дополнительной информацией об агентах-источниках и уровнях угрозы.

Kafka

При создании точки входа типа Kafka требуется указать значения следующих параметров:

1. **Адреса подключения**: список адресов серверов в кластере Kafka для подключения. Формат: IP-адрес:порт, например, 10.14.22.123:9092, 10.14.23.332:9092.

   Чтобы добавить адрес, нажмите на кнопку **Добавить**. Отобразится строка добавления адреса.

   Чтобы удалить адрес, нажмите на кнопку  в строке адреса.

2. **Название группы потребителей**: название Consumer group, которое будет использоваться для получения событий из Kafka, например, event-consumer-group.
3. **Названия топиков**: названия топиков Kafka, из которых будут читаться события.

   Чтобы добавить топик, нажмите на кнопку **Добавить**. Отобразится строка добавления топика.

   Чтобы удалить топик, нажмите на кнопку  в строке с названием топика.

4. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Logstash

При создании точки входа типа Logstash требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

   Если необходимо подключиться к узлу кластера Kubernetes по порту, отличному от порта точки входа, установите переключатель **Отдельный порт для узла кластера** в активное положение. В появившемся поле укажите порт узла кластера. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

2. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

3. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

MongoDB

Точка входа типа MongoDB позволяет собирать данные из одноименной базы данных. Это может быть полезно при необходимости получить логи приложений, которые записываются в MongoDB, для их дальнейшей обработки в системе.

При создании точки входа типа MongoDB требуется указать значения следующих параметров:

1. **Адрес подключения**: секрет типа **Строка подключения**, хранящего URI-адрес БД MongoDB в формате `mongodb://`. Например: `mongodb://127.0.0.1:27017`.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

2. **База данных**: имя БД, из которой необходимо считывать документы. Формат ввода:
   - Регистр символов учитывается.
   - Имя не может содержать пробелы, а также следующие символы:

     ```
     /\."
     ```

3. **Коллекция**: имя коллекции MongoDB, из которой необходимо считывать документы. Формат ввода:
    - Указывается только имя коллекции, без имени БД.
    - Регистр символов учитывается.
    - Не допускаются имена системных коллекций вида "system.*".
4. **Query-запрос**: запрос к MongoDB в формате JSON.

    Пример запроса приведен в разделе Пример запроса к MongoDB.

5. **Поле идентификатора**: поле идентификатора, которое позволит уникально идентифицировать каждый документ внутри коллекции. При переводе переключателя в активное состояние необходимо задать следующие параметры:

    a. **Ключ**: имя поля документа MongoDB, которое будет использоваться в качестве поля идентификатора.

    b. **Тип данных**: тип данных поля идентификатора. Доступные типы данных:
        - **ObjectId**;
        - **Timestamp**;
        - **DateTime**;
        - **Int32**;
        - **Int64**;
        - **Double**;
        - **Decimal128**;
        - **String**.

    c. **Значение**: стартовое значение поля идентификатора, начиная с которого начнется чтение документов. Формат задания значений зависит от выбранного типа данных, но допустимо задавать значения не длиннее 255 символов.

    Чтобы использовать поле идентификатора в запросе, необходимо обратиться к нему через двойные фигурные скобки: `{{<field_key>}}`.

    Здесь:

    - `<field_key>` — ключ, указанный в поле **Ключ**.

    Пример запроса с полем идентификатора приведен в разделе Пример запроса к MongoDB с полем идентификатора.

6. **Интервал запроса**: интервал между запросами к MongoDB в секундах. Интервал **не включает** в себя время на выполнение запроса. Интервал по умолчанию: 10 секунд.
7. **Максимальное количество документов** (опционально): максимальное количество документов, которое может быть прочитано за один запрос к MongoDB. Допустимые значения: 0 — 1 000 000.

Пример запроса к MongoDB

Предположим, что в текущей коллекции, указанной в поле **Коллекция**, содержится информация о событиях в определенном приложении. Стоит задача получить список пользователей, действия которых генерируют эти события, и дополнить их данными из коллекции `users`, хранящей информацию о пользователях в системе.

Пусть в текущей коллекции есть поле `user_id`, которое хранит идентификатор пользователя, ответственного за событие, а в коллекции `users` идентификатор пользователя хранится в поле `_id`.

Тогда, чтобы получить список пользователей и их данных, можно использовать следующий запрос:

```
[
  {
    "$group": {
      "_id": "$user_id"
    }
  },
  {
    "$lookup": {
      "from": "users",
      "localField": "_id",
      "foreignField": "_id",
      "as": "user_info"
    }
  },
  {
    "$unwind": "$user_info"
  }
]
```

В представленном запросе:

1. Оператор `$group` группирует все события из текущей коллекции по полю `user_id`. Это позволяет избавиться от дублирующихся записей. Например, если пользователь совершил 1000 действий, в дальнейшую обработку попадет только одна запись с его идентификатором.

   После группировки значение поля `_id` меняется на значение поля `user_id`.

2. Оператор `$lookup` объединяет результаты группировки и коллекцию `users` по следующему алгоритму:

   a. Находятся совпадения поля `_id` из результата группировки с полем `_id` из коллекции `users`.

   b. Для каждого совпадения в поле `user_info` записываются соответствующие данные из коллекции `users`.

3. Оператор `$unwind` создает отдельные документы для каждого пользователя из поля `user_info`.

Таким образом, запрос позволяет получить данные по всем пользователям, идентификатор которых упоминается в текущей коллекции.

Пример запроса к MongoDB с полем идентификатора

Предположим, что в текущей коллекции, указанной в поле **Коллекция**, содержатся основные логи приложений, а дополнительные подробности, например, метаданные или текст ошибки, вынесены в отдельную коллекцию `logDetails`. Требуется порционно получать подробную информацию по последним 10 000 событий.

Пусть в текущей коллекции поле `detail_id` хранит ссылку на подробные сведения, а в коллекции `logDetails` поле `_id` является идентификатором этих сведений.

Тогда, чтобы получить интересующие нас данные, можно использовать следующий запрос:

```
[
  {
    "$match": {
      "_id": {
        "$lt": {{batch}}
      }
    }
  },
  {
    "$sort": {
      "_id": -1
    }
  },
  {
    "$limit": 10000
  },
  {
    "$lookup": {
      "from": "logDetails",
      "localField": "detail_id",
      "foreignField": "_id",
      "as": "details"
    }
  },
  {
    "$unwind": "$details"
  }
]
```

В представленном запросе:

1. Оператор `$match` отбирает документы, у которых значение поля `_id` меньше значения поля идентификатора `batch`. Это позволяет выгружать документы порционно, запрашивая записи, идущие до последней полученной порции данных.
2. Оператор `$sort` сортирует документы по полю `_id` в порядке убывания. Таким образом, документы будут выводиться от новых к старым.
3. Оператор `$limit` ограничивает выборку до 10 000 документов. Ограничение выборки

выполняется до объединения коллекций, чтобы не тратить ресурсы на обработку ненужных данных.

4. Оператор `$lookup` объединяет текущую коллекцию и коллекцию `logDetails` по следующему алгоритму:
   a. Находятся совпадения полей `detail_id` и `_id`.
   b. Для каждого совпадения в поле `details` исходной коллекции записываются соответствующие данные из коллекции `logDetails`.
5. Оператор `$unwind` создает отдельные документы для каждой записи из поля `details`.

Таким образом, запрос позволяет получить подробные данные по 10 000 последних логов приложений. При этом данные выгружаются порционно благодаря полю идентификатора `batch`.

NATS JetStream

Точка входа типа NATS JetStream предназначена для получения данных с внешних систем через брокер сообщений NATS.

При создании точки входа типа NATS JetStream требуется указать значения следующих параметров:

1. **Адрес подключения**: полный путь для подключения к NATS в формате URL- или IP-адреса с указанием порта. Например: `nats://cluster.nats.com` или `nats://127.0.0.1:4242`.

   Для обмена сообщениями через NATS по умолчанию используется порт 4222.

2. **Максимум сообщений в обработке**: максимальное количество сообщений NATS, отправленных на точку входа, но еще не подтвержденных системой. Должно быть целым числом без знака, в диапазоне от `0` до `100000`. По умолчанию установлено ограничение в 1000 сообщений.
3. **Название потока** (опционально): название существующего потока (хранилища сообщений), из которого необходимо выполнять чтение тем.

   Поток должен быть создан в NATS до настройки точки входа.

4. **Тема**: список тем NATS, из которых необходимо получать сообщения.

   Темы должны быть созданы в NATS до настройки точки входа.

   Чтобы добавить тему, нажмите на кнопку **Добавить**. Отобразится строка добавления темы.

   Чтобы удалить тему, нажмите на кнопку  в строке темы.

   Особенности ввода тем:

   - Название темы вводится в кодировке Unicode.
   - Для задания иерархии и обозначения связанных тем можно использовать точку. Например: `time.us.east`.
   - Допустимо использование подстановочных символов:
     - `*` — единственный токен. Например: `time.*.east`.

- `>` — множественные токены. Данный символ может использоваться только в конечной позиции. Например: `time.us.>`.

5. **Домен** (опционально): название домена NATS JetStream.

   Указанный домен должен быть уникален и не может использоваться в других точках входа, в том числе в точках входа типа R-Vision Endpoint.

6. **Ключ темы** (опционально): ключ, под которым будет записана тема NATS.
7. **Потребитель**: название потребителя, который будет получать сообщения.
8. **Название соединения** (опционально): название соединения с NATS.
9. **Лимит сообщений** (опционально): максимальное количество сообщений, которые могут храниться в буфере. Буфер используется для накопления сообщений, когда они не могут быть направлены в точку входа (например, когда сообщения поступают быстрее, чем точка входа может их обработать). Сообщения находятся в буфере до тех пор, пока их не удастся успешно обработать точкой входа. При достижении лимита новые сообщения не будут сохраняться в буфере.
10. **Тип аутентификации** (опционально): выбор типа аутентификации во внешней системе:
11. **Базовая**: выбор секрета типа **Учетные данные**, хранящего имя пользователя и пароль для авторизации.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

12. **Токен**: выбор секрета типа **Токен**, хранящего токен для авторизации.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

13. **NKey**: выбор секрета типа **NATS Key**, хранящего приватный и публичный ключи для подключения к NATS.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

14. **Формат событий**: формат получения событий из внешней системы. Доступные форматы:
    - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

      ```
      { "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
      }] }
      ```

    - **BYTES**. Данный формат выбран по умолчанию.
    - **GELF**.
    - **INFLUXDB**.
    - **JSON**.
    - **NATIVE**.

- **NATIVE_JSON**.
- **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
- **SYSLOG**.
- **VRL**. При выборе этого формата требуется указать код на языке VRL для обработки каждого поступающего события, а также ввести имя часового пояса для преобразования временных меток, которые не содержат часовой пояс в явном виде.

Netflow



Точка входа поддерживает следующие версии протокола Netflow: v5, v7, v9, IPFIX.

При создании точки входа типа Netflow требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

   Если необходимо подключиться к узлу кластера Kubernetes по порту, отличному от порта точки входа, установите переключатель **Отдельный порт для узла кластера** в активное положение. В появившемся поле укажите порт узла кластера. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

2. **Размер буфера** (опционально): определение максимального размера буфера для входящих сообщений, в байтах.

SFTP

Точка входа типа SFTP позволяет собирать данные текстовых файлов с удаленных ресурсов с помощью протокола SFTP.



Чтение данных происходит построчно. Для корректного чтения файла его последняя строка должна быть пустой, то есть последним байтом должен быть символ `\n`. В противном случае последняя строка данных в файле будет проигнорирована.

SFTP (Secure File Transfer Protocol) — безопасный протокол, позволяющий передавать файлы по сети с использованием SSH-шифрования. SFTP применяется для безопасной передачи **конфиденциальной информации**.



Для корректной работы точки входа типа SFTP необходимо наличие доверенного сертификата. Подробности о добавлении сертификатов приведены в разделе Добавление SSL-сертификата.

При создании точки входа типа SFTP требуется указать значения следующих параметров:

1. **Адрес подключения**: полный адрес для подключения к серверу в виде URI, например, `127.0.0.1:9898/logs` или `somehost:9898/logs`.

По умолчанию используется порт 22.

2. **Тип аутентификации** (опционально): выбор типа аутентификации на удаленном сервере:
   - **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   - **Приватный ключ**: выберите секрет типа **Контейнер PKCS**, хранящий файл сертификата и файл приватного ключа для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

3. **Маска файла**: путь или маска для поиска текстовых файлов на сервере, например, `/var/log/**/*.log`.
4. **Интервал запроса, секунд** (опционально): интервал между запросами в секундах. Интервал включает в себя время выполнения запроса данных. Интервал по умолчанию: 15 секунд.

   Интервал запроса рекомендуется устанавливать выше значения таймаута запроса.

5. **Многострочная обработка** (опционально): объединение нескольких строк лога в одно событие. При включении многострочной обработки необходимо:
   a. Указать шаблон начала события в виде регулярного выражения. Данный шаблон определяет начало нового события, к которому будут добавляться агрегированные строки.

      Примеры регулярных выражений для начала события приведены в разделе Примеры настройки многострочной обработки.

   b. Выбрать способ агрегации строк лога. Доступные варианты:

      - **По шаблону + 1** — к событию добавляются последовательные строки, удовлетворяющие шаблону, а также первая строка, для которой шаблон не сработал.

        Агрегация **По шаблону + 1** полезна, когда запись лога состоит из нескольких строк, которые заканчиваются маркером продолжения, например, обратной косой чертой.

        Данный способ выбран по умолчанию.

      - **По шаблону** — к событию добавляются последовательные строки, удовлетворяющие шаблону.

        Агрегация **По шаблону** полезна, когда строки-продолжения обладают общим признаком, например, ведущими отступами в виде пробелов или табуляции.

      - **Не по шаблону** — к событию добавляются последовательные строки, не удовлетворяющие шаблону.

Агрегация **Не по шаблону** полезна, когда начало записи лога определяется маркером или признаком, например, типом сообщения. В таких случаях строки без этого признака относятся к предыдущей записи.

- **Не по шаблону + 1** — к событию добавляются последовательные строки, не удовлетворяющие шаблону, а также первая строка, для которой шаблон сработал.

  Агрегация **Не по шаблону + 1** полезна, когда сообщение заканчивается маркером (например, точкой с запятой), после которого начинается новое сообщение.



Примеры логов для каждого способа агрегации приведены в разделе Примеры настройки многострочной обработки.

c. Указать шаблон условия агрегации в виде регулярного выражения.



Шаблон условия может не совпадать с шаблоном начала события.



Примеры регулярных выражений для условия агрегации приведены в разделе Примеры настройки многострочной обработки.

d. Ввести таймаут агрегации в секундах, по истечении которого событие отправляется, даже если агрегация не завершена. Таймаут по умолчанию: 1 секунда.

6. **Кодировка** (опционально): кодировка данных, полученных от источника. Позволяет задать кодировку, если требуется специальная обработка текстовых данных.

   Доступные кодировки приведены в разделе Кодировки в точках входа.

7. **Цифровой отпечаток**: количество строк, которые будут считываться с начала файла для создания его уникального идентификатора.

   Если файл содержит меньше строк, чем указано в поле, он будет пропущен.

SMB

Точка входа типа SMB позволяет собирать данные текстовых файлов с удаленных ресурсов с помощью протокола SMB.



Чтение данных происходит построчно. Для корректного чтения файла его последняя строка должна быть пустой, то есть последним байтом должен быть символ \n. В противном случае последняя строка данных в файле будет проигнорирована.

SMB (Server Message Block) — протокол, позволяющий организовать **совместный доступ** к файлам и принтерам в локальных сетях.



SMB по умолчанию применяется для организации общего доступа в ОС Windows. Для работы с SMB в Unix-подобных ОС необходимо использовать программное обеспечение Samba.

При создании точки входа типа SMB требуется указать значения следующих параметров:

1. **Путь к ресурсу**: полный сетевой путь с именем общедоступного ресурса и опциональным подкаталогом, например, `\\server\share\path\to\files`.
2. **Тип аутентификации** (опционально): выбор типа аутентификации для доступа к ресурсу:
   - **Auto** — при выборе данного типа система будет последовательно перебирать способы аутентификации.
   - **Kerberos**.
   - **NTLM**.
3. **Учетные данные**: выбор секрета типа **Учетные данные**, хранящего имя пользователя и пароль для авторизации.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

4. **Версия SMB**: выбор версии протокола SMB:
   - **SMB2**.
   - **SMB3** — при выборе данной версии система первично применяет протокол SMB3. Если применить его невозможно, система автоматически переключается на протокол SMB2 без шифрования.
5. **Маска файла**: маска для поиска текстовых файлов, например, `*.log`.
6. **Интервал запроса, секунд** (опционально): интервал между запросами в секундах. Интервал включает в себя время выполнения запроса данных. Интервал по умолчанию: 15 секунд.

   Интервал запроса рекомендуется устанавливать выше значения таймаута запроса.

7. **Таймаут запроса, секунд** (опционально): максимальное время выполнения запроса в секундах. Если запрос будет выполняться дольше указанного времени, его выполнение будет прекращено. Таймаут по умолчанию: 5 секунд.
8. **Многострочная обработка** (опционально): объединение нескольких строк лога в одно событие. При включении многострочной обработки необходимо:
   a. Указать шаблон начала события в виде регулярного выражения. Данный шаблон определяет начало нового события, к которому будут добавляться агрегированные строки.

      Примеры регулярных выражений для начала события приведены в разделе Примеры настройки многострочной обработки.

   b. Выбрать способ агрегации строк лога. Доступные варианты:

      - **По шаблону + 1** — к событию добавляются последовательные строки, удовлетворяющие шаблону, а также первая строка, для которой шаблон не сработал.

        Агрегация **По шаблону + 1** полезна, когда запись лога состоит из нескольких строк, которые заканчиваются маркером продолжения, например, обратной косой чертой.

Данный способ выбран по умолчанию.

- **По шаблону** — к событию добавляются последовательные строки, удовлетворяющие шаблону.

  Агрегация **По шаблону** полезна, когда строки-продолжения обладают общим признаком, например, ведущими отступами в виде пробелов или табуляции.

- **Не по шаблону** — к событию добавляются последовательные строки, не удовлетворяющие шаблону.

  Агрегация **Не по шаблону** полезна, когда начало записи лога определяется маркером или признаком, например, типом сообщения. В таких случаях строки без этого признака относятся к предыдущей записи.

- **Не по шаблону + 1** — к событию добавляются последовательные строки, не удовлетворяющие шаблону, а также первая строка, для которой шаблон сработал.

  Агрегация **Не по шаблону + 1** полезна, когда сообщение заканчивается маркером (например, точкой с запятой), после которого начинается новое сообщение.



Примеры логов для каждого способа агрегации приведены в разделе Примеры настройки многострочной обработки.

c. Указать шаблон условия агрегации в виде регулярного выражения.



Шаблон условия может не совпадать с шаблоном начала события.



Примеры регулярных выражений для условия агрегации приведены в разделе Примеры настройки многострочной обработки.

d. Ввести таймаут агрегации в секундах, по истечении которого событие отправляется, даже если агрегация не завершена. Таймаут по умолчанию: 1 секунда.

9. **Кодировка** (опционально): кодировка данных, полученных от источника. Позволяет задать кодировку, если требуется специальная обработка текстовых данных.

   Доступные кодировки приведены в разделе Кодировки в точках входа.

10. **Цифровой отпечаток**: количество строк, которые будут считываться с начала файла для создания его уникального идентификатора.

    Если файл содержит меньше строк, чем указано в поле, он будет пропущен.

Socket

При создании точки входа типа Socket (TCP, UDP) требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

Если необходимо подключиться к узлу кластера Kubernetes по порту, отличному от порта точки входа, установите переключатель **Отдельный порт для узла кластера** в активное положение. В появившемся поле укажите порт узла кластера. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

2. **Протокол**: выбор протокола передачи данных. Варианты: TCP, UDP.
3. **Размер буфера** (опционально): определение максимального размера буфера для входящих сообщений, в байтах.

   Настройка размера буфера доступна только при выборе протокола UDP.

4. **Оригинальный IP-адрес**: получение оригинального IP-адреса источника. Если переключатель в активном положении, в поле `host.ip` метаданных базового события сохраняется IP-адрес его источника.

   По умолчанию переключатель в неактивном положении и в `host.ip` сохраняется IP-адрес узла кластера Kubernetes.

   Чтобы обратиться к полю `host.ip` метаданных события с помощью языка VRL, используйте синтаксис `%host.ip`.

5. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Syslog

При создании точки входа типа Syslog требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

   Если необходимо подключиться к узлу кластера Kubernetes по порту, отличному от порта точки входа, установите переключатель **Отдельный порт для узла кластера** в активное положение. В появившемся поле укажите порт узла кластера. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

2. **Протокол**: выбор протокола передачи данных. Варианты: TCP, UDP.
3. **Размер буфера** (опционально): определение максимального размера буфера для входящих

сообщений, в байтах.

4. **Оригинальный IP-адрес**: получение оригинального IP-адреса источника. Если переключатель в активном положении, в поле `host.ip` метаданных базового события сохраняется IP-адрес его источника.

   По умолчанию переключатель в неактивном положении и в `host.ip` сохраняется IP-адрес узла кластера Kubernetes.

   Чтобы обратиться к полю `host.ip` метаданных события с помощью языка VRL, используйте синтаксис `%host.ip`.

5. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Vector

При создании точки входа типа Vector требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

   Если необходимо подключиться к узлу кластера Kubernetes по порту, отличному от порта точки входа, установите переключатель **Отдельный порт для узла кластера** в активное положение. В появившемся поле укажите порт узла кластера. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

2. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

WMI

Точка входа типа WMI (Windows Management Instrumentation) предназначена для получения данных с удаленных хостов на базе ОС Windows. Точка входа собирает события журналов и другие системные данные через WMI-запросы, обеспечивая доступ к информации о процессах, службах и других ресурсах системы.

При создании точки входа WMI требуется указать значения следующих параметров:

1. **Адрес подключения**: имя хоста или IP-адрес удаленного хоста, с которого будут собираться данные, например, localhost, 198.51.100.123.

   Не рекомендуется создавать в одном коллекторе несколько точек WMI, подключенных к одному адресу.

   Если необходимо получать данные из нескольких журналов, выберите их все в одной точке.

2. **Учетные данные**: выбор учетной записи, которая будет использоваться для подключения к удаленному хосту и доступа к журналам. Необходимо, чтобы учетная запись имела права на доступ к журналам безопасности на целевом хосте.

   Учетные данные выбираются из списка секретов.

3. **Журнал**: выбор журналов, из которых необходимо собирать события.

   Доступны следующие журналы:

   - **Application** — журнал приложений.
   - **Security** — журнал безопасности.
   - **System** — журнал системы.
   - **Setup** — журнал установки.
   - **Forwarded Events** — журнал событий передовой диагностики.
   - **Hardware Events** — журнал аппаратного обеспечения.
   - **Microsoft-Windows-Sysmon/Operational** — журнал операций System Monitor.
   - **Microsoft-Windows-PowerShell/Operational** — журнал операций PowerShell.
   - **Microsoft-Windows-Bits-Client/Operational** — журнал службы Background Intelligent Transfer Service.
   - **Microsoft-Windows-DNSServer/Audit** — журнал аудита DNS-запросов.
   - **Microsoft-Windows-TaskScheduler/Operational** — журнал операций Планировщика задач.
   - **Microsoft-Windows-WMI-Activity/Operational** — журнал активности WMI.
   - **Microsoft-Windows-WinRM/Operational** — журнал операций Windows Remote Management.
   - **Microsoft-Windows-AppLocker/EXE and DLL** — журнал применения политик AppLocker для EXE

и DLL.

- **DNS Server** — журнал DNS-сервера.
- **Microsoft-Windows-DHCP Server Events/Operational** — журнал аудита конфигурации DHCP.
- **Microsoft-Windows-DHCP Server Events/FilterNotifications** — журнал событий фильтрации DHCP-сервера.



Название журнала, из которого было получено событие, сохраняется в поле `@wmi.channel` в метаданных события.

4. **Интервал запроса, секунд**: интервал между запросами, включая время выполнения запроса. По умолчанию установлено значение 60 секунд.



Интервал между инициациями запросов охватывает время исполнения запросов, что может фактически продлить интервал между запусками.

5. **Кодировка** (опционально): кодировка данных, полученных от источника. Позволяет задать кодировку, если требуется специальная обработка текстовых данных. Формат: название кодировки, например, UTF-8, Windows-1251.

Примеры настройки многострочной обработки

Для точек входа FTP, SFTP и SMB доступна многострочная обработка собираемых данных из логов. В данном разделе приведены примеры настройки многострочной обработки для различных ситуаций.

- Поиск и обработка сообщений с маркерами продолжения
- Поиск и обработка сообщений с ведущими отступами
- Поиск и обработка сообщений с индикаторами в начале строки
- Поиск и обработка сообщений с маркерами завершения

Поиск и обработка сообщений с маркерами продолжения

Предположим, что сообщения в логах начинаются с указания типа. Нам необходимы только сообщения с типом `Error:`. Сообщения могут состоять из нескольких строк. При этом перенос сообщения на следующую строку определяется наличием маркера продолжения в текущей строке. В качестве маркера продолжения выступает обратная косая черта.

Сообщение с маркерами продолжения

```
Error: Unable to connect to database. \
Please check network settings \
and try again.
```

В таком случае для поиска сообщений удобно задать следующие настройки многострочной обработки:

- **Шаблон начала события**:

```
^Error:
```

Все строки, начинающиеся с `Error:` будут создавать новое событие.

- **Способ агрегации**: По шаблону + 1.

  К событию будут добавляться все строки, удовлетворяющие шаблону условия, а также первая строка, которая не удовлетворяет условию.

- **Шаблон условия**:

  ```
  .*\\$
  ```

  Проверяется наличие обратной косой черты в конце строки.

Таким образом, каждое событие будет начинаться с `Error:` и в него будут добавляться строки, оканчивающиеся на обратную косую черту, а также первая строка без обратной косой черты.

Поиск и обработка сообщений с ведущими отступами

Предположим, что сообщения в логах начинаются с указания типа. Нам необходимы только сообщения с типом `Error:`. Сообщения могут состоять из нескольких строк. При этом продолжение сообщения в следующей строке начинается с ведущих отступов: пробелов или табуляции.

Сообщение с ведущими отступами

```
Error: Unable to connect to database.
    Please check network settings
    and try again.
```

В таком случае для поиска сообщений удобно задать следующие настройки многострочной обработки:

- **Шаблон начала события**:

  ```
  ^Error:
  ```

  Все строки, начинающиеся с `Error:` будут создавать новое событие.

- **Способ агрегации**: По шаблону.

  К событию будут добавляться все строки, удовлетворяющие шаблону условия.

- **Шаблон условия**:

  ```
  ^\s+
  ```

Проверяется наличие пробелов или табуляции в начале строки.

Таким образом, каждое событие будет начинаться с `Error:` и в него будут добавляться строки, начинающиеся с пробелов или табуляции.

Поиск и обработка сообщений с индикаторами в начале строки

Предположим, что сообщения в логах начинаются со специальных индикаторов `[INFO]`, `[WARN]` или `[ERROR]`. Нам нужны только события с индикатором `[ERROR]`.

Сообщения с индикаторами в начале строки

```
[ERROR] Failed to load configuration.
File not found.

[INFO] System started successfully.

[WARN] Low disk space.
```

В примере выше приведены три сообщения разных типов, при этом первое сообщение (с индикатором `[ERROR]`) состоит из двух строк.

В таком случае для поиска сообщений удобно задать следующие настройки многострочной обработки:

- **Шаблон начала события**:

  ```
  ^$$ERROR$$
  ```

  Все строки, начинающиеся с `[ERROR]` будут создавать новое событие.

- **Способ агрегации**: Не по шаблону.

  К событию будут добавляться все строки, не удовлетворяющие шаблону условия.

- **Шаблон условия**:

  ```
  ^$$(INFO|WARN)$$
  ```

  Проверяется наличие индикатора `[INFO]` или `[WARN]` в начале строки.

Таким образом, каждое событие будет начинаться с `[ERROR]` и в него будут добавляться строки до тех пор, пока не появится строка, начинающаяся на `[INFO]` или `[WARN]`.

Поиск и обработка сообщений с маркерами завершения

Предположим, что сообщения в логах начинаются с указания типа. Нам необходимы только сообщения с типом `Error:`. Сообщения могут состоять из нескольких строк. При этом конец сообщения в логах определяется маркером завершения. В качестве маркера завершения выступает

точка с запятой.

Сообщение с маркером завершения

```
Error: User login failed:
Invalid credentials;
```

В таком случае для поиска сообщений удобно задать следующие настройки многострочной обработки:

- **Шаблон начала события**:

  ```
  ^Error:
  ```

  Все строки, начинающиеся с `Error:` будут создавать новое событие.

- **Способ агрегации**: Не по шаблону + 1.

  К событию будут добавляться все строки, не удовлетворяющие шаблону условия, а также первая строка, которая удовлетворяет условию.

- **Шаблон условия**:

  ```
  .*;$
  ```

  Проверяется наличие точки с запятой в конце строки.

Таким образом, каждое событие будет начинаться с `Error:` и в него будут добавляться строки до первой строки, оканчивающейся на точку с запятой, включительно.

Кодировки в точках входа

В таблице ниже приведены форматы кодировки данных, доступные для точек входа FTP, SFTP и SMB.



Более подробную информацию о кодировках можно найти в официальной документации Encoding Standard.

Актуальные кодировки

Стандарт	Кодировка

UTF-8	`unicode-1-1-utf-8`
	`unicode11utf8`
	`unicode20utf8`
	`utf-8`
	`utf8`
	`x-unicode20utf8`

Устаревшие однобайтовые кодировки

Стандарт	Кодировка

IBM866	`866`
	`cp866`
	`csibm866`
	`ibm866`

ISO-8859-2	`csisolatin2`
	`iso-8859-2`
	`iso-ir-101`
	`iso8859-2`
	`iso88592`
	`iso_8859-2`
	`iso_8859-2:1987`
	`l2`
	`latin2`

Стандарт	Кодировка
ISO-8859-3	csisolatin3 iso-8859-3 iso-ir-109 iso8859-3 iso88593 iso_8859-3 iso_8859-3:1988 l3 latin3
ISO-8859-4	csisolatin4 iso-8859-4 iso-ir-110 iso8859-4 iso88594 iso_8859-4 iso_8859-4:1988 l4 latin4
ISO-8859-5	csisolatincyrillic cyrillic iso-8859-5 iso-ir-144 iso8859-5 iso88595 iso_8859-5 iso_8859-5:1988

Стандарт	Кодировка
ISO-8859-6	arabic asmo-708 csiso88596e csiso88596i csisolatinarabic ecma-114 iso-8859-6 iso-8859-6-e iso-8859-6-i iso-ir-127 iso8859-6 iso88596 iso_8859-6 iso_8859-6:1987
ISO-8859-7	csisolatingreek ecma-118 elot_928 greek greek8 iso-8859-7 iso-ir-126 iso8859-7 iso88597 iso_8859-7 iso_8859-7:1987 sun_eu_greek

Стандарт	Кодировка
ISO-8859-8	csiso88598e csisolatinhebrew hebrew iso-8859-8 iso-8859-8-e iso-ir-138 iso8859-8 iso88598 iso_8859-8 iso_8859-8:1988 visual
ISO-8859-8-I	csiso88598i iso-8859-8-i logical
ISO-8859-10	csisolatin6 iso-8859-10 iso-ir-157 iso8859-10 iso885910 l6 latin6
ISO-8859-13	iso-8859-13 iso8859-13 iso885913
ISO-8859-14	iso-8859-14 iso8859-14 iso885914

Стандарт	Кодировка

ISO-8859-15	`csisolatin9`
	`iso-8859-15`
	`iso8859-15`
	`iso885915`
	`iso_8859-15`
	`l9`

ISO-8859-16	`iso-8859-16`

KOI8-R	`cskoi8r`
	`koi`
	`koi8`
	`koi8-r`
	`koi8_r`

KOI8-U	`koi8-ru` `koi8-u`

macintosh	`csmacintosh`
	`mac`
	`macintosh`
	`x-mac-roman`

windows-874	`dos-874`
	`iso-8859-11`
	`iso8859-11`
	`iso885911`
	`tis-620`
	`windows-874`

windows-1250	`cp1250`
	`windows-1250`
	`x-cp1250`

windows-1251	`cp1251`
	`windows-1251`
	`x-cp1251`

Стандарт	Кодировка
windows-1252	ansi_x3.4-1968 ascii cp1252 cp819 csisolatin1 ibm819 iso-8859-1 iso-ir-100 iso8859-1 iso88591 iso_8859-1 iso_8859-1:1987 l1 latin1 us-ascii windows-1252 x-cp1252
windows-1253	cp1253 windows-1253 x-cp1253

Стандарт	Кодировка

windows-1254	`cp1254`
	`csisolatin5`
	`iso-8859-9`
	`iso-ir-148`
	`iso8859-9`
	`iso88599`
	`iso_8859-9`
	`iso_8859-9:1989`
	`l5`
	`latin5`
	`windows-1254`
	`x-cp1254`

windows-1255	`cp1255`
	`windows-1255`
	`x-cp1255`

windows-1256	`cp1256`
	`windows-1256`
	`x-cp1256`

windows-1257	`cp1257`
	`windows-1257`
	`x-cp1257`

windows-1258	`cp1258`
	`windows-1258`
	`x-cp1258`

x-mac-cyrillic	`x-mac-cyrillic` `x-mac-ukrainian`

Устаревшие многобайтовые кодировки для китайского (упрощенного)

Стандарт	Кодировка

GBK	`chinese`
	`csgb2312`
	`csiso58gb231280`
	`gb2312`
	`gb_2312`
	`gb_2312-80`
	`gbk`
	`iso-ir-58`
	`x-gbk`

gb18030	`gb18030`

Устаревшие многобайтовые кодировки для китайского (традиционного)

Стандарт	Кодировка

Big5	`big5`
	`big5-hkscs`
	`cn-big5`
	`csbig5`
	`x-x-big5`

Устаревшие многобайтовые кодировки для японского

Стандарт	Кодировка

EUC-JP	`cseucpkdfmtjapanese`
	`euc-jp`
	`x-euc-jp`

ISO-2022-JP	`csiso2022jp` `iso-2022-jp`

Стандарт	Кодировка

Shift_JIS	`csshiftjis`
	`ms932`
	`ms_kanji`
	`shift-jis`
	`shift_jis`
	`sjis`
	`windows-31j`
	`x-sjis`

Устаревшие многобайтовые кодировки для корейского

Стандарт	Кодировка

EUC-KR	`cseuckr`
	`csksc56011987`
	`euc-kr`
	`iso-ir-149`
	`korean`
	`ks_c_5601-1987`
	`ks_c_5601-1989`
	`ksc5601`
	`ksc_5601`
	`windows-949`

Прочие устаревшие кодировки

Стандарт	Кодировка

replacement	`csiso2022kr`
	`hz-gb-2312`
	`iso-2022-cn`
	`iso-2022-cn-ext`
	`iso-2022-kr`
	`replacement`

Стандарт	Кодировка

UTF-16BE	`unicodefffe`
	
	`utf-16be`

UTF-16LE	`csunicode`
	
	`iso-10646-ucs-2`
	
	`ucs-2`
	
	`unicode`
	
	`unicodefeff`
	
	`utf-16`
	
	`utf-16le`

x-user-defined	`x-user-defined`

Работа с VRL-трансформациями

Данный раздел описывает процесс работы с VRL-трансформациями на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О VRL-трансформациях

VRL-трансформация — это элемент конвейера, который позволяет обрабатывать и преобразовывать события с применением языка VRL.

На диаграмме конфигурации конвейера VRL-трансформация имеет один вход и один выход. Этот элемент конвейера принимает на вход события, проводит их преобразование и на выходе предоставляет уже преобразованные события.

В отличие от нормализаторов, VRL-трансформация позволяет задать простые правила преобразования событий на языке VRL без предварительного создания правил нормализации и привязки к ним.

VRL-трансформация удобна в тех случаях, когда требуется провести несложные модификации в структуре событий, например:

- использовать одну или несколько функций для упрощения дальнейшей работы с событиями;
- добавить, изменить или удалить какое-нибудь поле в структуре событий.

Для проведения более сложных преобразований, применения одних и тех же преобразований в нескольких конвейерах, поддержки версионирования (возможности изменения правил преобразований без перезапуска конвейера) рекомендуется использовать нормализаторы, основанные на правилах нормализации.

Доступные операции над VRL-трансформацией:

- Добавление VRL-трансформации

- Изменение VRL-трансформации
- Удаление VRL-трансформации

Добавление VRL-трансформации

Чтобы добавить VRL-трансформацию на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **VRL-трансформация**. Отобразится окно добавления VRL-трансформации.
4. При необходимости выберите шаблон для автоматического заполнения полей VRL-трансформации. По умолчанию выбран вариант **Без шаблона**.
5. Введите название VRL-трансформации.
6. Введите правило, по которому будет преобразовываться событие, на языке VRL.



В редакторе кода доступны горячие клавиши.



В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

7. При необходимости из выпадающего списка **Обработка ошибок** выберите шину коллектора, в которую будет отправлено событие в случае возникновения ошибок обработки. По умолчанию выбран вариант **События с ошибкой**.
8. Установите флажок **Сохранить как шаблон**, если настройку VRL-трансформации необходимо сохранить в качестве шаблона для дальнейшего использования.
9. Нажмите на кнопку **Добавить**. Новая VRL-трансформация отобразится на диаграмме конфигурации.



Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Пример:

Условие на языке VRL, согласно которому исходное событие будет обогащаться дополнительным полем `responseStatus`. В нем будет устанавливаться значение "Bad request", если в поле `status` приходящего на VRL-трансформацию события установлено значение 401 или 404:

Пример VRL-трансформации, язык VRL

```
if .status == "404" || .status == "401" {
.responseStatus = "Bad request"
}
```

Изменение VRL-трансформации

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление VRL-трансформации

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Работа с фильтрами

Данный раздел описывает процесс работы с фильтрами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О фильтрах

Фильтр — это элемент конвейера, который позволяет сократить объем обрабатываемых событий путем их отсеивания на основе условий, заданных на языке VRL. На диаграмме конфигурации конвейера фильтр имеет один вход и один выход. На входе каждое событие проверяется по условию фильтра и появляется на выходе только в том случае, если условие выполняется. В противном случае событие отсеивается и не участвует в дальнейшей обработке.

Доступные операции над фильтром:

- Добавление фильтра
- Изменение фильтра
- Удаление фильтра

Добавление фильтра

Чтобы добавить фильтр на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Фильтр**. Отобразится окно добавления фильтра.
4. При необходимости выберите шаблон для автоматического заполнения полей фильтра. По умолчанию выбран вариант **Без шаблона**.
5. Введите название фильтра.
6. Задайте условие фильтрации входящих событий на языке VRL.

Условия обычно формулируются через имена полей и их значения, и для их определения используются математические и логические операторы.



В редакторе кода доступны горячие клавиши.



В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

7. Установите флажок **Сохранить как шаблон**, если настройку фильтра необходимо сохранить в качестве шаблона для дальнейшего использования.
8. Нажмите на кнопку **Добавить**. Новый фильтр отобразится на диаграмме конфигурации.



Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Пример фильтра

Условие фильтра, пропускающего только события, поля `externalId` и `categoryOutcome` которых имеют значения "4648" и "Success" соответственно:

```
.externalId == "4648" && .categoryOutcome == "Success"
```

Изменение фильтра

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление фильтра

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Работа с маршрутизаторами

Данный раздел описывает процесс работы с маршрутизаторами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О маршрутизаторах

Маршрутизатор — это элемент конвейера, который позволяет на основе заданных фильтров (маршрутов) направлять события на разные этапы обработки в зависимости от их содержимого или

других условий. На диаграмме конфигурации конвейера маршрутизатор имеет один вход и несколько выходов: по одному выходу на каждый настроенный в нем маршрут. На входе каждое событие проверяется по условиям маршрутов и появляется на выходах тех маршрутов, где условие выполняется.

Если событие не удовлетворяет условиям ни одного из маршрутов, то оно направляется на выход маршрутизатора **По умолчанию** (если установлен флажок **Добавить маршрут по умолчанию**) или удаляется (если флажок **Добавить маршрут по умолчанию** не установлен).

Доступные операции над маршрутизатором:

- Добавление маршрутизатора
- Изменение маршрутизатора
- Удаление маршрутизатора
- Настройка маршрутов

Добавление маршрутизатора

Чтобы добавить маршрутизатор на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Маршрутизатор**. Отобразится окно добавления маршрутизатора.
4. Введите название маршрутизатора.

   Для маршрутизатора недопустимо использование названий "_unmatched" и "_default".

5. Настройте список маршрутов маршрутизатора.
6. При необходимости установите флажок **Добавить маршрут по умолчанию**. В этом случае к маршрутизатору будет добавлен маршрут **По умолчанию**, на выход которого будут попадать те события, которые не удовлетворяют условиям остальных маршрутов.

   При неустановленном флажке события, не удовлетворяющие условиям ни одного из маршрутов, будут удаляться.

7. Нажмите на кнопку **Добавить**. Новый маршрутизатор отобразится на диаграмме конфигурации.

Изменение маршрутизатора

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление маршрутизатора

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Настройка маршрутов

Маршруты предназначены для фильтрации поступающих в маршрутизатор событий по различным условиям, что позволяет более гибко выстраивать процесс обработки событий.

На входе маршрутизатора события проверяются по условиям маршрутов и отправляются на выходы тех маршрутов, условиям которых удовлетворяют.

Если событие не удовлетворяет условиям ни одного из маршрутов, то оно направляется на выход маршрутизатора **По умолчанию** (если установлен флажок **Добавить маршрут по умолчанию**) или удаляется (если флажок **Добавить маршрут по умолчанию** не установлен).

В нижней части окна добавления или редактирования маршрутизатора отображается таблица имеющихся в нем маршрутов. Таблица содержит следующие столбцы:

- **Название маршрута** — название маршрута, используемое для его идентификации в системе.
- **Условие** — условие фильтрации исходных событий, заданное на языке VRL.



В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

Доступные операции над маршрутами:

- Добавление маршрута
- Изменение маршрута
- Удаление маршрута
- Удаление маршрута

Добавление маршрута

Чтобы добавить маршрут:

1. Нажмите на кнопку **Добавить маршрут**. Система отобразит окно добавления маршрута.
2. Введите название маршрута.
3. Введите условие фильтрации исходных событий на языке VRL.

   Пример условия

   ```
   .externalId == "4648" && .categoryOutcome == "Success"
   ```

   В редакторе кода доступны горячие клавиши.

4. Нажмите на кнопку **Добавить**. Маршрут будет добавлен в список маршрутов маршрутизатора.

Изменение маршрута

Чтобы изменить маршрут:

1. Нажмите на иконку редактирования в строке требуемого маршрута в списке. Система отобразит окно изменения этого маршрута.
2. Внесите изменения в требуемые поля.
3. Нажмите на кнопку **Сохранить**. Измененные данные маршрута будут сохранены.

Удаление маршрута

Чтобы удалить маршрут, нажмите на иконку удаления в строке требуемого маршрута в списке. Маршрут будет удален из списка маршрутов маршрутизатора без подтверждения действия.

Работа с корреляторами

Данный раздел описывает процесс работы с корреляторами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О корреляторах

Коррелятор — это элемент конвейера, который представляет собой комплекс правил корреляции. Правила корреляции анализируют поток событий и позволяют распознавать среди них определенные последовательности. Эти правила также могут инициировать действия после обнаружения, такие как создание корреляционных событий и оповещений.

Коррелятор позволяет применять к событиям несколько правил корреляции. Порядок применения правил определяется последовательностью их добавления в коррелятор. Таким образом, события проходят через набор правил в установленном порядке, что позволяет осуществлять сложные сценарии корреляции данных.

На диаграмме конфигурации конвейера коррелятор имеет один вход и один выход. Коррелятор на входе применяет правила корреляции к потоку событий. На выходе выдает корреляционные события.

Доступные операции над коррелятором:

- Добавление коррелятора
- Изменение коррелятора
- Удаление коррелятора
- Настройка коррелятора

Режим распределенной корреляции



Императивные правила корреляции не поддерживаются в режиме распределенной корреляции.

Распределенная корреляция использует ресурсы нескольких узлов в кластере для параллельной обработки событий. Данный режим позволяет распределять нагрузку между несколькими экземплярами предпроцессора и коррелятором, увеличивая общую производительность коллектора.

Распределенный коррелятор состоит из двух компонентов: предпроцессора и коррелятора.

Предпроцессор предназначен для первичной обработки событий, поступающих на коррелятор. В предпроцессоре происходит фильтрация событий и создание псевдонимов (aliases). Псевдонимы — это ссылки на события, используемые для упрощения их обработки и анализа. Предпроцессор создает и выпускает корреляционные события. Предпроцессоров может быть несколько, но должен быть минимум один.

Коррелятор в составе распределенного коррелятора отвечает за группировку и составление цепочек событий. Цепочки событий представляют собой последовательности взаимосвязанных событий, обнаруженных на основе логики, определенной в поле `join` правила корреляции. В составе распределенного коррелятора может быть только один коррелятор.

Операции полей `join` и `group` декларативного правила вычисляются на корреляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. События могут испускаться как предпроцессором, так и коррелятором в зависимости от конфигурации правила корреляции.

Для работы коррелятора в распределенном режиме необходимо активировать настройку **Распределенный коррелятор** и выделить дополнительные ресурсы в настройках коллектора. Изменение этой настройки требует перезапуска коллектора.

Проверка исключений

Для управления обработкой корреляционных событий коррелятор поддерживает механизм исключений на основе черного и белого списков. Этот механизм позволяет по-разному обрабатывать события, связанные с запрещенными и доверенными сущностями, например, IP-адресами, пользователями или хостами.

Черный список служит для указания подозрительных сущностей, для которых результаты корреляции должны обязательно обрабатываться. Если событие соответствует записи в черном списке, проверка по белому списку для такого события не выполняется.

Белый список используется для указания доверенных сущностей. Если событие соответствует записи в белом списке, это событие не обрабатывается далее.

При проверке исключений коррелятор действует в следующем порядке:

1. Событие проверяется по черному списку. Если есть совпадение, событие обрабатывается без проверки по белому списку.
2. Если совпадений в черном списке нет, событие проверяется по белому списку. Если есть совпадение, событие далее не обрабатывается.
3. Если нет совпадения в белом списке, событие обрабатывается в обычном порядке.

Черный и белый списки являются системными активными списками. Просмотреть и добавить в них записи можно в разделе **Ресурсы → Активные списки** веб-интерфейса системы.

Добавлять исключения можно также на основе событий в разделе **Поиск**.

Добавление коррелятора

Чтобы добавить коррелятор на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.

2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Коррелятор**. Отобразится окно добавления коррелятора.
4. Введите название коррелятора.
5. Задайте интервал очистки событий в секундах. По истечении указанного периода в императивных правилах корреляции будет происходить очистка событий с истекшим временем жизни. Время жизни задается в поле `expire_after_sec` в структуре правила.

   По умолчанию интервал очистки равен 300 секундам.
6. Если нужно проверять события по черному и белому спискам, установите флажок **Проверка исключений**.
7. Добавьте в коррелятор правила корреляции.

   Если коллектор, на котором размещен коррелятор, настроен для работы в режиме распределенной корреляции, он не поддерживает работу императивных правил корреляции.

   Чтобы добавить правило корреляции:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило корреляции.
   b. Установите флажок слева от правила корреляции, которое необходимо добавить в коррелятор.
8. Нажмите на кнопку **Добавить**. Новый коррелятор отобразится на диаграмме конфигурации.



Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Изменение коррелятора

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление коррелятора

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Настройка коррелятора

Настройка коррелятора выполняется при его добавлении или изменении.

В окне настроек коррелятора отображается список правил корреляции в выбранном каталоге. Список представлен в виде таблицы со следующими столбцами:

- **ID** — уникальный идентификационный код правила корреляции, присваиваемый системой автоматически при его создании.
- **Название** — имя правила корреляции, используемое для его идентификации в системе.
- **Теги** — пользовательские теги, присвоенные данному правилу корреляции.
- **Версия** — версия правила корреляции, представленная в формате SemVer v2.0.

При работе с таблицей правил корреляции доступны следующие операции:

- Поиск правил по полям **ID**, **Название** и **Теги**.
- Фильтрация правил по тегам. Теги выбираются из выпадающего списка.

  Также доступна фильтрация по наличию правил в коррелляторе с помощью флажка **Добавлено в коррелятор**. При установке флажка в таблице отображаются только правила, уже добавленные в коррелятор.

- Сортировка правил по полям **ID**, **Название** и **Версия**.

В окне настроек доступен функционал просмотра и добавления правил корреляции, а также обновления их версии.

Доступные операции по настройке коррелятора:

- Просмотр правила корреляции
- Добавление правил корреляции
- Обновление версии правила корреляции
- Изменение версии правила корреляции

Просмотр правила корреляции

Чтобы просмотреть конфигурацию правила корреляции из окна настройки коррелятора, нажмите на ID соответствующего правила в списке. Система отобразит окно просмотра выбранного правила на новой вкладке браузера.

Добавление правил корреляции

Чтобы добавить правило корреляции в коррелятор:

1. Раскройте дерево каталогов в левой части окна настроек коррелятора и выберите каталог, который содержит нужное правило корреляции.
2. Установите флажок слева от правила корреляции, которое необходимо добавить в коррелятор.

Обновление версии правила корреляции

Обновление версий возможно только для правил, которые уже были добавлены в коррелятор ранее.

Вы можете обновить правила следующими способами:

- Обновить одно конкретное правило. Для этого нажмите на кнопку в правой части строки правила.
- Обновить сразу все правила. Для этого нажмите на кнопку справа от поля поиска над таблицей.



Кнопка отображается, если существует более новая версия добавленного правила.



Вы можете также выбрать последнюю версию правила в выпадающем списке версий.

Изменение версии правила корреляции

Чтобы установить другую версию правила корреляции:

- Нажмите на кнопку редактирования в правой части строки правила в списке.
- Выберите нужную версию в выпадающем списке и нажмите на кнопку **Сохранить**. В рамках коррелятора будет установлена выбранная версия правила.

Работа с нормализаторами

Данный раздел описывает процесс работы с нормализаторами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О нормализаторах

Нормализатор — это элемент конвейера, который представляет собой комплекс правил нормализации. Каждое такое правило состоит из двух основных компонентов: кода нормализации на языке VRL и фильтра. При наличии фильтра выполнение соответствующего правила нормализации инициируется только при соблюдении условий этого фильтра для входящего события.

Нормализатор применяет к каждому событию только одно правило нормализации из добавленных — то, которое первым удовлетворило условиям фильтрации. Порядок проверки фильтров определяется последовательностью добавления правил в нормализатор.



Пусть в нормализатор добавлены правила нормализации в следующем порядке:

1. правило нормализации А;
2. правило нормализации Б;
3. правило нормализации В.

Если для поступившего в нормализатор события условия фильтрации удовлетворяются во всех правилах, то применено будет только правило А,

поскольку оно стоит первым в очереди на проверку фильтров. Остальные правила будут проигнорированы.

Если условия фильтрации удовлетворяются только для правил Б и В, то в соответствии с порядком будет применено правило Б, а правило В будет проигнорировано.

На диаграмме конфигурации конвейера нормализатор имеет один вход и один выход. Нормализатор на входе принимает события, преобразует их и выдает преобразованные события на выходе.



Использование нормализаторов необходимо для приведения событий к структуре, ожидаемой конечной точкой. В противном случае возможны потери в потоке событий, если структура событий не соответствует требуемой структуре.

Доступные операции над нормализатором:

- Добавление нормализатора
- Изменение нормализатора
- Удаление нормализатора
- Настройка нормализатора

Добавление нормализатора

Чтобы добавить нормализатор на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Нормализатор**. Отобразится окно добавления нормализатора.
4. Введите название нормализатора.
5. При необходимости из выпадающего списка **Обработка ошибок** выберите шину коллектора, в которую будет отправлено событие в случае возникновения ошибок обработки. По умолчанию выбран вариант **События с ошибкой**.
6. Если требуется отслеживать состояние источника данных, выберите политику аудита источников. В противном случае установите переключатель **Аудит** в неактивное положение.

   По умолчанию выбрана системная **Политика аудита источников по маске**.



Если политика аудита источников включена в нескольких нормализаторах на конвейере, то в метриках по источнику количество событий будет умножено на количество нормализаторов с аудитом.

Рекомендуется включать аудит только в одном нормализаторе на конвейере.



При выборе политики с методом расчета **По среднему EPS** системе необходимо собрать минимальное количество данных по источнику для корректного

расчета среднего значения EPS. После включения политики система ожидает 24 часа, накапливая данные. В этот период нарушения политики по отклонениям EPS не фиксируются.

7. Добавьте в нормализатор правила нормализации.

   Чтобы добавить правило нормализации:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило нормализации.

   b. Установите флажок слева от правила нормализации, которое необходимо добавить в нормализатор.

8. Нажмите на кнопку **Добавить**. Новый нормализатор отобразится на диаграмме конфигурации.



Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Изменение нормализатора

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление нормализатора

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Настройка нормализатора

Настройка нормализатора выполняется при его добавлении или изменении.

В окне настроек нормализатора отображается список правил нормализации в выбранном каталоге. Список представлен в виде таблицы со следующими столбцами:

- **ID** — уникальный идентификационный код правила нормализации, присваиваемый системой автоматически при его создании.
- **Название** — имя правила нормализации, используемое для его идентификации в системе.
- **Теги** — пользовательские теги, присвоенные данному правилу нормализации.
- **Версия** — версия правила нормализации, представленная в формате SemVer v2.0.

При работе с таблицей правил нормализации доступны следующие операции:

- Поиск правил по полям **ID**, **Название** и **Теги**.

- Фильтрация правил по тегам. Теги выбираются из выпадающего списка.

  Также доступна фильтрация по наличию правил в нормализаторе с помощью флажка **Добавлено в нормализатор**. При установке флажка в таблице отображаются только правила, уже добавленные в нормализатор.

- Сортировка правил по полям **ID**, **Название** и **Версия**.

В окне настроек доступен функционал просмотра и добавления правил нормализации, а также обновления их версии.

Доступные операции по настройке нормализатора:

- Просмотр правила нормализации
- Добавление правил нормализации
- Обновление версии правила нормализации
- Изменение версии правила нормализации

Просмотр правила нормализации

Чтобы просмотреть конфигурацию правила нормализации из окна настройки нормализатора, нажмите на ID соответствующего правила в списке. Система отобразит окно просмотра выбранного правила на новой вкладке браузера.

Добавление правил нормализации

Чтобы добавить правило нормализации в нормализатор:

1. Раскройте дерево каталогов в левой части окна настроек нормализатора и выберите каталог, который содержит нужное правило нормализации.
2. Установите флажок слева от правила нормализации, которое необходимо добавить в нормализатор.

Обновление версии правила нормализации

Обновление версий возможно только для правил, которые уже были добавлены в нормализатор ранее.

Вы можете обновить правила следующими способами:

- Обновить одно конкретное правило. Для этого нажмите на кнопку ⟳ в правой части строки правила.
- Обновить сразу все правила. Для этого нажмите на кнопку ⟳ справа от поля поиска над таблицей.



Кнопка ⟳ отображается, если существует более новая версия добавленного правила.



Вы можете также выбрать последнюю версию правила в выпадающем списке

версий.

Изменение версии правила нормализации

Чтобы установить другую версию правила нормализации:

- Нажмите на кнопку редактирования в правой части строки правила в списке.
- Выберите нужную версию в выпадающем списке и нажмите на кнопку **Сохранить**. В рамках нормализатора будет установлена выбранная версия правила.

Работа с агрегаторами

Данный раздел описывает процесс работы с агрегаторами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

Об агрегаторах

Агрегатор — это элемент конвейера, который использует правила агрегации для объединения и анализа групп событий безопасности. Каждое правило агрегации включает в себя критерии для группировки событий, условия начала и завершения агрегации, а также стратегии агрегации для каждого поля. Агрегация инициируется в соответствии с условиями фильтрации, определенными в правилах, и преобразует входящий поток событий в агрегированные данные.

Агрегатор применяет к каждому событию только одно правило агрегации из добавленных — то, которое первым удовлетворило условиям фильтрации. Порядок проверки фильтров определяется последовательностью добавления правил в агрегатор.

Пусть в агрегатор добавлены правила агрегации в следующем порядке:

1. правило агрегации А;
2. правило агрегации Б;
3. правило агрегации В.

Если для поступившего в агрегатор события условия фильтрации удовлетворяются во всех правилах, то применено будет только правило А, поскольку оно стоит первым в очереди на проверку фильтров. Остальные правила будут проигнорированы.

Если условия фильтрации удовлетворяются только для правил Б и В, то в соответствии с порядком будет применено правило Б, а правило В будет проигнорировано.

На диаграмме конфигурации конвейера агрегатор имеет один вход и один выход. Агрегатор на входе принимает поток событий, производит их агрегацию и выдает агрегированные события на выходе.

Доступные операции над агрегатором:

- Добавление агрегатора
- Изменение агрегатора

- Удаление агрегатора
- Настройка агрегатора

Добавление агрегатора

Чтобы добавить агрегатор на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Агрегатор**. Отобразится окно добавления агрегатора.
4. Введите название агрегатора.
5. Добавьте в агрегатор правила агрегации.

   Чтобы добавить правило агрегации:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило агрегации.

   b. Установите флажок слева от правила агрегации, которое необходимо добавить в агрегатор.
6. Нажмите на кнопку **Добавить**. Новый агрегатор отобразится на диаграмме конфигурации.



Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Изменение агрегатора

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление агрегатора

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Настройка агрегатора

Настройка агрегатора выполняется при его добавлении или изменении.

В окне настроек агрегатора отображается список правил агрегации в выбранном каталоге. Список представлен в виде таблицы со следующими столбцами:

- **ID** — уникальный идентификационный код правила агрегации, присваиваемый системой автоматически при его создании.
- **Название** — имя правила агрегации, используемое для его идентификации в системе.
- **Теги** — пользовательские теги, присвоенные данному правилу агрегации.
- **Версия** — версия правила агрегации, представленная в формате SemVer v2.0.

При работе с таблицей правил агрегации доступны следующие операции:

- Поиск правил по полям **ID**, **Название** и **Теги**.
- Фильтрация правил по тегам. Теги выбираются из выпадающего списка.

  Также доступна фильтрация по наличию правил в агрегаторе с помощью флажка **Добавлено в агрегатор**. При установке флажка в таблице отображаются только правила, уже добавленные в агрегатор.

- Сортировка правил по полям **ID**, **Название** и **Версия**.

В окне настроек доступен функционал просмотра и добавления правил агрегации, а также обновления их версии.

Доступные операции по настройке агрегатора:

- Просмотр правила агрегации
- Добавление правил агрегации
- Обновление версии правила агрегации
- Изменение версии правила агрегации

Просмотр правила агрегации

Чтобы просмотреть конфигурацию правила агрегации из окна настройки агрегатора, нажмите на ID соответствующего правила в списке. Система отобразит окно просмотра выбранного правила на новой вкладке браузера.

Добавление правил агрегации

Чтобы добавить правило агрегации в агрегатор:

1. Раскройте дерево каталогов в левой части окна настроек агрегатора и выберите каталог, который содержит нужное правило агрегации.
2. Установите флажок слева от правила агрегации, которое необходимо добавить в агрегатор.

Обновление версии правила агрегации

Обновление версий возможно только для правил, которые уже были добавлены в агрегатор ранее.

Вы можете обновить правила следующими способами:

- Обновить одно конкретное правило. Для этого нажмите на кнопку  в правой части строки правила.

- Обновить сразу все правила. Для этого нажмите на кнопку справа от поля поиска над таблицей.



Кнопка отображается, если существует более новая версия добавленного правила.



Вы можете также выбрать последнюю версию правила в выпадающем списке версий.

Изменение версии правила агрегации

Чтобы установить другую версию правила агрегации:

- Нажмите на кнопку редактирования в правой части строки правила в списке.
- Выберите нужную версию в выпадающем списке и нажмите на кнопку **Сохранить**. В рамках агрегатора будет установлена выбранная версия правила.

Работа с сервисами оповещений

Данный раздел описывает процесс работы с сервисом оповещений на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О сервисе оповещений

Сервис оповещений — это элемент конвейера, который позволяет настраивать связи правил сегментации и правил корреляции. Такие связи определяют, как корреляционные события будут распределяться по оповещениям.

С помощью сервиса оповещений можно настроить отправку корреляционных событий, создаваемых при срабатывании правила корреляции, в разные оповещения в зависимости от условий, заданных правилами сегментации.

Для сервиса оповещений можно настроить правило сегментации по умолчанию. Оно автоматически применяется к потоку корреляционных событий, которые не удовлетворяют условиям, заданным связями с правилами сегментации, и распределяет этот поток по оповещениям. При отсутствии такого правила все корреляционные события, сгенерированные одним и тем же правилом корреляции и не удовлетворяющие условиям правил сегментации, будут объединяться в одно и то же оповещение.

Оповещения, созданные с помощью сервиса оповещений, можно отправлять через интеграцию во внешние системы.

На диаграмме конфигурации конвейера сервис оповещений имеет один вход и не имеет выходов. Сервис оповещений на входе принимает поток корреляционных событий и на основе заданных связей с правилами сегментации определяет, к какому оповещению должно быть отправлено то или иное корреляционное событие.

Название оповещения формируется на основе логики, определенной в поле `alert_name` правила сегментации. Если поле `alert_name` не определено, оповещению будет автоматически присвоено название правила корреляции, которое первым привело к его срабатыванию. Если правило

корреляции не ассоциировано с правилом сегментации, название оповещения будет автоматически сформировано из названия правила корреляции.

Доступные операции над сервисом оповещений:

- Добавление сервиса оповещений
- Изменение сервиса оповещений
- Удаление сервиса оповещений
- Настройка сервиса оповещений

Добавление сервиса оповещений



Сервис оповещений недоступен для конвейеров в вынесенных коллекторах.

Чтобы добавить сервис оповещений на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Сервис оповещений**. Отобразится окно добавления сервиса оповещений.
4. Введите название сервиса оповещений.
5. Выберите из выпадающего списка хранилище с событиями, на основе которых будут формироваться оповещения.
6. При необходимости выберите интеграции для отправки оповещений во внешние системы.
7. При необходимости выберите правило сегментации по умолчанию, которое будет применяться ко всем несвязанным правилам корреляции.

   Если правило по умолчанию не выбрано, то все корреляционные события, сгенерированные одним и тем же правилом корреляции и не удовлетворяющие условиям правил сегментации, будут объединяться в одно и то же оповещение.

8. Настройте связи с правилами сегментации.
9. Нажмите на кнопку **Добавить**. Новый сервис оповещений отобразится на диаграмме конфигурации.

Изменение сервиса оповещений

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление сервиса оповещений

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Настройка сервиса оповещений

Настройка сервиса оповещений выполняется при его добавлении или изменении.

В левой части окна настроек сервиса оповещений отображается список связей с правилами сегментации. При нажатии на связь, в правой части окна настроек отображается дерево каталогов со списком связанных правил корреляции.

В окне настроек доступен функционал просмотра правил сегментации и корреляции, добавления, изменения и удаления связей с правилами сегментации, а также обновления их версии.

Доступные операции по настройке сервиса оповещений:

- Просмотр правила сегментации
- Просмотр правила корреляции
- Добавление связи с правилом сегментации
- Изменение связи с правилом сегментации
- Удаление связи с правилом сегментации
- Обновление правила сегментации
- Изменение версии правила сегментации

Просмотр правила сегментации

Чтобы просмотреть конфигурацию правила сегментации из окна настройки сервиса оповещений:

- В режиме черновика: нажмите на кнопку ⋮ в строке требуемой связи и выберите опцию **Посмотреть правило**.
- В режиме актуальной конфигурации: нажмите на кнопку **Посмотреть правило** (↗) в строке требуемой связи.

Система отобразит окно просмотра правила сегментации выбранной связи на новой вкладке браузера.

Просмотр правила корреляции

Чтобы просмотреть конфигурацию правила корреляции, связанного с правилом сегментации, из окна настройки сервиса оповещений, нажмите на кнопку **Посмотреть правило** (↗) в строке требуемого правила корреляции. Система отобразит окно просмотра выбранного правила на новой вкладке браузера.

Добавление связи с правилом сегментации

Чтобы добавить связь с правилом сегментации:

1. Нажмите на кнопку **Добавить связь**. Откроется окно добавления связи.
2. Выберите правило сегментации из выпадающего списка. В окне отобразится дерево всех правил корреляции в системе, которые еще не имеют связей с другими правилами сегментации текущего сервиса оповещений.

- Одно правило корреляции может быть связано только с одним правилом сегментации.
- Одно правило сегментации может быть связано с любым количеством правил корреляции.

3. Выберите требуемые правила корреляции, установив флажок слева от их названий.
4. Нажмите на кнопку **Добавить**. Добавленная связь с правилом сегментации отобразится в окне настройки сервиса оповещений.

Изменение связи с правилом сегментации

Чтобы изменить связь с правилом сегментации:

1. Нажмите на кнопку ⋮ в строке требуемой связи и выберите опцию **Изменить связь**. Откроется окно изменения связи.
2. Внесите изменения в настройки связи.
3. Нажмите на кнопку **Сохранить**. Изменения, внесенные в связь с правилом сегментации, будут сохранены.

Удаление связи с правилом сегментации

Чтобы удалить связь с правилом сегментации:

1. Нажмите на кнопку ⋮ в строке требуемой связи и выберите опцию **Удалить связь**. Откроется окно подтверждения удаления.
2. Нажмите на кнопку **Удалить**. Связь с правилом сегментации будет удалена.

Обновление правила сегментации

Обновление версий возможно только для правил, которые уже были добавлены в сервис оповещений ранее.

Вы можете обновить правила следующими способами:

- Обновить одно конкретное правило. Для этого нажмите на кнопку ↻ в правой части строки правила.
- Обновить сразу все правила. Для этого нажмите на кнопку ↻ справа от поля поиска над таблицей.

Кнопка ↻ отображается, если существует более новая версия добавленного правила.

Чтобы отобразить в списке только связи с правилами сегментации, для которых существуют новые версии, установите флажок **Новая версия**.

Аналогичным образом можно обновить версию правила сегментации, установленного по умолчанию.



Вы можете также выбрать последнюю версию правила в выпадающем списке версий.



При обновлении версии правила сегментации, корреляционные события будут генерировать новое оповещение, а не привязываться к старому.

Изменение версии правила сегментации

Чтобы установить другую версию правила сегментации:

- Нажмите на номер версии в правой части строки правила в списке.
- Выберите нужную версию в выпадающем списке и нажмите на кнопку **Сохранить**. В рамках сервиса оповещения будет установлена выбранная версия правила.

Работа со сбором метрик

Данный раздел описывает процесс работы со сбором метрик на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О сборе метрик

Сбор метрик — это элемент конвейера, который собирает метрики различных элементов конвейеров для их дальнейшей обработки и анализа.

Например, с помощью сбора метрик можно отслеживать поступление событий от источника на конвейер и выявлять ситуации, когда точка входа перестает получать события. Это позволяет своевременно детектировать и исправлять ошибки в работе конвейера.

На диаграмме конфигурации конвейера сбор метрик имеет один выход и не имеет входов. На выходе выдает события, содержащие собранные метрики.

Доступные операции над сбором метрик:

- Добавление сбора метрик
- Изменение сбора метрик
- Удаление сбора метрик

Добавление сбора метрик

Чтобы добавить сбор метрик на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент**. Отобразится выпадающее меню элементов конвейера.
4. Выберите пункт **Сбор метрик**. Отобразится окно добавления сбора метрик.
5. Введите название сбора метрик.
6. Задайте интервал между сборами метрик в секундах в поле **Период сбора метрик**. Период

можно ввести вручную или задать с помощью кнопок — и +.

7. Выберите элементы для сбора метрик из дерева элементов. Для выбора доступны элементы всех конвейеров в рамках коллектора.

   Чтобы выбрать элемент, установите флажок слева от его названия в списке.

   Элементы в дереве сгруппированы по конвейерам. Чтобы просмотреть список элементов нужного конвейера, нажмите на кнопку ⌄ справа от его названия.

   Чтобы скрыть отображенный список элементов конвейера, нажмите на кнопку ⌃ справа от его названия.

8. Нажмите на кнопку **Добавить**. Новый сбор метрик отобразится на диаграмме конфигурации.

Изменение сбора метрик

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление сбора метрик

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Работа с конечными точками

Данный раздел описывает процесс работы с конечными точками на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О конечных точках

Конечная точка — это элемент конвейера, который отвечает за отправку событий. Тип конечной точки зависит от пункта назначения, в который пересылаются события. На диаграмме конфигурации конвейера конечная точка имеет один вход и не имеет выходов.

В системе доступны следующие типы конечных точек:

- Хранилище событий
- Внешняя система
- Глобальная шина



Если структура события отличается от структуры, ожидаемой выбранным типом конечной точки, то данное событие отбрасывается. Чтобы минимизировать потери в потоке событий, необходимо использовать нормализаторы для приведения событий к требуемой структуре.

Доступные операции над конечной точкой:

- Добавление конечной точки
- Изменение конечной точки

- Удаление конечной точки
- Настройка конечной точки

Добавление конечной точки

Чтобы добавить конечную точку на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная точка**. Отобразится окно добавления конечной точки.
4. При необходимости выберите шаблон для автоматического заполнения полей конечной точки. По умолчанию выбран вариант **Без шаблона**.
5. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):
    - Название конечной точки.
    - Тип конечной точки.
    - Формат конечной точки.
6. Установите флажок **Сохранить как шаблон**, если настройку конечной точки необходимо сохранить в качестве шаблона для дальнейшего использования.
7. Нажмите **Добавить**. Новая конечная точка отобразится на диаграмме конфигурации.

Изменение конечной точки

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление конечной точки

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Настройка конечной точки

Набор параметров конечной точки отличается в зависимости от ее типа. В системе доступны следующие типы конечных точек:

- Хранилище событий
- Внешняя система
- Глобальная шина

Хранилище событий

Конечная точка типа **Хранилище событий** позволяет сохранять события в выбранное хранилище.



- Конечная точка типа **Хранилище событий** недоступна для конвейеров в вынесенных коллекторах. Пример обработки событий, собранных в вынесенном

коллекторе, рассмотрен в разделе Сбор событий из вынесенного коллектора.

- Если структура события отличается от структуры, ожидаемой выбранным типом конечной точки, то данное событие отбрасывается. Чтобы минимизировать потери в потоке событий, необходимо использовать нормализаторы для приведения событий к требуемой структуре.

  Для конечной точки типа **Хранилище событий** ожидаемая структура определяется моделью события, лежащей в основе выбранного хранилища.



Вы можете просматривать события хранилища в разделе **Поиск** или визуализировать их данные с помощью дашбордов.

При создании конечной точки типа **Хранилище событий** требуется задать следующие параметры:

1. **Хранилище событий**: выбор хранилища событий, в которое будут передаваться события.
2. **Режим отладки**: включение режима отладки. В режиме отладки, поля, которые не соответствуют модели события выбранного хранилища, будут принудительно заполнены значениями по умолчанию, при этом в лог будет записано сообщение о наличии проблем в полях исходного события.



Поля модели, которые отсутствуют в исходном событии, заполняются значениями по умолчанию в зависимости от типа:

- `Bool` — `false`;
- `Float` — `0.0`;
- `UUID` — `00000000-0000-0000-0000-000000000000`.

Перед тем как переслать события в хранилище, система автоматически производит их нормализацию:

- С помощью функции `compact` удаляются пустые поля, если они присутствуют в данных.

  ```
  . = compact(.) ?? .
  ```

- С помощью функции `rv_ipv6_to_ipv4` адреса IPv6 преобразуются в IPv4, если это возможно.

Внешняя система

Конечная точка типа **Внешняя система** позволяет отправлять события во внешнюю систему.



Если структура события отличается от структуры, ожидаемой выбранным типом конечной точки, то данное событие отбрасывается. Чтобы минимизировать потери в потоке событий, необходимо использовать нормализаторы для приведения событий к требуемой структуре.

> Для конечной точки типа **Внешняя система** ожидаемая структура определяется выбранной внешней системой.

При создании конечной точки типа **Внешняя система** требуется выбрать **Тип внешней системы** и задать параметры, зависящие от выбранного типа.

Доступны следующие типы внешних систем:

- Blackhole
- ClickHouse
- HTTP
- Kafka
- NATS JetStream
- Socket
- R-Vision TIP
- Vector

Blackhole

Конечная точка Blackhole предназначена для отбрасывания поступающих событий. Ее рекомендуется использовать для отладки конвейера, когда не нужно сохранять обработанные события в хранилище или отправлять их во внешнюю систему.

Параметры конечной точки Blackhole:

1. **Интервал запроса** (опционально): время в секундах между записями активности конечной точки в логи коллектора.
2. **Количество событий** (опционально): максимальное количество событий в секунду, которые могут обрабатываться конечной точкой.
3. **Использовать расширенные настройки**: настройка буферизации.

ClickHouse

Конечная точка ClickHouse предназначена для записи событий во внешнюю базу данных ClickHouse.

Параметры конечной точки ClickHouse:

1. **Адрес**: адрес внешней системы в формате URL- или IP-адреса с указанием порта. Например: `https://hostname:6000` или `192.0.2.0:6000`.
2. **База данных**: база данных, содержащая таблицу для записи данных. По умолчанию `default`.
3. **Таблица**: название таблицы для записи данных.
4. **Тип сжатия данных**: выбор типа сжатия данных:
   - **NONE** — без сжатия. Данный тип выбран по умолчанию.
   - **GZIP**.
   - **SNAPPY**.

- **ZLIB**.
- **ZSTD**.

5. **Тип аутентификации** (опционально): выбор типа аутентификации на HTTP-сервере:
   - **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   - **Токен**: выберите секрет типа **Токен**, хранящий токен для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

6. **Тип данных**: формат представления данных для записи в ClickHouse:
   - **JSONEACHROW** — каждая строка интерпретируется как набор полей таблицы. Данный тип выбран по умолчанию.
   - **JSONASOBJECT** — каждый объект JSON интерпретируется как одно значение типа `JSON`.
   - **JSONASSTRING** — каждый объект JSON интерпретируется как одна строка.
7. **Массовая отправка**: переключатель, позволяющий включить отправку данных пакетами. При включении доступны следующие настройки:
   - **Максимальный размер пакета**: максимальный размер несжатого и несериализованного пакета. Значение по умолчанию — `10 000 000`.
   - **Максимальное количество событий**: предельное количество событий в пакете, после которого происходит отправка.
   - **Время ожидания**: максимальное время в секундах, в течение которого формируется пакет. Значение по умолчанию — `1`.
8. **Кодировка**: переключатель, позволяющий настроить преобразования событий для сериализации. При включении доступны следующие настройки:
   - **Исключаемые поля**: поля, исключаемые из отправляемых событий.

     Чтобы добавить поле, нажмите на кнопку **Добавить**. Отобразится строка добавления поля.

     Чтобы удалить поле, нажмите на кнопку в строке поля.

   - **Включаемые поля**: поля, включаемые в отправляемые события.

     Чтобы добавить поле, нажмите на кнопку **Добавить**. Отобразится строка добавления поля.

     Чтобы удалить поле, нажмите на кнопку в строке поля.

   - **Формат временных меток**: формат для полей с временными метками:
     - **RFC3339** — формат RFC 3339;
     - **UNIX** — формат Unix-времени;
     - **UNIX_FLOAT** — формат Unix-времени с плавающей точкой;

- **UNIX_MS** — формат Unix-времени в миллисекундах;
- **UNIX_NS** — формат Unix-времени в наносекундах;
- **UNIX_US** — формат Unix-времени в микросекундах.

9. **Пропуск отсутствующих полей**: переключатель, позволяющий игнорировать поля, которые не входят в схему таблицы ClickHouse.
10. **Преобразовать даты**: переключатель, позволяющий использовать встроенные функции преобразования формата RFC 3339/ISO 8601 в ClickHouse.
11. **Включить TLS**: включение TLS-сертификата.
12. **Использовать расширенные настройки**: настройка буферизации.

HTTP

Конечная точка HTTP предназначена для подключения к внешнему серверу по HTTP-протоколу и отправки событий на этот сервер. Отправка событий происходит за счет выполнения HTTP-запросов, посылаемых от конечной точки к серверу.

Параметры конечной точки HTTP:

1. **Префикс** (опционально): строка, с которой начинается полезная нагрузка (payload) в теле HTTP-запроса.
2. **Суффикс** (опционально): строка, которой заканчивается полезная нагрузка (payload) в теле HTTP-запроса.
3. **Адрес**: адрес внешней системы в формате URL- или IP-адреса с указанием порта. Например: `https://hostname:6000` или `192.0.2.0:6000`.
4. **Тип аутентификации** (опционально): выбор типа аутентификации на HTTP-сервере:
   - **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   - **Токен**: выберите секрет типа **Токен**, хранящий токен для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

5. **Тип сжатия**: выбор типа сжатия данных:
   - **NONE** — без сжатия. Данный тип выбран по умолчанию.
   - **GZIP**.
   - **SNAPPY**.
   - **ZLIB**.
   - **ZSTD**.
6. **Формат событий**: формат передачи событий во внешнюю систему. Доступные форматы:
   - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

```
{ "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
}] }
```

- **CSV**. При выборе этого формата требуется ввести поля, которые требуется преобразовать в формат CSV. Например: `dvendor, dproduct, msg`.



Преобразование в формат CSV не поддерживается для полей с типом **Array** или **KeyValue**. Значения этих полей передаваться не будут.

- **GELF**.
- **JSON**. Данный формат выбран по умолчанию.
- **LOGFMT**.
- **NATIVE**.
- **NATIVE_JSON**.
- **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
- **RAW_MESSAGE**.
- **TEXT**.

7. **HTTP-заголовки** (опционально): заголовки HTTP-запроса. Заголовки позволяют уточнить HTTP-запрос и определяют, какие данные требуется передавать в запросе.

▼ *Об HTTP-заголовках*

HTTP-заголовок — это строка в HTTP-запросе, состоящая из пары "имя: значение". Он используется для уточнения запроса или описания содержимого сообщения. Формат заголовка определяется стандартом RFC 822.

HTTP-заголовки делятся на следующие категории:

- Основные заголовки (General Headers) — применяются во всех сообщениях клиента и сервера.
- Заголовки запроса (Request Headers) — используются только в запросах клиента.
- Заголовки ответа (Response Headers) — используются только в ответах от сервера.
- Заголовки сущности (Entity Headers) — применяются к содержимому сообщения.

Полный список HTTP-заголовков можно найти в MDN Web Docs.



Пример HTTP-заголовка

Для отправки данных на сервер порциями по 512 байт, в HTTP-запросе необходимо указать заголовок `Content-Length`. Значение этого заголовка должно соответствовать максимальному размеру поступающих данных в байтах, в данном случае это будет `512`.

Чтобы задать заголовок, нажмите на кнопку **Добавить**. Отобразится строка добавления

заголовка. Введите заголовок и его значение. Для одного заголовка можно задать несколько значений.

Чтобы удалить заголовок, нажмите на кнопку в строке нужного заголовка.

8. **Метод запроса** (опционально): выбор метода отправляемых HTTP-запросов:
   - **DELETE** — удаляет данные с сервера.
   - **GET** — запрашивает данные с сервера с полным HTTP-ответом.
   - **HEAD** — запрашивает данные с сервера без тела HTTP-ответа.
   - **PATCH** — используется для частичного изменения ресурса.
   - **POST** — отправляет данные на сервер.
   - **PUT** — заменяет данные на сервере данными из HTTP-запроса.
   - **OPTIONS** — запрашивает список методов, разрешенных на сервере.
   - **TRACE** — выполняет подключение к серверу и запрашивает от него ответ.
9. **Адрес HTTP-прокси** (опционально): полный адрес для подключения к HTTP-прокси в виде URI. Применяется, если HTTP-сервер использует прокси.
10. **Включить TLS**: включение TLS-сертификата.
11. **Использовать расширенные настройки**: настройка буферизации.

Kafka

Конечная точка Kafka предназначена для передачи событий с помощью распределенного брокера сообщений Apache Kafka.

Параметры конечной точки Kafka:

1. **Адрес**: IP-адреса серверов с портами для подключения к кластеру Kafka. Адреса вводятся через запятую. Например: `1.0.0.0:9092,1.0.0.1:9092`.
2. **Название топика**: название топика кластера Kafka, в который будут записываться передаваемые события.
3. **Формат событий**: формат передачи событий во внешнюю систему. Доступные форматы:
   - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

     ```
     { "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
     }] }
     ```

   - **CSV**. При выборе этого формата требуется ввести поля, которые требуется преобразовать в формат CSV. Например: `dvendor, dproduct, msg`.

     Преобразование в формат CSV не поддерживается для полей с типом **Array** или **KeyValue**. Значения этих полей передаваться не будут.

   - **GELF**.

- **JSON**. Данный формат выбран по умолчанию.
- **LOGFMT**.
- **NATIVE**.
- **NATIVE_JSON**.
- **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
- **RAW_MESSAGE**.
- **TEXT**.

4. **Включить TLS**: включение TLS-сертификата.
5. **Использовать расширенные настройки**: настройка буферизации.

NATS JetStream

Конечная точка NATS JetStream предназначена для передачи событий во внешнюю систему через брокер сообщений NATS.

Параметры конечной точки NATS JetStream:

1. **Адрес**: полный путь для подключения к NATS в формате URL- или IP-адреса с указанием порта. Например: `nats://cluster.nats.com` или `nats://127.0.0.1:4242`.

   Для обмена сообщениями через NATS по умолчанию используется порт 4222.

2. **Тема**: название темы NATS, в которую будут записываться события.

   Тема должна быть создана в NATS до настройки конечной точки.

3. **Формат событий**: формат передачи событий во внешнюю систему. Доступные форматы:
   - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

     ```
     { "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
     }] }
     ```

   - **CSV**. При выборе этого формата требуется ввести поля, которые требуется преобразовать в формат CSV. Например: `dvendor, dproduct, msg`.

     Преобразование в формат CSV не поддерживается для полей с типом **Array** или **KeyValue**. Значения этих полей передаваться не будут.

   - **GELF**.
   - **JSON**. Данный формат выбран по умолчанию.
   - **LOGFMT**.
   - **NATIVE**.

- **NATIVE_JSON**.
- **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
- **RAW_MESSAGE**.
- **TEXT**.

4. **Тип аутентификации** (опционально): выбор типа аутентификации на HTTP-сервере:
   - **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   - **Токен**: выберите секрет типа **Токен**, хранящий токен для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   - **NKey**: выберите секрет типа **NATS Key**, хранящий приватный и публичный ключи для подключения к NATS.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

5. **Включить TLS**: включение TLS-сертификата.
6. **Использовать расширенные настройки**: настройка буферизации.

Socket

Конечная точка Socket предназначена для передачи событий во внешние системы через сетевой сокет.

Параметры конечной точки Socket:

1. **Адрес**: адрес внешней системы в формате URL- или IP-адреса с указанием порта. Например: `https://hostname:6000` или `192.0.2.0:6000`.
2. **Протокол**: выбор протокола передачи событий. Доступные протоколы:
   - **TCP**.
   - **UDP**.
3. **Формат событий**: формат передачи событий во внешнюю систему. Доступные форматы:
   - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

     ```
     { "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
     }] }
     ```

   - **CSV**. При выборе этого формата требуется ввести поля, которые требуется преобразовать в

формат CSV. Например: `dvendor, dproduct, msg`.



Преобразование в формат CSV не поддерживается для полей с типом **Array** или **KeyValue**. Значения этих полей передаваться не будут.

- **GELF**.
- **JSON**. Данный формат выбран по умолчанию.
- **LOGFMT**.
- **NATIVE**.
- **NATIVE_JSON**.
- **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
- **RAW_MESSAGE**.
- **TEXT**.

4. **Включить TLS**: включение TLS-сертификата.
5. **Использовать расширенные настройки**: настройка буферизации.

R-Vision TIP

Конечная точка R-Vision TIP предназначена для передачи событий в систему R-Vision TIP с целью извлечения индикаторов компрометации из значений полей событий.



Данные о событиях передаются в R-Vision TIP в формате `TEXT`.

Параметры конечной точки R-Vision TIP:

1. **Адрес**: адрес внешней системы в формате URL- или IP-адреса с указанием порта. Например: `https://hostname:6000` или `192.0.2.0:6000`.
2. **Протокол**: выбор протокола передачи событий. Доступные протоколы:
   - **TCP**.
   - **UDP**.
3. **Использовать расширенные настройки**: настройка буферизации.

Vector

Конечная точка Vector предназначена для передачи событий в систему работы с логами Vector.

Параметры конечной точки Vector:

1. **Адрес**: адрес внешней системы в формате URL- или IP-адреса с указанием порта. Например: `https://hostname:6000` или `192.0.2.0:6000`.
2. **Включить TLS**: включение TLS-сертификата.
3. **Использовать расширенные настройки**: настройка буферизации.

Включение TLS-сертификата

Для внешних систем всех типов, кроме Blackhole и R-Vision TIP, доступен переключатель **Включить TLS**, позволяющий настроить шифрование TLS.

При включении TLS необходимо:

1. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

2. При необходимости выбрать секрет типа **Корневой сертификат**, хранящий данные корневого сертификата.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Расширенные настройки

Для всех внешних систем доступен переключатель **Использовать расширенные настройки**, позволяющий настроить буферизацию в конечной точке. Буфер используется для накопления событий, когда они не могут быть направлены конечной точкой во внешнюю систему (например, когда события направляются во внешнюю систему быстрее, чем она может их обработать). События находятся в буфере до тех пор, пока их не удастся успешно отправить.

При включении расширенных настроек необходимо:

1. Выбрать тип буфера:
   - **RAM** — для накопления событий используется оперативная память. Тип буфера **RAM** установлен по умолчанию.

     Для этого типа характерны высокая скорость буферизации, но низкая степень сохранности данных в случае сбоев системы. Тип буфера **RAM** рекомендуется использовать, когда требуется обеспечить максимальную производительность буферизации.

     При выборе этого типа требуется указать максимальное количество событий, которое может храниться в буфере.

   - **HDD** — для накопления событий используется дисковое пространство.

     Для этого типа характерны низкая скорость буферизации, но высокая степень сохранности данных за счет их хранения в дисковом пространстве. Тип буфера **HDD** рекомендуется использовать, когда требуется обеспечить сохранность передаваемых данных.

При выборе этого типа требуется указать размер буфера в байтах. Минимальный размер буфера — приблизительно 256 МБ (268435488 байт).

2. Выбрать поведение буфера при его заполнении:
   - **Ожидание** — поступление событий будет прекращено до освобождения места в буфере. Режим ожидания установлен по умолчанию.



Режим ожидания гарантирует сохранность данных, но может привести к замедлению работы буферизации.

   - **Отбрасывать новые события** — все события, поступающие в буфер после его заполнения, будут отброшены.



Режим отбрасывания новых событий обеспечивает высокую производительность буферизации, но может привести к потере данных.

Глобальная шина

Конечная точка типа **Глобальная шина** позволяет отправлять события в глобальную шину.



Данная конечная точка необходима для работы с коллекторами вне центрального пространства, но также может использоваться и с коллекторами в центральном пространстве.

Необходимость использования глобальной шины для отправки и получения событий из вынесенных коллекторов обусловлена тем, что запись событий в хранилище доступна только на конвейере коллектора, находящегося в центральном пространстве.

Пример обработки событий, собранных в вынесенном коллекторе, рассмотрен в разделе Сбор событий из вынесенного коллектора.

При создании конечной точки типа **Глобальная шина** не требуется задавать дополнительных параметров.

Получить события, сохраненные в глобальной шине, можно с помощью точки входа типа Global Bus.

Работа с шинами

Данный раздел описывает процесс работы с шинами коллектора в системе. Работа осуществляется на вкладке **Шины** в карточке коллектора.

О шинах

Шина — это компонент коллектора, который позволяет направлять потоки событий между конвейерами. В коллекторе может быть одна или несколько шин. Шина поддерживает два типа соединения: на получение и на отправку событий. Два конвейера событий могут сообщаться друг с другом посредством шин.

Список шин коллектора отображается на вкладке **Шины** в его карточке. Над списком расположено

поле для поиска шин по названию.

Доступные операции над шиной:

- Добавление шины
- Просмотр шины
- Изменение шины
- Удаление шины
- Работа с шиной на конвейере

Добавление шины

Чтобы добавить шину:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления шины.
5. Введите название шины.
6. При необходимости введите описание шины.
7. Нажмите на кнопку **Добавить**. Шина будет создана и отобразится в списке шин на вкладке **Шины** в карточке коллектора.

Просмотр шины

Чтобы отобразить информацию о шине:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. Нажмите на кнопку ⌄ в строке шины. Отобразится карточка шины.

В верхней части карточки отображается название шины. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранной шины.

Карточка шины содержит следующую информацию:

- **ID** — уникальный идентификационный код шины, генерируемый автоматически при ее создании.
- **Описание** — краткое описание функциональности шины.

- **Дата создания** — дата и время создания шины.
- **Создал** — пользователь, инициировавший создание шины.
- **Дата изменения** — дата и время последнего изменения шины.
- **Изменил** — пользователь, выполнивший последнее изменение шины.

Изменение шины

Чтобы изменить шину:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. Нажмите на кнопку действий ⋮ в строке шины и выберите опцию **Изменить**. Система отобразит окно редактирования шины.
5. Внесите изменения в требуемые поля.
6. Нажмите на кнопку **Сохранить**. Измененные данные шины будут сохранены.

Удаление шины

Чтобы удалить шину:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. Нажмите на кнопку действий ⋮ в строке шины и выберите опцию **Удалить**. Система отобразит окно подтверждения действия.
5. Нажмите **Удалить**. Шина будет удалена из коллектора.



Удаление недоступно для базовой шины **События с ошибкой** и для шин, используемых на конвейерах.

Работа с шиной на конвейере

Чтобы отправлять события на шину и/или получать из нее события, требуется добавить соединение с шиной на конфигурацию конвейера.

Существуют следующие типы соединения с шиной на конвейере:

- **Получение** — имеет один вход и не имеет выходов.
- **Отправка** — имеет один выход и не имеет входов.

На конфигурации конвейера шина принимает события на входе соединения типа **Получение** и

пересылает их на выход соединения типа **Отправка**. С помощью такой связи посредством использования шины можно настраивать обмен событиями между несколькими конвейерами в рамках одного коллектора.

Доступные операции над шиной на конвейере:

- Добавление шины на конвейер
- Изменение шины на конвейере
- Удаление шины с конвейера

Добавление шины на конвейер

Чтобы добавить соединение с шиной на конфигурацию конвейера:

1. Перейдите в конфигурацию требуемого конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Шина**. Отобразится окно добавления соединения с шиной.
4. Выберите из выпадающего списка шину, с которой требуется задать соединение. На выбор доступны только шины текущего коллектора.
5. Введите название соединения с шиной.
6. Выберите из выпадающего списка тип соединения с шиной: **Получение** или **Отправка**.
7. Нажмите на кнопку **Добавить**. Соединение с шиной отобразится на диаграмме конфигурации.

Изменение шины на конвейере

Способ изменения идентичен для всех элементов конвейера и приведен в разделе Изменение элемента конвейера.

Удаление шины с конвейера

Способ удаления идентичен для всех элементов конвейера и приведен в разделе Удаление элемента конвейера.

Работа с обогащениями

Данный раздел описывает процесс работы с обогащениями коллектора в системе. Работа осуществляется на вкладке **Обогащение** в карточке коллектора.

Об обогащении

Обогащение — это компонент коллектора, который позволяет использовать таблицы обогащения в коррелляторах и/или нормализаторах конвейеров. В коллекторе может быть одно или несколько обогащений. Они используются для обогащения и дополнения данных, передаваемых в полях событий: например, перевода кодов ошибок на понятный пользователю язык или расшифровки аббревиатур в наименовании оборудования.

Список обогащений коллектора отображается на вкладке **Обогащение** в его карточке. Над списком расположено поле для поиска обогащений по названию.

Доступные операции над обогащением:

- Добавление обогащения
- Просмотр обогащения
- Изменение обогащения
- Обновление обогащения
- Удаление обогащения

Добавление обогащения

Чтобы добавить обогащение:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления обогащения.
5. Введите название обогащения.
6. Выберите из выпадающего списка таблицу обогащения. Таблица будет использована для создания обогащения в коллекторе.
7. При необходимости введите описание обогащения.
8. Нажмите на кнопку **Добавить**. Обогащение будет создано и отобразится в списке обогащений на вкладке **Обогащение** в карточке коллектора.

Просмотр обогащения

Чтобы отобразить информацию об обогащении:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. Нажмите на кнопку ⌄ в строке обогащения. Отобразится карточка обогащения.

В верхней части карточки отображается название обогащения. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранного обогащения.

Карточка обогащения содержит следующую информацию:

- **ID обогащения** — уникальный идентификационный код компонента обогащения, генерируемый автоматически при добавлении его в коллектор.
- **Описание** — краткое описание функциональности обогащения.
- **ID таблицы обогащения** — уникальный идентификационный код таблицы обогащения, которая была использована при создании обогащения.
- **Таблица обогащения** — название таблицы обогащения, которая была использована при создании обогащения.
- **Версия таблицы обогащения** — версия таблицы обогащения, указанная в поле `version` в настройках таблицы.
- **Дата создания** — дата и время создания обогащения.
- **Создал** — пользователь, инициировавший создание обогащения.
- **Дата изменения** — дата и время последнего изменения обогащения.
- **Изменил** — пользователь, выполнивший последнее изменение обогащения.

Изменение обогащения

Чтобы изменить обогащение:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. Нажмите на кнопку действий ⋮ в строке обогащения и выберите опцию **Изменить**. Система отобразит окно редактирования обогащения.
5. Внесите изменения в требуемые поля.



Изменение выбора таблицы обогащения, которая была использована для создания обогащения, недоступно.

6. Нажмите на кнопку **Сохранить**. Измененные данные обогащения будут сохранены.

Обновление обогащения

При обновлении версии таблицы обогащения, становится доступным обновление обогащения, связанного с таблицей, для актуализации его данных.

Чтобы обновить обогащение:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку

этого коллектора с подробной информацией о нем.

3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. Нажмите на кнопку **Обновить** в строке нужного обогащения. Кнопка отображается только в том случае, если версия таблицы обогащения, связанной с обогащением, обновлена.

   Данные обогащения будут актуализированы в соответствии с новой версией таблицы обогащения.

Удаление обогащения

Чтобы удалить обогащение:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. Нажмите на кнопку действий ⋮ в строке обогащения и выберите опцию **Удалить**. Система отобразит окно подтверждения действия.
5. Нажмите **Удалить**. Обогащение будет удалено из коллектора.

> Удаление доступно только для обогащений, таблицы обогащения которых не задействованы в правилах корреляции или нормализации, используемых в коррелляторах или нормализаторах на конвейерах текущего коллектора.

Работа с моделями событий

Данный раздел описывает процесс работы с моделями событий в системе. Работа осуществляется в разделе **Ресурсы** → **Модели событий** веб-интерфейса системы.

> Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О моделях событий

Модель события определяет формат хранилища событий — имена, типы полей и типы данных в полях. При создании хранилища событий обязательно указывается используемая в нем модель события.

В системе доступны модели событий следующих типов:

- **Системные модели событий** — создаются системой автоматически. Их нельзя изменять или удалять.
- **Пользовательские модели событий** — создаются вручную или в результате импорта.

Существуют следующие системные модели событий:

- **Универсальная модель события** — предназначена для хранения информации о событиях в унифицированном виде.
- **Универсальная модель события 2.0** — представляет собой развитие предыдущей универсальной модели со следующими целями:
  - Повысить консистентность заполнения полей, чтобы упростить интерпретацию событий.
  - Упростить разработку правил нормализации.
  - Сделать работу с карточкой события более удобной для аналитика.



Поддержка предыдущей универсальной модели будет прекращена в ближайших версиях системы. При создании новых моделей и хранилищ событий рекомендуется использовать универсальную модель 2.0.

- **Модель события аудита** — предназначена для хранения информации о событиях сервиса аудита.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую модель события.
- Поле **Поиск** предназначено для быстрого поиска моделей событий в отображаемом списке по названию. Выпадающий список **Статус** позволяет настроить отображение списка моделей событий в соответствии с выбранным статусом.
- Кнопки **Экспорт** (↑) и **Импорт** (↓) позволяют выгружать существующие модели событий из системы и загружать в систему модели событий извне.

В рабочей области раздела отображается таблица имеющихся моделей событий. Таблица содержит следующие столбцы:

- **Название** — название модели события, используемое для ее идентификации в системе.
- **Описание** — описание модели события.
- **Тип** — тип модели события: системная или пользовательская.
- **Статус** — текущий статус модели события: **Включена** или **Выключена**.
- **Дата создания** — дата и время создания модели события.
- **Дата изменения** — дата и время последнего изменения модели события.

При работе с таблицей моделей событий доступны следующие операции:

- Поиск моделей по полю **Название**.
- Фильтрация моделей по полю **Статус**. Статус выбирается из выпадающего списка.
- Сортировка моделей по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

Работа с моделью события

Доступные операции над моделями событий:

- Создание модели события
- Просмотр информации о модели события
- Включение модели события
- Изменение модели события
- Импорт модели события
- Экспорт модели события
- Создание новой модели события на базе текущей
- Удаление модели события

Создание модели события

Чтобы создать модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания модели события.
3. Введите название модели события.
4. Введите описание модели события.
5. В нижней части окна находится раздел **Схема**, содержащий список полей, которые будут использоваться в этой модели события.

   По умолчанию система отображает набор **служебных** полей. Список можно расширить, добавив вручную **регулярные** поля.

   Для обеспечения оптимальной производительности системы рекомендуется создавать модели с количеством полей не более 1000.

6. Нажмите на кнопку **Создать**. Система создаст модель событий и отобразит соответствующее уведомление. Новая модель появится в списке раздела **Ресурсы** → **Модели событий**.

Добавление регулярных полей

Чтобы добавить регулярное поле:

1. Нажмите на кнопку **Добавить поле**. Система отобразит окно добавления поля.
2. Введите уникальный ключ идентификации нового поля.

   Ключ может состоять только из латинских букв, цифр и подчеркиваний. При этом он должен начинаться со строчной буквы.

3. При необходимости установите флажок **Поле с изменяемой семантикой**, чтобы использовать значения из другого поля при построении семантики названия нового поля.

Выберите поле из выпадающего списка **Поле, хранящее название**, чтобы определить, какое поле будет использовать система для формирования его названия. Можно использовать только следующие типы полей: **String**, **LCString**, **Enum**.



Пример

Создадим поле с ключом `collector_number` и укажем, что для задания названия этого поля нужно использовать поле `id_list`. Предположим, поле `id_list` принимает значение `collector_id`, а поле `collector_number` — `250`.

При применении этой модели события для поля с ключом `collector_number` в событии отобразится значение: `collector_id:250`.

4. Введите описание нового поля.
5. Выберите тип данных нового поля.



Чтобы работать со вложенными полями `JSON`, укажите тип данных **JSON**.

6. Укажите параметры в зависимости от выбранного типа данных:

Array

Выберите тип элемента массива из списка.

Enum

Укажите список предопределенных значений.



Enum-список может содержать уникальные целочисленные значения от 0 до 127.

Чтобы добавить значение в Enum-список:

1. В секции **Элементы** нажмите на кнопку **Добавить**. Отобразится строка настроек нового значения.
2. Введите ключ нового значения. Особенности задания ключа:
   - Ключ может содержать числа, специальные символы, латинские и кириллические буквы.
   - Ключ может содержать не более 128 символов.
   - Ключ должен быть уникален в рамках текущего Enum-списка.
3. Введите новое значение. Особенности задания значения:
   - Значением может быть любое целое число от 0 до 127.
   - Значение должно быть уникальным в рамках текущего Enum-списка.

Чтобы удалить значение из Enum-списка, нажмите на кнопку в правой части строки этого значения.

KeyValue

Выберите тип значения KeyValue-пары из списка.

UInt

Выберите разрядность из списка. Доступные варианты: 8, 16, 32, 64, 128, 256.

7. При необходимости установите переключатель **Может быть null** в активное положение, чтобы поле могло хранить значение типа `null`. Если переключатель выключен, а поле не заполнено, событие в хранилище не будет создано.



В полях типов **Array**, **KeyValue**, **LCString** этот переключатель не используется.

8. При необходимости установите переключатель **Отображать поле** в активное положение, чтобы поле отображалось в списке результатов поиска в разделе **Поиск**.



Эту настройку можно изменить при просмотре модели события или при настройке внешнего вида списка событий.

9. Нажмите на кнопку **Добавить**.

Чтобы изменить добавленное поле:

1. Нажмите на кнопку в его строке. Отобразится окно изменения поля.
2. Внесите требуемые изменения в настройки поля.
3. Нажмите на кнопку **Сохранить**. Измененные данные поля будут сохранены.

Чтобы удалить добавленное поле, нажмите на кнопку в его строке. Поле будет удалено из списка полей модели события.

Просмотр информации о модели события

Чтобы просмотреть информацию о модели события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области карточку модели со списком ее полей.

Название модели события отображается в верхней части карточки.

Чтобы закрыть список полей модели и вернуться в раздел **Ресурсы** → **Модели событий**, нажмите на кнопку **Назад** в верхней части карточки или перейдите в раздел **Ресурсы** → **Модели событий** на боковой панели.

На панели инструментов карточки расположены следующие компоненты:

- Кнопка позволяет изменить текущую модель.
- Кнопка позволяет удалить текущую модель.

- Кнопка позволяет экспортировать текущую модель.
- Кнопка позволяет создать новую модель на базе текущей.
- Переключатель статуса позволяет включить или выключить текущую модель.
- Идентификатор и описание модели предоставляют краткую информацию о ее предназначении.
- Поле **Поиск** предназначено для быстрого поиска полей в списке по ключу.

Список полей модели представлен в виде таблицы со следующими столбцами:

- **Ключ поля** — уникальный ключ идентификации поля.
- **Описание** — краткое описание функциональности поля.
- **Тип поля** — тип поля:
  - **Служебное** — поле, создаваемое системой автоматически.
  - **Регулярное** — поле, добавленное в модель события вручную.
- **Тип данных** — тип данных значений, хранящихся в поле модели события.
- **Параметры типа данных** — дополнительные параметры значений поля, зависящие от выбранного типа данных:
  - для **UInt** — разрядность;
  - для **KeyValue** — тип данных значения в KeyValue-паре;
  - для **Enum** — список ключей и их значений;
  - для **Array** — тип данных элементов массива.
- **Поле хранящее название** — ключ поля, определяющего семантику названия текущего поля.
- **Может быть null** — возможность хранить значение типа `null` в поле модели события.
- **Отображать поле** — отображение поля при просмотре списка событий модели. Чтобы поле отображалось, установите переключатель в активное положение. По умолчанию отображаются следующие поля:
  - для универсальной модели события: `timestamp`, `dvendor`, `dproduct`, `name`, `msg`, `dvc`, `outcome`;
  - для универсальной модели 2.0: `timestamp`, `device_product`, `device_hostname`, `description`, `focus`;
  - для модели события аудита: `timestamp`, `raw`, `suser`, `eventType`, `moduleKey`, `eventRiskLevel`, `data`, `meta`;
  - для пользовательской модели события: `id`, `type`, `timestamp`, `sourceIp`.

При работе с таблицей доступен поиск полей по их ключу.

Включение модели события



Включение и выключение доступно только для моделей событий с типом **Пользовательская**.

Для того чтобы модель события можно было использовать при создании хранилища событий, ее нужно включить.

При создании модель события включается автоматически.

Чтобы включить модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Установите переключатель состояния модели события, расположенный над списком полей модели, в активное положение. Система включит модель и отобразит соответствующее уведомление. Модель события появится в списке доступных моделей при создании хранилища событий.

Чтобы выключить работающую модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Установите переключатель состояния модели события, расположенный над списком полей модели, в неактивное положение. Отобразится окно подтверждения выключения модели события.
4. Нажмите на кнопку **Подтвердить**. Система выключит модель и отобразит соответствующее уведомление. Модель события исчезнет из списка доступных моделей при создании хранилища событий.

Изменение модели события

Вы можете изменить модель события через панель инструментов ее карточки.

- Изменение доступно только для моделей событий с типом **Пользовательская**.
- Измененная модель события автоматически обновляется во всех хранилищах, в которых она используется.

Импорт модели события

Для загрузки модели события в систему применяется функционал импорта, поддерживающий обработку файлов в формате JSON.

Чтобы импортировать модель событий:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на кнопку ⤓ на панели инструментов. Система отобразит окно импорта модели события.
3. Нажмите **Выбрать файл** (⊕) и выберите файл модели в формате JSON.
4. Нажмите на кнопку **Импортировать**. Начнется валидация и импорт моделей. Система отобразит

окно с итогами импорта.



Модель будет успешно импортирована, если соответствует следующим критериям:

- Идентификатор модели не совпадает с идентификаторами моделей, уже существующих в системе.
- Ключи полей уникальны и содержат только латинские буквы, цифры и подчеркивания, при этом не начинаются с цифры.
- В полях указаны корректные значения.
- Поля типов **LCString**, **Array** и **KeyValue** не содержат пустых значений.
- Поля с измененной семантикой ссылаются на существующие поля.

Экспорт модели события

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Нажмите на кнопку ↑, расположенную над списком полей модели. Система загрузит данные модели на устройство пользователя в формате JSON и отобразит соответствующее уведомление.

Создание новой модели события на базе текущей

Чтобы создать новую модель события на базе текущей:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Нажмите на кнопку +, расположенную над списком полей модели. Отобразится окно создания новой модели события, в котором автоматически будут добавлены все поля текущей модели.
4. Введите название модели события.
5. Введите описание модели события.
6. При необходимости измените состав полей новой модели события.
7. Нажмите на кнопку **Создать**. Система создаст модель события и отобразит соответствующее уведомление. Новая модель появится в списке раздела **Ресурсы** → **Модели событий**.

Удаление модели события

Вы можете удалить модель события через панель инструментов ее карточки.



Удаление доступно только для моделей событий с типом **Пользовательская**, не используемых в других сущностях системы.

Универсальная модель события

Универсальная модель события предназначена для хранения информации о событиях в едином, универсальном, виде. Универсальная модель позволяет приводить поступающие в систему базовые события к унифицированному формату с единым набором полей.

Разложение данных базовых событий по полям универсальной модели происходит в ходе процесса нормализации, выполняющегося при работе конвейера обработки событий.

Универсальная модель события является встроенной моделью и недоступна для изменения или удаления. Она отображается в разделе **Ресурсы → Модели событий**.

При необходимости вы можете создать собственную (пользовательскую) модель события с требуемым набором полей.



Никакое поле универсальной модели события не может хранить в себе дополнительное значение типа `null`. Чтобы хранить такое значение, требуется создать собственную модель события и добавить в нее поле с установленным переключателем **Может быть null**.

Поля универсальной модели разделяются на следующие группы:

- Служебные поля
- Регулярные поля
- Регулярные настраиваемые поля
- Поля активов

Служебные поля

Служебные поля — обязательные поля модели события, создаваемые системой автоматически. Служебные поля образуют минимальный набор полей, необходимый для получения, хранения, анализа, обработки и отправки событий.

Служебные поля по умолчанию содержатся во всех моделях событий и не могут быть изменены или удалены.

Служебные поля универсальной модели события

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`id`	`UUID`	Да	Уникальный идентификатор события. Генерируется коллектором после нормализации.
`sourceIp`	`IPv6`	Нет	IP-адрес источника, от которого получено базовое событие.
`tenantId`	`LCString`	Да	Идентификатор тенанта.

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
art	DateTime64	Нет	Дата и время получения события системой-источником или агентом.
timestamp	DateTime64	Да	Дата и время получения события коллектором.
raw	String	Нет	Текст базового (сырого) события.
collectorId	LCString	Да	Идентификатор коллектора, получившего событие.
aggregationRuleName	LCString	Нет	Название правила агрегации, обработавшего событие.
aggregationRuleId	LCString	Нет	Идентификатор правила агрегации, обработавшего событие.
cnt	UInt32	Нет	Количество базовых событий, на основе которых создано агрегированное событие.
type	Enum	Нет	Тип события. Возможные значения: • -1 — не указано; • 0 — базовое событие; • 1 — нормализованное событие; • 2 — агрегированное событие; • 3 — корреляционное событие; • 4 — событие аудита.
correlationRuleName	LCString	Нет	Имя правила корреляции, на основе которого создано корреляционное событие.
correlationRuleId	LCString	Нет	Идентификатор правила корреляции, на основе которого создано корреляционное событие.
baseEventIds	Массив UUID	Нет	Список идентификаторов базовых событий, на основе которых создано агрегированное или корреляционное событие.
baseEventTimestamps	Массив DateTime64	Нет	Список дат и времен получения базовых событий в коллекторе, на основе которых создано агрегированное или корреляционное событие. Используется для идентификации событий, связанных с корреляционным или агрегированным событием.

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`groupedBy`	Массив `LCString`	Нет	Список полей модели события, по которому группируются события.
`correlationSeverity`	`Enum`	Нет	Уровень угрозы корреляционного события. Возможные значения: • -1 — не указано; • 1 — низкий; • 2 — средний; • 3 — высокий; • 4 — критический.

Регулярные поля

Регулярные поля — дополнительные поля модели события, не являющиеся обязательными. В пользовательских моделях событий регулярными полями являются все поля, добавленные пользователем при создании или редактировании модели.

В отличие от служебных полей, набор регулярных полей не является универсальным и может изменяться от модели к модели.

Регулярные поля универсальной модели события

Ключ поля	Тип данных	Описание
`aid`	`LCString`	Идентификатор агента, от которого получено событие.
`at`	`LCString`	Тип агента, от которого получено событие. Например: "Endpoint".
`av`	`LCString`	Версия агента, от которого получено событие.
`dvendor`	`LCString`	Название производителя источника журнала. Значение берется из базового события.
`dproduct`	`LCString`	Название продукта из источника журнала. Значение берется из базового события.
`dversion`	`LCString`	Версия продукта из источника журнала Значение берется из базового события.

Ключ поля	Тип данных	Описание
`deviceEventClassId`	`LCString`	Идентификатор типа события из источника журнала. Некоторые источники журнала определяют категорию событий.
`name`	`String`	Название события. Значение берется из базового события.
`severity`	`LCString`	Уровень важности события. Значение берется из базового события.
`act`	`LCString`	Действие, которое предпринято устройством. Например: `blocked`, `detected`.
`app`	`LCString`	Протокол уровня приложений. Например: HTTP, HTTPS, Telnet.
`cat`	`LCString`	Категория базового события. Устройства часто используют собственную схему категоризации для классификации событий.
`destinationDnsDomain`	`LCString`	DNS-часть полного доменного имени (FQDN) точки назначения, если базовое событие содержит сведения об отправителе и получателе данных.
`destinationServiceName`	`LCString`	Название сервиса на принимающей стороне. Например: "sshd". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`destinationTranslatedAddress`	`IPv6`	IP-адрес устройства, принимающего трафик (после трансляции). Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`destinationTranslatedPort`	`UInt16`	Номер порта на устройстве приемника трафика (после трансляции адреса приемника). Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
deviceDirection	Enum	Направление соединения из базового события. Значения: • -1 — не указано; • 0 — входящее соединение; • 1 — исходящее соединение.
deviceDnsDomain	LCString	DNS-часть полного доменного имени (FQDN) IP-адреса устройства, с которого пришло базовое событие.
deviceExternalId	LCString	Внешний идентификатор устройства. Значение при его наличии берется из базового события.
deviceFacility	LCString	Значение параметра Facility. Значение при его наличии берется из базового события. Например: в Syslog в параметре Facility может передаваться название компоненты ОС, в которой произошла ошибка.
deviceInboundInterface	LCString	Название интерфейса входящего соединения.
deviceNtDomain	LCString	Доменное имя Windows-устройства.
deviceOutboundInterface	LCString	Название интерфейса исходящего соединения.
devicePayloadId	String	Идентификатор полезной нагрузки, который ассоциирован с базовым событием.
deviceProcessName	LCString	Название процесса. Значение берется из базового события.
deviceTranslatedAddress	IPv6	Ретранслированный IP-адрес устройства, с которого пришло базовое событие.
dhost	LCString	Название хоста, принимающего трафик. Полное доменное имя (FQDN) приемника трафика, если доступно. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dmac	MAC	MAC-адрес устройства, принимающего трафик. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
dntdom	LCString	Доменное имя Windows-устройства, принимающего трафик. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dpid	UInt64	Идентификатор системного процесса, ассоциированного с приемником трафика в базовом событии. Например: если в событии указано `Process ID 105`, то `destinationProcessId=105`. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dpriv	String	Названия security-ролей, которые определяют пользовательские привилегии на стороне точки назначения. Например: "User", "Guest", "Administrator". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dproc	LCString	Название системного процесса в точке назначения. Например: "sshd", "telnet". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dpt	Uint16	Номер порта на стороне точки назначения. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dst	IPv6	IPv6-адрес точки назначения. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dtz	LCString	Часовой пояс устройства, на котором сгенерировано событие.
duid	LCString	Идентификатор пользователя на стороне точки назначения. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
duser	LCString	Имя пользователя на стороне точки назначения. Может содержать адрес электронной почты пользователя. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dvc	IPv6	IPv6-адрес устройства, с которого получено событие.
dvchost	LCString	Название хоста устройства, с которого получено событие. Полное доменное имя (FQDN) устройства, если доступно.
dvcpid	UInt64	Идентификатор системного процесса устройства, сгенерировавшего событие.
endTime	DateTime64	Дата и время завершения действия. Значение берется из базового события.
externalId	LCString	Идентификатор устройства, сгенерировавшего событие.
fileCreateTime	DateTime64	Дата и время создания файла. Значение берется из базового события.
fileHash	String	Хэш-код файла.
fileId	LCString	Идентификатор файла.
fileModificationTime	DateTime64	Дата и время последнего изменения файла.
filePath	String	Путь к файлу (с указанием имени файла).
filePermission	String	Список разрешений к файлу.
fileType	LCString	Тип файла. Например: application, pipe, socket.
fname	String	Имя файла (без указания пути к файлу).
fsize	UInt64	Размер файла.
bytesIn	UInt64	Количество байтов, полученных из источника и переданных получателю. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
msg	String	Краткое описание ошибки (проблемы) из события.
oldFileCreateTime	DateTime64	Дата и время создания OLD-файла из события.
oldFileHash	String	Хэш-код OLD-файла.
oldFileId	LCString	Идентификатор OLD-файла (при наличии).
oldFileModificationTime	DateTime64	Дата и время последнего изменения OLD-файла.
oldFileName	String	Имя OLD-файла (без указания пути к файлу).

Ключ поля	Тип данных	Описание
oldFilePath	String	Путь к OLD-файлу (с указанием имени файла).
oldFilePermission	String	Список разрешений к OLD-файлу.
oldFileSize	UInt64	Размер OLD-файла.
oldFileType	LCString	Тип OLD-файла. Например: application, pipe, socket.
bytesOut	UInt64	Количество отправленных байтов. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
outcome	LCString	Результат выполнения действия act на устройстве.
proto	LCString	Название протокола 4 уровня OSI. Например: TCP, UDP.
reason	String	Причина появления события. Например: "bad password" или "unknown user". Также может содержать ошибку или код. Например: “0x1234”
request	String	Запрошенный URL-адрес.
requestClientApplication	String	Агент, который обработал запрос request.
requestContext	String	Описание контекста запроса.
requestCookies	String	Файлы cookie, связанные с запросом.
requestMethod	LCString	Метод, который использовался для доступа к веб-адресу. Например: POST, GET.
rt	DateTime64	Дата и время получения устройством сведений о наблюдаемом событии.
shost	LCString	Название хоста источника трафика. Полное доменное имя (FQDN) источника трафика, если доступно. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
smac	MAC	MAC-адрес устройства источника трафика. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
`sntdom`	`LCString`	Доменное имя Windows-устройства источника трафика. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`sourceDnsDomain`	`LCString`	DNS-часть полного доменного имени (FQDN) Windows-устройства источника трафика. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`sourceServiceName`	`LCString`	Название сервиса на стороне источника трафика. Например: "sshd". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`sourceTranslatedAddress`	`IPv6`	IPv6-адрес перехода источника. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`sourceTranslatedPort`	`UInt16`	Номер порта перехода на стороне источника. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`spid`	`UInt64`	Идентификатор системного процесса, ассоциированного с источником трафика в базовом событии. Например: если в событии указано `Process ID 105`, то `SourceProcessId=105`. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`spriv`	`String`	Названия security-ролей, которые определяют пользовательские привилегии на стороне источника. Например: "User", "Guest", "Administrator". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`sproc`	`String`	Название системного процесса на стороне источника. Например: "sshd", "telnet". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
spt	UInt16	Номер порта на стороне источника. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
src	IPv6	IPv6-адрес источника. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
startTime	DateTime64	Дата и время начала действия.
suid	LCString	Идентификатор пользователя на стороне источника.
suser	LCString	Имя пользователя на стороне источника. Может содержать адрес электронной почты пользователя.
logonType	String	Тип входа.
sourceSessionId	String	Идентификатор сессии источника события.
targetSessionId	String	Идентификатор целевой сессии события.
cmd	String	Команда в командной строке.
logonProcessName	LCString	Имя процесса авторизации.
authPackageName	LCString	Имя пакета авторизации.
objType	LCString	Тип объекта действия.
objName	String	Имя объекта действия.
objPath	String	Путь к объекту действия.
accessMask	LCString	Запрошенная маска доступа.
accessList	LCString	Список запрошенных прав доступа.
accesses	LCString	Запрошенные права доступа в читаемом формате.
dGroup	LCString	Имя целевой группы SID (объекта) действия.
dGroupDom	LCString	Имя домена целевой группы SID (объекта) действия.
status	LCString	Статус события или действия из источника.

Регулярные настраиваемые поля

Регулярные настраиваемые поля — это особый тип регулярных полей в универсальной модели события, который не привязан к какому-то конкретному способу применения. Эти поля позволяют хранить в модели события данные того или иного типа, не предусмотренные заранее прочими полями модели.



Чтобы значение поля, хранящего название, могло корректно использоваться, оно должно состоять только из латинских букв, цифр и подчеркиваний и начинаться со строчной буквы.



Пример

В событии передается адрес электронной почты пользователя, совершившего подозрительное действие. По умолчанию в универсальной модели события для хранения электронной почты нет специальных полей.

В таком случае можно использовать регулярные настраиваемые поля. Данные об адресе электронной почты можно записать в поле `cs1`, а в поле `cs1Label` указать предназначение поля `cs1`, например, "email".

Регулярные настраиваемые поля универсальной модели события

Ключ поля	Тип данных	Описание
`cn1`	`UInt64`	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn1Label`	`LCString`	Поле, хранящее название для `cn1`.
`cn2`	`UInt64`	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn2Label`	`LCString`	Поле, хранящее название для `cn2`.
`cn3`	`UInt64`	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn3Label`	`LCString`	Поле, хранящее название для `cn3`.
`cn4`	`UInt64`	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn4Label`	`LCString`	Поле, хранящее название для `cn4`.
`cn5`	`UInt64`	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn5Label`	`LCString`	Поле, хранящее название для `cn5`.
`cs1`	`String`	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
`cs1Label`	`LCString`	Поле, хранящее название для `cs1`.
`cs2`	`String`	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
`cs2Label`	`LCString`	Поле, хранящее название для `cs2`.
`cs3`	`String`	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
`cs3Label`	`LCString`	Поле, хранящее название для `cs3`.

Ключ поля	Тип данных	Описание
cs4	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs4Label	LCString	Поле, хранящее название для cs4.
cs5	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs5Label	LCString	Поле, хранящее название для cs5.
cs6	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs6Label	LCString	Поле, хранящее название для cs6.
cs7	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs7Label	LCString	Поле, хранящее название для cs7.
cs8	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs8Label	LCString	Поле, хранящее название для cs8.
deviceCustomDate1	DateTime64	Поле для маппинга значения даты и времени, которое не может быть сопоставлено любому другому полю модели события.
deviceCustomDate1Label	LCString	Поле, хранящее название для deviceCustomDate1.
deviceCustomDate2	DateTime64	Поле для маппинга значения даты и времени, которое не может быть сопоставлено любому другому полю модели события.
deviceCustomDate2Label	LCString	Поле, хранящее название для deviceCustomDate2.
deviceCustomDate3	DateTime64	Поле для маппинга значения даты и времени, которое не может быть сопоставлено любому другому полю модели события.
deviceCustomDate3Label	LCString	Поле, хранящее название для deviceCustomDate3.
c6a1	IPv6	Поле для маппинга значения IPv6 (или IPv4 в нотации IPv6), которое не может быть сопоставлено любому другому полю модели события.
c6a1Label	LCString	Поле, хранящее название для c6a1.
c6a2	IPv6	Поле для маппинга значения IPv6 (или IPv4 в нотации IPv6), которое не может быть сопоставлено любому другому полю модели события.
c6a2Label	LCString	Поле, хранящее название для c6a2.

Ключ поля	Тип данных	Описание
c6a3	IPv6	Поле для маппинга значения IPv6 (или IPv4 в нотации IPv6), которое не может быть сопоставлено любому другому полю модели события.
c6a3Label	LCString	Поле, хранящее название для c6a3.
c6a4	IPv6	Поле для маппинга значения IPv6 (или IPv4 в нотации IPv6), которое не может быть сопоставлено любому другому полю модели события.
c6a4Label	LCString	Поле, хранящее название для c6a4.

Поля активов

Поля активов — это особый тип регулярных полей, позволяющий хранить сведения об активах организации. Активы представляют собой ресурсы организации, используемые в ее IT-инфраструктуре: например, оборудование, персонал.

Поля активов универсальной модели события

Ключ поля	Тип данных	Описание
sourcePersonID	LCString	Уникальный идентификатор пользователя на стороне источника.
sourcePersonFullName	LCString	ФИО пользователя на стороне источника.
sourcePersonPosition	LCString	Должность пользователя на стороне источника.
sourcePersonUserName	LCString	Учетная запись пользователя на стороне источника. Может содержать несколько значений.
targetPersonID	LCString	Уникальный идентификатор пользователя на стороне точки назначения.
targetPersonFullName	LCString	ФИО пользователя на стороне точки назначения.
targetPersonPosition	LCString	Должность пользователя на стороне точки назначения.
targetPersonUserName	LCString	Учетная запись пользователя на стороне точки назначения. Может содержать несколько значений.
sourceDeviceID	LCString	Уникальный идентификатор оборудования на стороне источника.
sourceDeviceName	LCString	Название оборудования на стороне источника.
sourceDeviceIPAddress	IPv6	IP-адрес оборудования на стороне источника.
sourceDeviceOS	LCString	Операционная система оборудования на стороне источника.
sourceDeviceOwner	LCString	Владелец оборудования на стороне источника.
targetDeviceID	LCString	Уникальный идентификатор оборудования на стороне точки назначения.
targetDeviceName	LCString	Название оборудования на стороне точки назначения.
targetDeviceIPAddress	IPv6	IP-адрес оборудования на стороне точки назначения.

Ключ поля	Тип данных	Описание
`targetDeviceOS`	`LCString`	Операционная система оборудования на стороне точки назначения.
`targetDeviceOwner`	`LCString`	Владелец оборудования на стороне точки назначения.

Универсальная модель события 2.0

Универсальная модель события 2.0 предназначена для хранения информации о событиях в едином универсальном виде. Универсальная модель позволяет приводить поступающие в систему базовые события к унифицированному формату с единым набором полей.



Универсальная модель 2.0 представляет собой развитие предыдущей универсальной модели со следующими целями:

- Повысить консистентность заполнения полей, чтобы упростить интерпретацию событий.
- Упростить разработку правил нормализации.
- Сделать работу с карточкой события более удобной для аналитика.

Поддержка предыдущей универсальной модели будет прекращена в ближайших версиях системы. При создании новых моделей и хранилищ событий рекомендуется использовать универсальную модель 2.0.

Разложение данных базовых событий по полям универсальной модели происходит в ходе процесса нормализации, выполняющегося при работе конвейера обработки событий.

Универсальная модель 2.0 является встроенной моделью и недоступна для изменения или удаления. Она отображается в разделе **Ресурсы** → **Модели событий**.

При необходимости вы можете создать собственную (пользовательскую) модель события с требуемым набором полей.



Никакое поле универсальной модели события не может хранить в себе дополнительное значение типа `null`. Чтобы хранить такое значение, требуется создать собственную модель события и добавить в нее поле с установленным переключателем **Может быть null**.

Поля универсальной модели разделяются на следующие группы:

- Служебные поля
- Регулярные поля

Служебные поля

Служебные поля — обязательные поля модели события, создаваемые системой автоматически. Служебные поля образуют минимальный набор полей, необходимый для получения, хранения, анализа, обработки и отправки событий.

Служебные поля по умолчанию содержатся во всех моделях событий и не могут быть изменены или удалены.

Признак **Заполняется системой по умолчанию** означает, что можно всегда ожидать, что поле будет заполнено системой.

Служебные поля универсальной модели события

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`id`	`UUID`	Да	Уникальный идентификатор события. Генерируется коллектором после нормализации.
`sourceIp`	`IPv6`	Нет	IP-адрес источника, от которого получено базовое событие.
`tenantId`	`UUID`	Да	Идентификатор тенанта.
`art`	`DateTime64`	Нет	Дата и время формирования события на агенте в формате ISO 8601 (UTC).
`timestamp`	`DateTime64`	Да	Дата и время получения события коллектором (UTC).
`raw`	`String`	Нет	Текст базового (сырого) события.
`collectorId`	`LCString`	Да	Идентификатор коллектора, обработавшего событие.
`aggregationRuleName`	`LCString`	Нет	Название правила агрегации, обработавшего событие.
`aggregationRuleId`	`LCString`	Нет	Идентификатор правила агрегации, обработавшего событие.
`cnt`	`UInt32`	Нет	Количество исходных событий, агрегированных в данном событии.
`type`	`Enum`	Нет	Тип события. Возможные значения: • -1 — не указано; • 0 — базовое событие; • 1 — нормализованное событие; • 2 — агрегированное событие; • 3 — корреляционное событие; • 4 — событие аудита.
`correlationRuleName`	`LCString`	Нет	Название правила корреляции, на основе которого создано корреляционное событие.

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`correlationRuleId`	`LCString`	Нет	Идентификатор правила корреляции, на основе которого создано корреляционное событие.
`baseEventIds`	Массив `LCString`	Нет	Список идентификаторов нормализованных событий, на основе которых создано агрегированное или корреляционное событие.
`baseEventTimestamps`	Массив `DateTime64`	Нет	Список дат и времен получения базовых событий в коллекторе, на основе которых создано агрегированное или корреляционное событие.
`groupedBy`	Массив `LCString`	Нет	Список полей модели события, по которому группируются события.
`correlationSeverity`	`Enum`	Нет	Уровень критичности корреляционного события. Определяет степень опасности инцидента и приоритет реагирования. Возможные значения: • -1 — не указано; • 1 — низкий; • 2 — средний; • 3 — высокий; • 4 — критический.

Регулярные поля

Регулярные поля — дополнительные поля модели события, не являющиеся обязательными. В пользовательских моделях событий регулярными полями являются все поля, добавленные пользователем при создании или редактировании модели.

В отличие от служебных полей, набор регулярных полей не является универсальным и может изменяться от модели к модели.

Регулярные поля универсальной модели события

Ключ поля	Тип данных	Описание
`device_time`	`DateTime64`	Метка времени события по системным часам устройства в формате ISO 8601 (UTC).
`device_timezone`	`LCString`	Часовой пояс устройства в момент генерации события.

Ключ поля	Тип данных	Описание
`description`	`String`	Описание события, содержащее ключевые контекстные данные, такие как: • инициатор действия; • целевой объект; • результат операции; • дополнительные параметры.
`category`	`LCString`	Основная категория события, определяющая его принадлежность к высокоуровневой группе.
`category_sub`	`LCString`	Подкатегория события, детализирующая его в рамках основной категории. Позволяет проводить более точный анализ и поиск событий.
`action`	`Enum`	Тип операции, которая была зафиксирована в событии. Используется для классификации событий и определения их семантики в контексте безопасности. Возможные значения перечислены в разделе Типы операций в событии.
`outcome`	`Enum`	Результат выполнения операции в стандартизированном виде. Возможные значения: • 1 — успех; • 2 — неудача; • 3 — попытка; • 4 — операция выполняется; • 5 — результат неизвестен; • 6 — операция выполнена частично.
`focus`	`Enum`	Фокус внимания, которое требует событие (уровень актуальности для аналитика). Возможные значения: • 1 — низкий; • 2 — средний; • 3 — высокий; • 4 — критический; • 5 — неизвестный.
`name`	`LCString`	Краткое наименование события.

Ключ поля	Тип данных	Описание
`device_class`	`LCString`	Класс устройства, определяющий его тип и функциональное назначение в инфраструктуре.
`device_vendor`	`LCString`	Производитель устройства или ПО, с которого получено событие.
`device_product`	`LCString`	Название продукта или модели устройства, с которого получено событие.
`device_version`	`LCString`	Версия продукта или ПО устройства, с которого получено событие.
`device_action`	`LCString`	Действие, которое предпринято устройством.
`device_outcome`	`LCString`	Результат выполнения операции из события.
`device_severity`	`LCString`	Уровень важности события. Значение берется из базового события. Например: `low`, `medium`, `high`, `critical`.
`device_event_category`	`LCString`	Класс базового события, определенный в самом событии.
`device_event_class`	`LCString`	Класс (категория) события. Некоторые источники событий определяют их категорию.
`device_event_code`	`LCString`	Код события, полученный от устройства. Например, код ошибки.
`device_event_id`	`LCString`	Уникальный идентификатор сообщения в системе устройства.
`device_message`	`String`	Сообщение от устройства.
`device_reason`	`LCString`	Причина появления события. Может содержать ошибку или код. Например, `invalid_signature` или `0x1234`.
`device_hostname`	`LCString`	Имя устройства без доменного суффикса.
`device_ip`	`IPv6`	Основной IP-адрес устройства, IPv4 или IPv6.  IP-адрес версии IPv4 требуется приводить к IPv6 в рамках правила нормализации событий.
`device_domain`	`LCString`	Доменное имя устройства.
`device_fqdn`	`LCString`	Полное доменное имя устройства.
`device_mac`	`MAC`	MAC-адрес основного сетевого интерфейса устройства.

Ключ поля	Тип данных	Описание
device_direction	Enum	Направление сетевого взаимодействия из базового события. Возможные значения: • -1 — не указано; • 0 — входящее соединение; • 1 — исходящее соединение.
device_interface_in	LCString	Имя сетевого интерфейса для входящего трафика.
device_interface_out	LCString	Имя сетевого интерфейса для исходящего трафика.
device_nat_ip	IPv6	IP-адрес устройства после трансляции (NAT).
device_nat_port	UInt16	Преобразованный порт источника после NAT для устройства.
protocol	LCString	Используемый сетевой протокол. Например: TCP, UDP, ICMP, HTTP.
count_bytes_in	UInt32	Объем входящего сетевого трафика в байтах, связанного с событием. Может быть использован для: • анализа сетевой активности; • выявления аномальных объемов трафика; • расчета коэффициентов входящей и исходящей нагрузки.
count_bytes_out	UInt32	Объем исходящего сетевого трафика в байтах, связанного с событием. Может быть использован для: • мониторинга исходящих соединений; • обнаружения утечек данных; • балансировки сетевой нагрузки.
agent_id	LCString	Идентификатор агента, от которого получено событие.
agent_hostname	LCString	Имя хоста агента, от которого получено событие.
agent_ip	LCString	IP-адрес агента, от которого получено событие.  IP-адрес версии IPv4 требуется приводить к IPv6 в рамках правила нормализации событий.
agent_name	LCString	Имя агента, от которого получено событие.
agent_type	LCString	Тип агента, от которого получено событие.
agent_version	LCString	Версия агента, от которого получено событие.
src_hostname	LCString	Имя исходного хоста без доменного суффикса.

Ключ поля	Тип данных	Описание
src_ip	IPv6	IP-адрес исходного узла, IPv4 или IPv6.  IP-адрес версии IPv4 требуется приводить к IPv6 в рамках правила нормализации событий.
src_port	UInt16	Номер порта исходного узла.
src_domain	LCString	Доменное имя исходного узла без поддоменов.
src_fqdn	LCString	Полное доменное имя исходного узла.
src_mac	MAC	MAC-адрес исходного сетевого интерфейса.
src_nat_ip	IPv6	Преобразованный IP-адрес исходного узла после NAT.
src_nat_port	UInt16	Преобразованный номер порта исходного узла после NAT.
src_geo_city	LCString	Город расположения исходного узла. Определяется по IP-адресу.
src_geo_country	LCString	Двухбуквенный код страны исходного узла согласно стандарту ISO 3166-1 alpha-2.
src_process_id	UInt64	Идентификатор (PID) исходного процесса.
src_process_name	LCString	Имя исполняемого файла исходного процесса.
src_process_cmdline	String	Команда запуска исходного процесса в командной строке.
src_process_cwd	String	Рабочая директория исходного процесса.
src_process_parent_path_full	String	Полный абсолютный путь к исполняемому файлу родительского процесса.
src_process_path	String	Путь к исполняемому файлу исходного процесса.
src_process_path_full	String	Полный путь к исполняемому файлу исходного процесса.
src_process_guid	UUID	Глобально уникальный идентификатор (GUID) исходного процесса.
src_process_hash_imphash	LCString	Импорт-хэш (IMPHASH) исполняемого файла исходного процесса.
src_process_hash_md5	LCString	Хэш MD5 исполняемого файла исходного процесса.
src_process_hash_sha	LCString	Хэш SHA-1 исполняемого файла исходного процесса.
src_process_hash_sha256	LCString	Хэш SHA-256 исполняемого файла исходного процесса.
src_process_hash_sha512	LCString	Хэш SHA-512 исполняемого файла исходного процесса.
src_process_parent_id	UInt64	Идентификатор родительского процесса (PPID) для исходного процесса.
src_process_parent_name	String	Имя исполняемого файла родительского процесса для исходного процесса.

Ключ поля	Тип данных	Описание
src_process_parent_cmdline	String	Команда запуска родительского процесса для исходного процесса в командной строке.
src_process_parent_path	String	Путь к каталогу исполняемого файла родительского процесса, без имени файла.
src_process_parent_guid	UUID	Глобально уникальный идентификатор (GUID) родительского процесса для исходного процесса.
src_user_id	LCString	Уникальный идентификатор исходного пользователя в системе. Например, логин или SAMAccountName.
src_user_name	LCString	Полное имя исходного пользователя (ФИО или отображаемое имя).
src_user_type	LCString	Тип учетной записи исходного пользователя. Например: local, domain, service, guest.
src_user_domain	LCString	Доменное имя исходного пользователя.
src_user_email	LCString	Адрес электронной почты исходного пользователя в формате RFC 5322.
src_user_dn	LCString	Отличительное имя (Distinguished Name) исходного пользователя в LDAP или Active Directory.
src_user_privileges	LCString	Уровень привилегий исходного пользователя. Например: User, Admin, Service.
src_user_session_id	LCString	Уникальный идентификатор сеанса исходного пользователя в системе.
src_service_id	LCString	Уникальный идентификатор исходного сервиса в системе.
src_service_name	LCString	Название исходного сервиса или приложения.
src_group_id	LCString	Уникальный идентификатор исходной группы пользователей или ресурсов.
src_group_name	LCString	Название исходной группы.
src_group_domain	LCString	Название домена исходной группы.
src_group_type	LCString	Тип группировки. Например: security, department, role, project.
src_file_id	LCString	Уникальный идентификатор исходного файла.
src_file_name	LCString	Имя исходного файла с расширением.
src_file_extension	LCString	Расширение исходного файла. Например: .pdf, .exe, .docx.

Ключ поля	Тип данных	Описание
src_file_owner	LCString	Владелец файла (пользователь или группа).
src_file_path	String	Относительный путь к исходному файлу, включая имя файла.
src_file_path_full	String	Абсолютный путь к исходному файлу в файловой системе.
src_file_permission	LCString	Права доступа к исходному файлу. Например, строка разрешений в Unix или ACL в Windows.
src_file_type	LCString	Тип исходного файла. Например: PDF, EXE, DOCX.
src_file_size	UInt64	Размер исходного файла в байтах.
src_file_sign	String	Информация о цифровой подписи файла при ее наличии.
src_file_time_creation	DateTime64	Дата и время создания исходного файла в формате ISO 8601.
src_file_time_modified	DateTime64	Дата и время последнего изменения исходного файла в формате ISO 8601.
src_file_hash_imphash	LCString	Импорт-хэш (IMPHASH) исходного файла.
src_file_hash_md5	LCString	Хэш MD5 исходного файла.
src_file_hash_sha	LCString	Хэш SHA-1 исходного файла.
src_file_hash_sha256	LCString	Хэш SHA-256 исходного файла.
src_file_hash_sha512	LCString	Хэш SHA-512 исходного файла.
src_object_class	LCString	Класс исходного объекта (Registry).
src_object_id	LCString	Уникальный идентификатор исходного объекта в системе.
src_object_name	LCString	Наименование исходного объекта.
src_object_type	LCString	Тип исходного объекта. Например: RegistryKey, Service.
src_object_value	LCString	Значение исходного объекта — например, ключа реестра или переменной.
src_object_version	LCString	Версия исходного объекта — например, исполняемого файла или библиотеки.
src_object_cmdline	String	Команда запуска исходного объекта в командной строке.
src_object_path	String	Путь к расположению исходного объекта без его имени.
src_object_path_full	String	Полный абсолютный путь к исходному объекту.
src_object_guid	UUID	Глобально уникальный идентификатор (GUID) исходного объекта.

Ключ поля	Тип данных	Описание
src_object_sign	LCString	Информация о подписи исходного объекта.
src_object_hash_md5	LCString	Хэш MD5 содержимого исходного объекта.
src_object_hash_sha	LCString	Хэш SHA-1 содержимого исходного объекта.
src_object_hash_sha256	LCString	Хэш SHA-256 содержимого исходного объекта.
src_query	String	Полный текст исходного запроса в оригинальном формате.
src_query_domain	LCString	Целевой объект запроса: например, домен, таблица, коллекция или сервис.
src_query_method	LCString	Метод, команда или тип операции запроса: • Для HTTP: GET, POST, PUT, DELETE. • Для SQL: SELECT, INSERT, UPDATE. • Для API: название вызываемого метода.
src_query_version	LCString	Версия протокола, используемого в запросе: • Для HTTP: 1.0, 1.1, 2.0. • Для TLS: 1.2, 1.3. • Для баз данных: версия протокола СУБД.
src_query_agent	String	Заголовок User-Agent, переданный в запросе. Содержит информацию о клиентском приложении, операционной системе и ее версии.
src_query_type	LCString	Тип или протокол запроса. Например: SQL, LDAP, HTTP, GraphQL, REST, SOAP, DNS.
src_query_cookies	String	Файлы Cookie исходного запроса.
src_query_response	String	Полный ответ на выполненный запрос в исходном формате: JSON, XML или бинарном.
dst_fqdn	LCString	Полное доменное имя целевого узла.
dst_hostname	LCString	Имя целевого хоста без доменного суффикса.
dst_ip	IPv6	IP-адрес целевого узла, IPv4 или IPv6.  IP-адрес версии IPv4 требуется приводить к IPv6 в рамках правила нормализации событий.
dst_port	UInt16	Номер порта целевого узла.
dst_domain	LCString	Доменное имя целевого узла без поддоменов.
dst_mac	MAC	MAC-адрес целевого сетевого интерфейса.
dst_nat_ip	IPv6	Преобразованный IP-адрес назначения после NAT.

Ключ поля	Тип данных	Описание
dst_nat_port	UInt16	Преобразованный порт назначения после NAT.
dst_geo_city	LCString	Город расположения целевого узла. Определяется по IP-адресу.
dst_geo_country	LCString	Двухбуквенный код страны целевого узла согласно стандарту ISO 3166-1 alpha-2.
dst_process_id	UInt64	Идентификатор (PID) целевого процесса.
dst_process_name	LCString	Имя исполняемого файла целевого процесса.
dst_process_cmdline	String	Команда запуска целевого процесса в командной строке.
dst_process_cwd	String	Рабочая директория целевого процесса.
dst_process_path	String	Путь к исполняемому файлу целевого процесса.
dst_process_path_full	String	Полный путь к исполняемому файлу целевого процесса.
dst_process_guid	UUID	Глобально уникальный идентификатор (GUID) целевого процесса.
dst_process_hash_imphash	LCString	Импорт-хэш (IMPHASH) исполняемого файла целевого процесса.
dst_process_hash_md5	LCString	Хэш MD5 исполняемого файла целевого процесса.
dst_process_hash_sha	LCString	Хэш SHA-1 исполняемого файла целевого процесса.
dst_process_hash_sha256	LCString	Хэш SHA-256 исполняемого файла целевого процесса.
dst_process_hash_sha512	LCString	Хэш SHA-512 исполняемого файла целевого процесса.
dst_process_parent_cmdline	String	Команда запуска родительского процесса для целевого процесса в командной строке.
dst_process_parent_id	UInt64	Идентификатор родительского процесса (PPID) для целевого процесса.
dst_process_parent_name	String	Имя исполняемого файла родительского процесса для целевого процесса.
dst_process_parent_path	String	Путь к каталогу исполняемого файла родительского процесса, без имени файла.
dst_process_parent_path_full	String	Полный абсолютный путь к исполняемому файлу родительского процесса.
dst_process_parent_guid	UUID	Глобально уникальный идентификатор (GUID) родительского процесса для целевого процесса.
dst_user_id	LCString	Уникальный идентификатор целевого пользователя в системе. Например, логин или SAMAccountName.
dst_user_name	LCString	Полное имя целевого пользователя (ФИО или отображаемое имя).

Ключ поля	Тип данных	Описание
`dst_user_type`	`LCString`	Тип учетной записи целевого пользователя. Например: `local`, `domain`, `service`, `guest`.
`dst_user_domain`	`LCString`	Доменное имя целевого пользователя.
`dst_user_email`	`LCString`	Адрес электронной почты целевого пользователя в формате RFC 5322.
`dst_user_dn`	`LCString`	Отличительное имя (Distinguished Name) целевого пользователя в LDAP или Active Directory.
`dst_user_privileges`	`LCString`	Уровень привилегий целевого пользователя. Например: `User`, `Admin`, `Service`.
`dst_user_session_id`	`LCString`	Уникальный идентификатор сеанса целевого пользователя в системе.
`dst_service_id`	`LCString`	Уникальный идентификатор целевого сервиса в системе.
`dst_service_name`	`LCString`	Название целевого сервиса или приложения.
`dst_group_id`	`LCString`	Уникальный идентификатор целевой группы пользователей или ресурсов.
`dst_group_name`	`LCString`	Название целевой группы.
`dst_group_domain`	`LCString`	Название домена целевой группы.
`dst_group_type`	`LCString`	Тип группировки. Например: `security`, `department`, `role`, `project`.
`dst_file_id`	`LCString`	Уникальный идентификатор целевого файла.
`dst_file_name`	`LCString`	Имя целевого файла с расширением.
`dst_file_path`	`String`	Относительный путь к целевому файлу, включая имя файла.
`dst_file_path_full`	`String`	Абсолютный путь к целевому файлу в файловой системе.
`dst_file_extension`	`LCString`	Расширение целевого файла. Например: `.pdf`, `.exe`, `.docx`.
`dst_file_owner`	`LCString`	Владелец файла (пользователь или группа).
`dst_file_permission`	`LCString`	Права доступа к целевому файлу. Например, строка разрешений в Unix или ACL в Windows.
`dst_file_type`	`LCString`	Тип целевого файла. Например: `PDF`, `EXE`, `DOCX`.
`dst_file_size`	`UInt64`	Размер целевого файла в байтах.

Ключ поля	Тип данных	Описание
dst_file_sign	String	Информация о цифровой подписи файла при ее наличии.
dst_file_time_creation	DateTime64	Дата и время создания файла в формате ISO 8601.
dst_file_time_modified	DateTime64	Дата и время последнего изменения файла в формате ISO 8601.
dst_file_hash_imphash	LCString	Импорт-хэш (IMPHASH) целевого файла.
dst_file_hash_md5	LCString	Хэш MD5 целевого файла.
dst_file_hash_sha	LCString	Хэш SHA-1 целевого файла.
dst_file_hash_sha256	LCString	Хэш SHA-256 целевого файла.
dst_file_hash_sha512	LCString	Хэш SHA-512 целевого файла.
dst_object_class	LCString	Класс целевого объекта (Registry).
dst_object_id	LCString	Уникальный идентификатор целевого объекта в системе.
dst_object_name	LCString	Наименование целевого объекта.
dst_object_type	LCString	Тип целевого объекта. Например: RegistryKey, Service.
dst_object_value	LCString	Значение целевого объекта — например, ключа реестра или переменной.
dst_object_version	LCString	Версия целевого объекта — например, исполняемого файла или библиотеки.
dst_object_path	String	Путь к расположению целевого объекта без его имени.
dst_object_path_full	String	Полный абсолютный путь к целевому объекту.
dst_object_cmdline	String	Команда запуска целевого объекта в командной строке.
dst_object_guid	UUID	Глобально уникальный идентификатор (GUID) целевого объекта.
dst_object_sign	LCString	Информация о подписи целевого объекта.
dst_object_hash_md5	LCString	Хэш MD5 содержимого целевого объекта.
dst_object_hash_sha	LCString	Хэш SHA-1 содержимого целевого объекта.
dst_object_hash_sha256	LCString	Хэш SHA-256 содержимого целевого объекта.
logon_method	LCString	Метод аутентификации, использованный при входе в систему.

Ключ поля	Тип данных	Описание
`logon_type`	`LCString`	Способ аутентификации пользователя. Возможные значения: • `interactive` — локальный вход; • `network` — сетевой вход; • `batch` — пакетное задание; • `service` — служба; • `unlock` — разблокировка; • `remoteinteractive` — RDP; • `cached` — кэшированные учетные данные.
`end_time`	`DateTime64`	Дата и время завершения действия в событии в формате ISO 8601 (UTC).
`start_time`	`DateTime64`	Дата и время начала действия в событии в формате ISO 8601 (UTC).
`duration`	`UInt16`	Длительность события в секундах или миллисекундах.
`mitre_tts`	Массив `LCString`	Список техник и тактик MITRE ATT&CK.
`model_version`	`LCString`	Текущая версия модели данных, в которую было записано событие.
`normalization_rule_id`	`LCString`	Идентификатор правила нормализации, которое обработало событие.
`normalization_rule_name`	`LCString`	Название правила нормализации, которое обработало событие.
`normalization_rule_version`	`LCString`	Версия правила нормализации, которое обработало событие.
`aggregation_rule_version`	`LCString`	Версия правила агрегации, которое обработало событие.

Ключ поля	Тип данных	Описание
`correlation_rule_type`	`Enum`	Тип правила корреляции, сформировавшего событие. Возможные значения: • 1 — `alert`: событие, сформированное правилом, будет использоваться в другом правиле корреляции; • 2 — `detect`: правило детектирования; • 3 — `hunt`: хантинговое правило, которое используется, чтобы обнаружить аномальную активность на хосте и выявить действия атакующего; • 4 — `profile`: правило на этапе профилирования; • 5 — `test`: правило на этапе тестирования; • 6 — `debug`: правило на этапе разработки и отладки; • 7 — `sync`: правило для работы с активными списками.
`correlation_rule_version`	`LCString`	Версия правила корреляции, сформировавшего событие.

Типы данных

Раздел содержит описания типов данных, поддерживаемых в системе, с особенностями их использования и примерами.

Поддерживаемые типы данных в системе

Тип данных	Описание	Пример

`DateTime`
 	
`DateTime64`
	Дата и время.
	
	Допустимые форматы значения:
	
	• `YYYY-MM-DDThh:mm:ss+/-hh:mm`;
	• `DD.MM.YYYY hh:mm:ss`;
	• `DD.MM.YYYY hh:mm:ss.sss` (для `DateTime64`) .
	
	Здесь:
	
	• `YYYY-MM-DD` — дата в формате "год-месяц-день".
	• `DD.MM.YYYY` — дата в формате "день.месяц.год".
	• `T` — начало обозначения времени.
	• `hh:mm:ss` — время в формате "час:минута:секунда".
	• `sss` — миллисекунды.
	• `+hh:mm` или `-hh:mm` — разница между часовым поясом пользователя и часовым поясом UTC+0.
	
	При использовании формата `DD.MM.YYYY hh:mm:ss` или `DD.MM.YYYY hh:mm:ss.sss` время указывается относительно часового пояса пользователя.
	
	Формат отображения дат и времени в системе задается в настройках профиля.
	
	В полях событий дата и время могут задаваться следующими способами:
	
	• `"<DateTime-значение>"` — используется для хранения даты-времени в виде строки. Например: `"15.05.2024 06:28:15"`.
	• `t'<DateTime-значение>'` — используется для возможности корреляции данных по дате-времени. Например: `t'15.05.2024 06:28:15'`.
		`2024-05-15T10:28:15-04:00`
		
		`15.05.2024 06:28:15`
		
		`15.05.2024 06:28:15.023` (для `DateTime64`)

`String`	Строка. Задается в двойных кавычках.	`"абвг123"`

Тип данных	Описание	Пример
`LCString`	Строка со словарным кодированием. Задается в двойных кавычках. Строковое значение записывается в словарь и заменяется на индекс его записи в словаре. При небольших размерах словаря (например, менее 10 000 записей) данные типа `LCString` обрабатываются быстрее по сравнению с типом `String`.	`"абв"`
`UInt`	Неотрицательное целое число фиксированной длины. Для определения диапазона значений требуется указать разрядность. Диапазоны значений, определяемые разрядностью: • `UInt8`: [0 : 2^{8}-1]; • `UInt16`: [0 : 2^{16}-1]; • `UInt32`: [0 : 2^{32}-1]; • `UInt64`: [0 : 2^{64}-1]; • `UInt128`: [0 : 2^{128}-1]; • `UInt256`: [0 : 2^{256}-1].	`255`
`Float`	Число с плавающей точкой. Для определения диапазона значений требуется указать разрядность. Приблизительные диапазоны значений, определяемые разрядностью: • `Float32`: [-3,4x10^{38} : 3,4x10^{38}]; • `Float64`: [-1,7x10^{308} : 1,7x10^{308}].	`0.999`
`UUID`	Универсальный уникальный идентификатор (UUID). Значение — пять групп из шестнадцатеричных цифр в формате `8-4-4-4-12`, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`	`61f0c404-5cb3-11e7-907b-a6006ad3dba0`

Тип данных	Описание	Пример

`Bool`	Логическое значение.	`True`
		
		`False`

`KeyValue`	Пара `ключ:значение`. Элементы пары указываются в фигурных скобках и разделяются двоеточием. Для ключа допускается только тип `String`. Для значения доступны следующие типы: • `DateTime`; • `DateTime64`; • `String`; • `LCString`; • `UInt`; • `Float32`; • `UUID`; • `IPv6`; • `IPv4`; • `MAC`.	`{"key1":"1"}`
`Enum`	Список предопределенных значений. Может хранить до 256 значений, пронумерованных в диапазоне [-128 : 127]. Элементы списка указываются в круглых скобках и разделяются запятой и пробелом.	`('high_severity', 'medium_severity', 'low_severity')`
`IPv4`	IPv4-адрес. Значение — четыре группы из трех десятичных цифр, разделенных точкой. Формат значения: `xxx.xxx.xxx.xxx`	`10.99.101.99`

Тип данных	Описание	Пример

`IPv6`	IPv6-адрес.
	
	Значение по стандарту RFC 5952 — восемь групп из четырех шестнадцатеричных цифр, разделенных двоеточием. Буквы — только строчные латинские.
	
	Формат значения:
	
	`xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`
	
	Допускается использование сжатой формы записи IPv6 в соответствии со следующими правилами:
	
	1. В каждой группе ведущие нули (нули слева) удаляются.
	2. Если группа состоит из одних нулей, она заменяется на один ноль.
	3. Одна непрерывная последовательность из нулей (самая длинная) заменяется символом двойного двоеточия (`::`). Может применяться только один раз.
	
	Допускается хранение IPv4 в нотации IPv6:
	
	1. Последние 32 младших бита записываются как IPv4-адрес.
	2. В начале идут только нулевые поля и одно единичное, записанные в нотации IPv6.
		`2a01:1f34::8a2e:765d`
		
		`2a02:e980:1e::1`

`MAC`	MAC-адрес. Значение — шесть двузначных шестнадцатеричных чисел, разделенных двоеточием. Буквы — только заглавные латинские. Формат значения: `XX:XX:XX:XX:XX:XX`	`00:53:00:B8:DF:B8`

Тип данных	Описание	Пример
`Array`	Массив значений заданного типа данных. Элементы массива указываются в квадратных скобках и разделяются запятой и пробелом. Допустимые типы данных значений массива: • `DateTime`; • `DateTime64`; • `String`; • `LCString`; • `UInt`; • `Float32`; • `UUID`; • `IPv6`; • `IPv4`; • `MAC`.	`[321, 123, 12344]`
`JSON`	Данные, представленные в формате JSON. Поддерживаются как пары "ключ: значение", так и структуры со вложенными объектами и массивами.	`{ "timestamp": "2025-08-15T15:07:89Z", "source": { "system": "firewall" }, "events": [ { "event_id": "evt-001" }, { "event_id": "evt-002" } ] }`

Типы операций в событии

Универсальная модель события 2.0 включает поле `action` — тип операции, которая была зафиксирована в событии. Значение поля используется для классификации событий и определения их семантики в контексте безопасности.

В таблице ниже перечислены возможные типы операций. Таблица содержит следующие столбцы:

- **Значение** — числовой код типа, который будет записан в поле `action` событий.

- **Наименование** — краткое наименование операции.
- **Примеры интерпретации** — примеры интерпретации типа операции.



В таблице приведены только некоторые типовые примеры интерпретации значений кодов. При работе с системой вам следует интерпретировать коды самостоятельно в зависимости от контекста возникновения событий.

Возможные типы операций

Значение	Наименование	Примеры интерпретации
1	`access`	• Выполнена аутентификация в системе. • Выполнено обращение к базе данных.
2	`admin`	• Выполнен запрос к API от имени администратора. • Открыт интерфейс администратора веб-приложения.
3	`allow`	• Разрешена установка сетевого соединения. • Сетевой фильтр разрешил выполнение запроса.
4	`alter`	Изменены данные профиля пользователя.
5	`apply`	К таблице базы данных применена политика безопасности.
6	`auth`	Выполнена попытка аутентификации.
7	`await`	Клиент ожидает новых электронных писем.
8	`backup`	• Выполняется резервное копирование файлов. • Выполняется резервное копирование таблиц базы данных.
9	`block`	• Заблокирована подозрительная сетевая активность. • Заблокирован выполняемый процесс.
10	`boot`	Операционная система запускается.
11	`call`	Windows: вызван метод WMI.
12	`cancel`	Установка пакета отменена.
13	`change`	• Изменен пароль пользователя. • Изменена конфигурация приложения.
14	`check`	• Выполнена проверка статуса запроса. • Выполнена проверка прав доступа к файлу.
15	`close`	Закрыта сетевая сессия.

Значение	Наименование	Примеры интерпретации
16	`connect`	• Установлено сетевое соединение. • Подключено внешнее устройство.
17	`copy`	• Скопирован файл. • Скопирован ключ реестра.
18	`create`	• Создан пользователь. • Создано правило файрвола.
19	`delay`	Отправка электронного письма задержана.
20	`delete`	• Выполнен запрос API на удаление. • Удален файл.
21	`deny`	• Запрос отклонен сетевым фильтром. • Электронное письмо отклонено почтовым фильтром.
22	`detect`	• Обнаружено вредоносное ПО. • Обнаружена попытка эксплуатации уязвимости.
23	`disable`	• Отключена учетная запись пользователя. • Отключена политика аудита конфигурации.
24	`disconnect`	• Сетевое соединение прервано. • Отключено внешнее устройство.
25	`down`	Сетевой интерфейс недоступен.
26	`download`	• Скачивается пакет. • По сети скачиваются данные.
27	`drop`	Удалена таблица в базе данных.
28	`enable`	• Активирована учетная запись. • Включен сетевой фильтр.
29	`end`	Завершена работа процесса.
30	`error`	• Произошла ошибка службы. • Произошла ошибка доступа к базе данных.
31	`execute`	Осуществляется атака на приложение.
32	`exploit`	Произошла попытка эксплуатации уязвимости.
33	`find`	Выполняется поиск пакета в репозитории.
34	`halt`	Linux: произошла остановка операционной системы без выключения питания.
35	`inbound`	Событие связано с входящим трафиком.
36	`incoming`	Событие связано с входящим трафиком.

Значение	Наименование	Примеры интерпретации
37	`info`	• Выполнен запрос API на получение информации. • Получена информация о драйвере.
38	`inject`	Обнаружена атака типа "внедрение процесса" (process injection).
39	`insert`	Выполнена вставка данных в базу данных.
40	`install`	• Выполнена установка пакета. • Выполнена установка драйвера внешнего устройства.
41	`launch`	Зафиксирован запуск атаки на приложение.
42	`load`	• Выполнена загрузка библиотеки. • Выполнена загрузка драйвера.
43	`lock`	Пользователь включил блокировку экрана.
44	`login`	Выполнена аутентификация пользователя в системе.
45	`logoff`	Windows: завершен сеанс выхода пользователя.
46	`logon`	Windows: начался сеанс входа пользователя.
47	`logout`	Выход пользователя из системы.
48	`modify`	• Изменены параметры конфигурации. • Изменены значения ключей реестра.
49	`notify`	Получено уведомление от службы мониторинга.
50	`open`	• Открыт сокет. • Открыто электронное письмо.
51	`outbound`	Событие связано с исходящим трафиком.
52	`outgoing`	Событие связано с исходящим трафиком.
53	`password_change`	Изменен пароль пользователя.
54	`progress`	Электронное письмо находится в процессе доставки.
55	`query`	Выполнен запрос на получение сетевой конфигурации.
56	`read`	Выполнено чтение файла.
57	`receive`	Получено электронное письмо.
58	`reject`	• Отклонен запрос на установление соединения. • Отклонен запрос к веб-приложению.

Значение	Наименование	Примеры интерпретации
59	remove	• Удалена учетная запись. • Удален пакет.
60	request	Отправлен запрос к веб-приложению.
61	reset	Перезапущен сеанс в системе.
62	response	Отправлен ответ веб-приложения.
63	run	Запущен процесс.
64	save	Windows: сохранена коллекция файлов.
65	scan	Произведено сканирование на наличие вредоносного ПО.
66	select	Выполнен запрос на чтение из базы данных.
67	send	• Отправлены данные по сети. • Отправлено электронное письмо.
68	set	• Пользователю назначена роль. • Установлен ключ реестра.
69	set_value	Установлено значение ключа реестра.
70	shutdown	Отключено питание устройства.
71	start	• Подключено питание устройства. • Запущена задача по расписанию.
72	startup	Запущен сервер базы данных.
73	stop	• Остановлена запланированная задача. • Остановлена служба.
74	teardown	Сетевое соединение удалено из таблицы соединений.
75	terminate	• Завершено выполнение процесса. • Завершено сетевое соединение.
76	uninstall	• Удален пакет. • Удален драйвер внешнего устройства.
77	unlock	Пользователь разблокировал экран.
78	up	Сетевой интерфейс доступен.
79	update	• Обновлен пакет. • Обновлена информация о пользователе.
80	use	Windows: использован метод WMI.
81	write	Осуществлена запись в файл.

Значение	Наименование	Примеры интерпретации
82	`get`	• Получены данные или значения. • Получено значение, установленное операцией `set`.

Работа с базами данных событий

Данный раздел описывает процесс работы с базами данных событий в системе. Работа осуществляется в разделе **Ресурсы** → **Базы данных событий** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О базах данных событий

Базы данных (БД) событий используются для хранения событий в системе. Единицей хранения событий в БД является хранилище событий: на уровне БД хранилище представляет собой таблицу. Столбцы и типы данных хранилища задаются моделью события.

В системе доступны БД событий следующих типов:

- **Системные БД событий** — регистрируются системой автоматически. Их нельзя изменять или удалять.

  Системные БД не могут быть использованы для создания хранилищ событий и активных списков.

  Существует системная БД событий аудита, которая лежит в основе системного хранилища событий, предназначенного для хранения событий сервиса аудита.

- **Пользовательские БД событий** — регистрируются вручную.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Зарегистрировать** (+) позволяет зарегистрировать новую БД событий.
- Поле **Поиск** предназначено для быстрого поиска БД событий в отображаемом списке по названию.

В рабочей области раздела отображается таблица имеющихся БД событий. Таблица содержит следующие столбцы:

- **Название** — название БД, используемое для ее идентификации в системе.
- **Статус** — текущий статус БД: **Активна** или **Недоступна**.
- **Идентификатор сервиса** — уникальный идентификатор сервиса ClickHouse, в котором

зарегистрирована БД событий, в системе обнаружения сервисов NATS Service Discovery.



Здесь и далее **сервис ClickHouse** обозначает сервис кластера Kubernetes, в котором работает СУБД ClickHouse.

- **Описание** — описание БД.
- **Дата создания** — дата и время создания БД.
- **Дата изменения** — дата и время последнего изменения БД.



Если в томах базы данных заканчивается свободное пространство, слева от ее записи в таблице отображается иконка  .

При работе с таблицей БД событий доступны следующие операции:

- Поиск баз данных по полю **Название**.
- Сортировка баз данных по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретной БД в правой части рабочей области отображается ее карточка с детальной информацией.

Работа с базами данных событий

Доступные операции над БД событий:

- Регистрация базы данных событий
- Просмотр базы данных событий
- Настройки томов базы данных событий
- Настройка хранения данных в базе данных событий по умолчанию
- Изменение базы данных событий
- Удаление базы данных событий

Регистрация базы данных событий

Чтобы зарегистрировать БД событий:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на кнопку **Зарегистрировать** (+). Отобразится окно регистрации БД событий.
3. Введите название БД событий.
4. Введите идентификатор сервиса ClickHouse, в котором необходимо зарегистрировать БД событий, в NATS Service Discovery.



Чтобы получить идентификатор сервиса ClickHouse в NATS Service Discovery, обратитесь к администратору системы или воспользуйтесь разделом

Получение идентификатора сервиса в NATS Service Discovery.

5. При необходимости введите описание БД событий.
6. Нажмите на кнопку **Зарегистрировать**. Система создаст БД событий и отобразит соответствующее уведомление. Новая база отобразится в списке раздела **Ресурсы → Базы данных событий**.

Просмотр базы данных событий

Чтобы просмотреть БД событий:

1. Перейдите в раздел **Ресурсы → Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.

В верхней части карточки отображается название БД событий. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранной БД.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка БД событий содержит следующие вкладки:

- **Информация** — общая информация о БД событий.
- **Тома** — информация о томах хранения данных в БД.

Вкладка Информация

На вкладке **Информация** отображаются основные сведения о БД событий:

- Метрики ресурсов сервиса ClickHouse, в котором зарегистрирована БД событий:
  - **CPU** — количество ядер и уровень загруженности процессора сервиса.
  - **RAM** — объем и уровень загруженности оперативной памяти сервиса.
  - **HDD** — объем и уровень загруженности жесткого диска сервиса.



  - Метрики ресурсов сервиса ClickHouse вычисляются суммарно по всем зарегистрированным в нем БД событий.

    Если используется вынесенная инсталляция ClickHouse, метрики CPU и RAM могут содержать погрешности из-за влияния процессов операционных систем, установленных на узлах ClickHouse.

    При расчете метрик учитываются все тома БД событий, в том числе том служебных данных **default**.
  - Расчет занятого и общего объема на жестком диске (HDD) осуществляется с суммированием соответствующих показателей по всем томам БД. Таким

образом, если тома расположены на одном физическом диске, то итоговые показатели HDD для БД могут превышать фактические значения.

- Если при получении данных о метриках произошла ошибка, в карточке отобразится соответствующее сообщение с текстом ошибки.

- **ID** — уникальный идентификационный код БД, генерируемый автоматически при ее создании.
- **Название** — название БД, используемое для ее идентификации в системе.
- **Идентификатор сервиса** — уникальный идентификатор сервиса ClickHouse, в котором зарегистрирована БД, в системе обнаружения сервисов NATS Service Discovery.
- **ID тенанта** — идентификатор тенанта, к которому относится БД.
- **Тип** — тип БД событий: системная или пользовательская.
- **Описание** — описание БД событий.
- **Статус** — текущий статус БД: **Активна** или **Недоступна**.
- **Хранилищ событий в БД** — количество хранилищ событий, содержащихся в БД.
- **Дата создания** — дата и время создания БД.
- **Создал** — пользователь, инициировавший создание БД.
- **Дата изменения** — дата и время последнего изменения БД.
- **Изменил** — пользователь, выполнивший последнее изменение БД.

Вкладка Тома

На вкладке **Тома** отображается информация о томах хранения данных.

Тома хранения данных в БД событий бывают следующих типов:

- **default** — том для хранения служебных данных ClickHouse.

  Настроить хранение событий в томе **default** нельзя.

- **hot_volume** — том для быстрого доступа к часто используемым данным (горячее хранение).
- **cold_volume** — том для хранения редко используемых данных (холодное хранение).

Для каждого тома отображаются следующие сведения:

- **Свободно/всего** — информация о свободном и общем объеме тома.
- **Уведомлять, если свободно менее** — порог свободного объема тома, при достижении которого система сформирует предупреждения в карточке БД и отобразит иконки ⓘ в таблице раздела.

  Для настройки порога свободного объема в томе предназначена кнопка **Настройка тома**.

  При исчерпании всего свободного объема тома происходит остановка конвейеров, связанных с текущей БД событий.

Для каждого тома доступна кнопка для его настройки.

В нижней части вкладки расположена кнопка для настройки хранения данных в БД по умолчанию.



Если при получении данных о томах произошла ошибка, в карточке отобразится соответствующее сообщение с текстом ошибки.

Настройки томов базы данных событий

Чтобы настроить том базы данных событий:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.
3. Перейдите на вкладку **Тома** в карточке БД событий. Система отобразит список томов БД.
4. Нажмите на кнопку **Настроить том** для требуемого тома. Отобразится окно его настройки.
5. Задайте пороговое значение свободного объема тома, при достижении которого система сформирует уведомление. Доступно задание относительного (в процентах) и абсолютного (в ГБ) свободного объема.
6. Нажмите на кнопку **Сохранить**. Система сохранит настройки тома базы данных и отобразит соответствующее уведомление.

Настройка хранения данных в базе данных событий по умолчанию

Настройки хранения данных в БД событий по умолчанию можно использовать для быстрой настройки хранения данных в хранилищах событий.

Чтобы настроить хранение данные в БД событий по умолчанию:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.
3. Перейдите на вкладку **Тома** в карточке БД событий. Система отобразит список томов БД.
4. Нажмите на кнопку **Настроить хранение по умолчанию** в нижней части карточки БД событий. Отобразится окно настроек хранения данных по умолчанию.
5. Задайте настройки хранения данных:
   a. Выберите из выпадающего списка том хранения данных:
      - **hot_volume** — том для быстрого доступа к часто используемым данным (горячее хранение).
      - **cold_volume** — том для хранения редко используемых данных (холодное хранение).
   b. Укажите срок хранения данных в днях.



Если поле для срока хранения данных оставить пустым, то срок хранения данных в этой БД будет неограниченным.

c. Выберите из выпадающего списка действие над данными в томе по окончании срока хранения:

   - **Переместить** — данные будут перемещены в другой том. Система добавит дополнительную строку для настройки нового тома, в который требуется переместить данные.
   - **Удалить** — данные в томе будут удалены.

6. Нажмите на кнопку **Сохранить**. Система сохранит настройки хранения данных в БД по умолчанию и отобразит соответствующее уведомление.

Изменение базы данных событий

Вы можете изменить БД событий следующими способами:

- через меню действий в карточке БД;
- через изменение полей в карточке БД.



Изменение доступно только для БД событий с типом **Пользовательская**.

Удаление базы данных событий

Вы можете удалить БД событий через меню действий в его карточке.



Удаление доступно только для БД событий с типом **Пользовательская**, не используемых в других сущностях системы.

Работа с хранилищами событий

Данный раздел описывает процесс работы с хранилищами событий в системе. Работа осуществляется в разделе **Ресурсы** → **Хранилища событий** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О хранилищах событий

Хранилище событий используется для хранения событий, поступивших в систему и обработанных коллектором. Оно представляет собой таблицу в БД событий. Формат хранилища определяется его моделью данных.

В системе доступны хранилища событий следующих типов:

- **Системные хранилища событий** — создаются системой автоматически. В них нельзя изменять название и описание, а также их нельзя удалять.



Системные хранилища не могут быть использованы в качестве хранилищ событий для конечных точек и сервиса оповещений.

- **Пользовательские хранилища событий** — создаются вручную.

Существует системное хранилище событий аудита, предназначенное для хранения событий сервиса аудита. Это хранилище построено на системной БД событий.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новое хранилище событий.
- Поле **Поиск** предназначено для быстрого поиска хранилищ событий в отображаемом списке по названию.

В рабочей области раздела отображается таблица имеющихся хранилищ событий. Таблица содержит следующие столбцы:

- **Название** — название хранилища событий, используемое для его идентификации в системе.
- **Описание** — описание хранилища событий.
- **База данных событий** — БД событий, в которой содержится хранилище.
- **Модель события** — модель события, используемая хранилищем.
- **Событий в хранилище** — количество событий в хранилище.
- **Дата создания** — дата и время создания хранилища событий.
- **Дата изменения** — дата и время последнего изменения хранилища событий.



Если в томах хранилища заканчивается свободное место, слева от его записи в таблице отображается иконка ⓘ.

При работе с таблицей хранилищ событий доступны следующие операции:

- Поиск хранилищ по полю **Название**.
- Сортировка хранилищ по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного хранилища в правой части рабочей области отображается его карточка с детальной информацией.

Работа с хранилищем событий

Доступные операции над хранилищами событий:

- Создание хранилища событий
- Просмотр хранилища событий
- Просмотр показателей сжатия по полям
- Обновление модели хранилища событий
- Изменение хранилища событий

- Удаление хранилища событий

Создание хранилища событий

Чтобы создать хранилище событий:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания хранилища событий.
3. Выберите из выпадающего списка БД событий, в которую требуется поместить хранилище. Система отобразит в нижней части поля для настройки томов хранения этой БД.

   В системных базах данных создание хранилищ событий не поддерживается.

4. Выберите из выпадающего списка модель события, которая будет использована для создаваемого хранилища.
5. Введите название хранилища событий.
6. При необходимости введите описание хранилища событий.
7. Задайте настройки хранения данных:

   a. Выберите том хранения данных:

      - **hot_volume** — том для быстрого доступа к часто используемым данным (горячее хранение).
      - **cold_volume** — том для хранения редко используемых данных (холодное хранение).

   b. Укажите срок хранения данных в днях.

      Если поле для срока хранения данных оставить пустым, то срок хранения данных в этом хранилище будет неограниченным.

   c. Выберите из выпадающего списка действие над данными в томе по окончании срока хранения:

      - **Переместить** — данные будут перемещены в другой том. Введите в появившейся строке настройки нового тома, в который требуется переместить данные.
      - **Удалить** — данные в томе будут удалены.

   При необходимости можно задать настройки хранения данных на основе настроек по умолчанию в выбранной БД. Для этого нажмите на кнопку **Применить настройки хранения по умолчанию БД**. Если настройки по умолчанию не заданы, система отобразит соответствующее сообщение.

8. Нажмите на кнопку **Создать**. Система создаст хранилище событий и отобразит соответствующее уведомление. Новое хранилище появится в списке раздела **Ресурсы** → **Хранилища событий**.

Просмотр хранилища событий

Чтобы просмотреть хранилище событий:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об

имеющихся хранилищах событий.

2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.

В верхней части карточки отображается название хранилища событий. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранного хранилища.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка хранилища событий содержит следующие вкладки:

- **Информация** — общая информация о хранилище событий.
- **Тома** — информация о томах хранения данных в хранилище.

Вкладка Информация

На вкладке **Информация** отображаются основные сведения о хранилище событий:

- **ID** — уникальный идентификационный код хранилища, генерируемый автоматически при его создании.
- **Название** — название хранилища, используемое для его идентификации в системе.
- **Описание** — описание хранилища событий.
- **ID тенанта** — идентификатор тенанта, к которому относится хранилище событий.
- **Тип** — тип хранилища событий: системное или пользовательское.
- **База данных событий** — БД событий, в которой содержится хранилище.
- **Модель события** — модель события, используемая хранилищем.

  Если модель содержит изменения, которые были внесены на уровне СУБД или при обновлении системы, в поле отображается кнопка для ее обновления в хранилище.

- **Событий в хранилище** — количество событий в хранилище.
- **Дата создания** — дата и время создания хранилища событий.
- **Создал** — пользователь, инициировавший создание хранилища.
- **Дата изменения** — дата и время последнего изменения хранилища событий.
- **Изменил** — пользователь, выполнивший последнее изменение хранилища.

На вкладке также отображаются показатели сжатия данных, хранящихся в таблице ClickHouse, которая ассоциирована с хранилищем событий:

- объем занимаемого пространства без сжатия;
- объем занимаемого пространства со сжатием;
- коэффициент сжатия, показывающий отношение занимаемого объема без сжатия к занимаемому объему со сжатием.

Для получения более подробной информации о сжатии предназначена кнопка **Подробнее**.

Если при получении данных о показателях сжатия произошла ошибка, в карточке отобразится соответствующее сообщение с текстом ошибки.

Вкладка Тома

На вкладке **Тома** отображается информация о томах хранения данных. Тома подгружаются в хранилище событий автоматически из БД событий, в которой содержится хранилище.

Тома хранения данных в хранилище событий бывают следующих типов:

- **hot_volume** — предназначен для быстрого доступа к часто используемым данным (горячее хранение).
- **cold_volume** — предназначен для хранения редко используемых данных (холодное хранение).

Для каждого тома отображаются следующие сведения:

- **Свободно/всего** — информация о свободном и общем объеме тома.
- **Количество событий в томе** — информация о количестве событий, хранящихся в томе.
- **Объем занимаемого места в томе** — информация о фактически занимаемом объеме тома.
- **Уведомлять, если свободно менее** — порог свободного объема тома, при достижении которого система уведомляет пользователя с помощью предупреждения в карточке хранилища и иконки ⓘ в таблице раздела.

  Настройка уведомления при достижении порога свободного объема тома выполняется в карточке БД событий, лежащей в основе текущего хранилища.

  При исчерпании всего свободного объема тома происходит остановка конвейеров, связанных с текущим хранилищем событий.

- **Срок хранения** — количество дней, в течение которых события хранятся в томе.

Если при получении данных о томах произошла ошибка, в карточке отобразится соответствующее сообщение с текстом ошибки.

Просмотр показателей сжатия по полям

Основная информация о сжатии данных в хранилище отображается на вкладке **Информация** в карточке хранилища.

Чтобы получить подробную информацию о показателях сжатия по каждому полю в модели события, лежащей в основе хранилища:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.

3. Убедитесь, что в карточке хранилища открыта вкладка **Информация**.
4. Нажмите на кнопку **Подробнее** справа от заголовка группы **Показатели сжатия** в карточке. Откроется окно показателей сжатия по полям.

В верхней части окна отображаются основные показатели сжатия данных, которые хранятся в таблице ClickHouse, ассоциированной с хранилищем событий:

- объем занимаемого пространства без сжатия;
- объем занимаемого пространства со сжатием;
- коэффициент сжатия, показывающий отношение занимаемого объема без сжатия к занимаемому объему со сжатием.

Также в окне представлена таблица показателей сжатия по всем полям модели события. Таблица содержит следующие столбцы:

- **Название** — название поля в модели события.
- **Объем без сжатия** — объем занимаемого полем пространства без сжатия.
- **Объем со сжатием** — объем занимаемого полем пространства со сжатием.
- **Коэффициент сжатия** — отношение занимаемого полем объема без сжатия к занимаемому объему со сжатием.



- Показатели сжатия по полям отображаются только после фактического выполнения оптимизации в хранилище и расчета показателей сжатия для каждого столбца.
- Из-за особенностей оптимизации столбцов в ClickHouse в ряде случаев показатели сжатия по полям могут быть равны 0.

При работе с таблицей доступны следующие операции:

- Поиск показателей по полю **Название**.
- Сортировка показателей по любому столбцу.

Вы также можете экспортировать таблицу показателей сжатия в файл формата CSV с помощью кнопки  .

Обновление модели хранилища событий

При внесении изменений в модель события через веб-интерфейс системы она автоматически обновляется во всех хранилищах, в которых используется.

Если изменения были внесены в пользовательскую модель на уровне СУБД или в системную модель при обновлении на более новую версию системы, то автоматического обновления модели в хранилищах не происходит. Это позволяет избежать сбоев на конвейерах обработки событий из-за неожиданного изменения моделей. В таком случае, чтобы использовать в хранилище актуальную версию модели события, ее необходимо обновить вручную. Для этого:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.

2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.
3. Убедитесь, что в карточке хранилища открыта вкладка **Информация**.
4. Нажмите на кнопку **Обновить** в поле **Модель события**. Система обновит модель события в хранилище до актуальной версии.



Если новая модель событий несовместима со старой, то при попытке обновления модели в хранилище произойдет ошибка. В таком случае для работы с новой версией модели необходимо пересоздать хранилище.

Изменение хранилища событий

Вы можете изменить хранилище событий следующими способами:

- через меню действий в карточке хранилища;
- через изменение полей в карточке хранилища.



- Изменение названия и описания доступно только для хранилищ событий с типом **Пользовательское**.
- При изменении хранилища доступен выбор новой модели события. Новую модель можно применить к хранилищу, если в составе ее полей содержатся все поля из старой модели.

  Поля моделей считаются одинаковыми, если у них совпадают параметры **Ключ поля**, **Тип поля** и **Может быть null**.

Удаление хранилища событий

Вы можете удалить хранилище событий через меню действий в его карточке.



Удаление доступно только для хранилищ событий с типом **Пользовательское**, не используемых в других сущностях системы.

Работа с активными списками

Данный раздел описывает процесс работы с активными списками в системе. Работа осуществляется в разделе **Ресурсы** → **Активные списки** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об активных списках

Активный список представляет собой двумерный массив данных для накопления информации, полученной из определенных полей событий, поступающих на коррелятор.

Активные списки предназначены для повышения эффективности корреляционных правил, упрощения процедур обогащения событий и сбора информации для последующего анализа. Они используются в системе для обеспечения перекрестной ссылки в процессе корреляционного анализа, выполняемого корреляторами.

Активные списки в системе используются в следующих ключевых сценариях:

- Реализация многоуровневых корреляционных сценариев, где одно правило обновляет список, а другое осуществляет его анализ.
- Использование в качестве справочных таблиц: загрузка эталонных данных и их вызов в рамках определенного правила.
- Работа с исключениями при корреляции данных.

В системе доступны активные списки следующих типов:

- **Системные активные списки** — создаются системой автоматически. Их нельзя изменять и удалять, но в них можно добавлять и импортировать записи.

  Существует два системных активных списка — черный и белый. Они используются в механизме исключений при корреляции.

- **Пользовательские активные списки** — создаются вручную или в результате импорта.

Один и тот же активный список может быть использован в разных корреляторах системы. При этом данные в пользовательский активный список можно добавлять только через элементы экспертизы в корреляторе, на котором применяется список.

В каждом списке пользователь системы имеет возможность задать конкретный состав полей и параметры хранения данных с помощью схемы активного списка. Эти списки могут содержать отдельные поля событий и обработанные данные, полученные из этих полей.

Для управления активными списками предусмотрены функции для получения, добавления и удаления записей. В ходе работы системы записи могут быть добавлены или удалены из активных списков с помощью этих функций.

Данные, занесенные в активный список, доступны для чтения и изменения другим элементам экспертизы, которые используют язык VRL.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый активный список.
- Поле **Поиск** предназначено для быстрого поиска активных списков в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся активных списков. При выборе конкретного активного списка на экране открывается карточка с детальной информацией о нем.

Таблица содержит следующие столбцы:

- **ID** — уникальный идентификационный код активного списка, генерируемый автоматически при его создании.
- **Схема** — название схемы, которая определяет конфигурацию и параметры активного списка.
- **База данных** — название базы данных событий, используемой активным списком.

  В системных базах данных активные списки создавать нельзя.

- **Тип** — тип создания активного списка: системный или пользовательский.
- **TTL** — время жизни записей в активном списке в секундах. Если имеет значение "0", то время жизни записей неограниченно.
- **Лимит записей** — максимально допустимое количество записей в активном списке.
- **Аудит записей** — статус активности аудита у активного списка:
  - ✓ — аудит включен.
  - — — аудит выключен.
- **Дата создания** — дата и время создания активного списка.
- **Дата изменения** — дата и время последнего изменения активного списка.
- **Ресурсы** — количество ресурсов системы, использующих активный список. При наведении курсора на количество ресурсов отображается всплывающее сообщение с полным перечнем связанных с активным списком ресурсов.
- **Версия схемы** — версия схемы активного списка, представленная в формате SemVer v2.0.

При работе с таблицей активных списков доступны следующие операции:

- Поиск активных списков по полю **Схема**.
- Сортировка активных списков по полям **Схема**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

Работа с активными списками

Доступные операции над активными списками:

- Создание активных списков
- Просмотр активных списков
- Настройка аудита записей активного списка
- Изменение активных списков
- Удаление активных списков

Создание активных списков



Создание вручную доступно только для пользовательских активных списков.

Чтобы создать активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания активного списка.
3. Выберите из выпадающего списка схему активного списка. В окне отобразятся сведения о выбранной схеме:
   - время жизни записей в секундах (TTL);
   - лимит записей;
   - поля активного списка, определенные схемой.

   Если в схеме активного списка отсутствует поле `ttl`, то время жизни записей по умолчанию не ограничено.
4. Выберите из выпадающего списка базу данных событий.
5. При необходимости введите описание активного списка.
6. При необходимости установите переключатель **Аудит записей активного списка** в активное положение.

   Включение аудита записей активного списка может существенно увеличить нагрузку. После включения выбранные события будут отображаться в общем списке событий аудита.

   Установите флажки слева от операций, которые необходимо регистрировать в хранилище событий аудита:
   - **Создание записей** — для регистрации операций создания новых записей.
   - **Изменение записей** — для регистрации операций изменения записей.
   - **Удаление записей** — для регистрации операций удаления записей.

     Удаление также включает окончание срока TTL записи.
7. Нажмите на кнопку **Создать**. Система создаст активный список и отобразит соответствующее уведомление. Новый активный список появится в разделе **Ресурсы** → **Активные списки**.

Просмотр активных списков

Чтобы просмотреть активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит экран с подробной информацией об активном списке.

В верхней части экрана отображается схема активного списка.

Чтобы вернуться к списку активных списков, нажмите на кнопку **Назад** в левом верхнем углу экрана.

Панель инструментов экрана включает в себя следующие компоненты:

- Кнопка позволяет изменить активный список.
- Кнопка позволяет удалить активный список.
- ID и описание активного списка предоставляют краткую информацию о его функциональности.
- Переключатель **Аудит записей активного списка** позволяет включить или выключить аудит активного списка.
- Кнопка позволяет выбрать тип событий для записи в хранилище событий аудита.

В рабочей области экрана отображается таблица имеющихся в активном списке записей. При выборе конкретной записи в правой части рабочей области отображается ее карточка с детальной информацией.

Над таблицей записей расположены следующие инструменты для управления записями:

- Кнопка **Добавить запись** ( ) позволяет добавить новую запись в активный список.
- Кнопки **Экспорт** ( ) и **Импорт** ( ) позволяют выгружать существующие записи из активного списка и загружать в активный список записи извне.
- Поле **Поиск** предназначено для быстрого поиска записей в активном списке по ключу.

Таблица содержит следующие столбцы:

- **Ключ** — ключ записи в активном списке.
- **Повторы** — количество повторов записи в активном списке.
- **Срок действия** — дата и время прекращения действия записи в активном списке.
- **Дата создания** — дата и время создания записи в активном списке.
- **Дата изменения** — дата и время последнего изменения записи в активном списке.

В таблице также присутствуют столбцы, соответствующие полям активного списка. Поля ключа обозначены иконкой в заголовках столбцов.

При работе с таблицей доступны следующие операции:

- Поиск записей по полю **Ключ**.
- Настройка отображения таблицы.

Настройка аудита записей активного списка

Система позволяет настроить отслеживание операций над записями активного списка, чтобы регистрировать соответствующие события в хранилище событий аудита.

Настроить аудит записей можно из экрана просмотра активного списка. Для этого:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит экран с подробной

информацией об активном списке.

3. Нажмите на кнопку ⚙ на панели инструментов. Система отобразит окно настройки аудита записей.
4. Установите флажки слева от операций, которые необходимо регистрировать в хранилище событий аудита:
   - **Создание записей** — для регистрации операций создания новых записей.
   - **Изменение записей** — для регистрации операций изменения записей.
   - **Удаление записей** — для регистрации операций удаления записей.

     Удаление также включает окончание срока TTL записи.
5. Нажмите на кнопку **Сохранить**. Система сохранит выбранные настройки аудита.
6. Установите переключатель **Аудит записей активного списка** на панели инструментов в активное положение. Система начнет отслеживать выбранные операции над записями активного списка.

Вы также можете настроить аудит записей активного списка при создании нового или изменении существующего активного списка.

Изменение активных списков

Вы можете изменить активный список через панель инструментов его карточки.

- Изменение доступно только для пользовательских активных списков.
- Схема активного списка выбирается только при создании активного списка. Изменение выбора схемы при редактировании активного списка недоступно.

Удаление активных списков

Вы можете удалить активный список через панель инструментов его карточки.

Удаление доступно только для пользовательских активных списков, не используемых в других сущностях системы. Если активный список не используется другими сущностями, в его поле **Ресурсы** отображается значение "0".

Вместе с активным списком также удаляются и все добавленные в него записи.

Поиск записей активного списка

Искать по активным спискам можно в разделе **Поиск**.

Чтобы просмотреть информацию о записи, нажмите на строку с ней в списке. В правой части экрана отобразится карточка записи.

Работа с записями активного списка

Доступные операции над записями активного списка:

- Добавление записей
- Просмотр записей
- Изменение записей
- Удаление записей
- Импорт записей
- Экспорт записей

Добавление записей

Чтобы добавить новую запись в активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит экран с подробной информацией об активном списке.
3. Нажмите на кнопку **Добавить запись** (+) над списком записей активного списка. Отобразится окно добавления записи.
4. Заполните значения полей ключа и полей данных записи.



- В поле типа `vrl_string` нельзя задавать операции, изменяющие исходное событие.
- Если запись содержит поле `ttl` типа `integer`, то в этом поле можно задавать время жизни записи в секундах. При этом:
  - Время жизни записи равняется меньшему из значений: `ttl` записи и TTL активного списка.
  - Чтобы сделать запись бессрочной, задайте полю значение `0`.

5. Нажмите на кнопку **Добавить**. Система добавит новую запись в активный список и отобразит соответствующее уведомление. Запись отобразится в перечне записей в карточке активного списка.



Новая запись может отобразиться в активном списке не сразу, а в течение минуты, когда произойдет синхронизация активных списков в интерфейсе системы и базе данных ClickHouse.

Добавление записей в черный и белый списки

Добавить запись в системный активный список можно аналогично тому, как это делается для пользовательского списка. Заполните следующие поля:

- `ruleId` — идентификатор правила, которое добавило запись в список.
- `id` — идентификатор записи в списке.
- `values` — условие на языке VRL, по которому проверяются события.



В условии нельзя задавать операции, изменяющие исходное событие.

- `owner` — владелец записи (`custom` или `vendor`).
- `ttl` — время жизни записи в секундах.



Чтобы запись оставалась в списке исключений бессрочно, укажите значение `0`.

- `createdBy` — идентификатор пользователя, создавшего запись.
- `createdAt` — дата добавления записи.



Добавлять записи в системные активные списки можно также в разделе **Поиск**.

Просмотр записей

Чтобы просмотреть запись в активном списке:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит экран с подробной информацией об активном списке.
3. Нажмите на строку записи в активном списке. Система отобразит в правой части рабочей области карточку этой записи.

В верхней части карточки отображается ключ записи. Справа от ключа записи расположены кнопки операций над записью активного списка.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

В карточке отображается информация о выбранной записи, сгруппированная по следующим секциям:

- **Поля записи** — сведения о полях ключа и полях данных записи. Поля ключа обозначены иконкой  .



Если запись содержит поле `ttl` типа `integer` со значением `0`, то для этого поля отображается надпись "Не ограничено".

- **Сведения** — общие сведения о записи:
  - **Повторы** — количество повторов записи.
  - **Срок действия** — дата и время прекращения действия записи.
  - **Дата создания** — дата и время создания записи.
  - **Дата изменения** — дата и время последнего изменения записи.

Изменение записей

Вы можете изменить запись в активном списке через кнопку в ее карточке.

- Изменение записей доступно только в пользовательских активных списках.
- Поля ключа записи задаются только при добавлении записи в активный список. Изменение значений полей ключа при редактировании записи недоступно.

Удаление записей

Вы можете удалить запись из активного списка через кнопку в ее карточке.

- Удаление записей доступно только в пользовательских активных списках.
- Удаленная запись может отображаться в активном списке в течение минуты после удаления, пока не произойдет синхронизация активных списков в интерфейсе системы и базе данных ClickHouse.

Импорт записей

Чтобы импортировать записи в активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит экран с подробной информацией об активном списке.
3. Нажмите на кнопку **Импорт** (⤓) над списком записей. Система отобразит окно импорта записей в активный список.
4. Нажмите **Выбрать файл** и выберите файл в формате CSV для импорта.
5. Нажмите на кнопку **Импортировать**. Система загрузит записи в активный список и отобразит соответствующее уведомление. Записи отобразятся в перечне записей в карточке активного списка.

Импортированные записи могут отобразиться с задержкой, поскольку операции вставки в активный список выполняются асинхронно и могут быть завершены после уведомления об успешном импорте.

При импорте файлов с большим количеством записей активного списка могут возникнуть ошибки. Вы можете ознакомиться с их решением в разделе Активные списки: ошибка при импорте записей.

Экспорт записей

Чтобы экспортировать записи из активного списка:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит экран с подробной информацией об активном списке.
3. Нажмите на кнопку **Экспорт** (⤒) над списком записей. Система запросит подтверждение выполнения экспорта.

4. Нажмите **Экспортировать**. Система загрузит записи активных списков на устройство пользователя в формате CSV и отобразит соответствующее уведомление.

Работа с интеграциями

Данный раздел описывает процесс настройки интеграции с другими системами. Настройка осуществляется в разделе **Ресурсы** → **Интеграции** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Интерфейс раздела

Панель инструментов содержит кнопку **Создать** (+), которая позволяет создать новую интеграцию.

В рабочей области отображается список настроенных интеграций. Для каждой интеграции в списке отображается следующая информация:

- Название интеграции.
- Тип интеграции.
- Адрес интеграции и порт.
- Даты создания и изменения интеграции.
- Текст последнего сообщения об ошибке.

В списке также присутствуют переключатели статуса для каждой интеграции.

При выборе конкретной интеграции в списке в правой части рабочей области отображается ее карточка с детальной информацией. Содержимое карточки зависит от типа интеграции. В карточке также присутствует переключатель статуса интеграции.

В верхнем правом углу карточки расположена кнопка действий ⋮ . При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранной интеграции: изменение и удаление.

При работе со списком интеграций доступны следующие операции:

- Поиск интеграций по полю **Название**.
- Фильтрация интеграций по полю **Тип**. Тип выбирается из выпадающего списка.

Работа с интеграцией

Доступные операции над интеграциями:

- Создание интеграции
- Просмотр интеграции
- Изменение интеграции
- Включение интеграции

- Удаление интеграции

Создание интеграции

Чтобы создать интеграцию:

1. Перейдите в раздел **Ресурсы** → **Интеграции**. Система отобразит сведения об имеющихся интеграциях.
2. Нажмите на кнопку **Создать** (+). Из выпадающего списка выберите тип создаваемой интеграции. Доступны следующие опции:
   - R-Vision SOAR;
   - SMTP.

Просмотр интеграции

Чтобы просмотреть интеграцию:

1. Перейдите в раздел **Ресурсы** → **Интеграции**. Система отобразит сведения об имеющихся интеграциях.
2. Нажмите на строку интеграции в списке. Система отобразит в правой части экрана карточку этой интеграции с подробной информацией о ней.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

В зависимости от типа интеграции, карточка может содержать следующие поля:

▼ *Поля интеграции SOAR*

- Идентификатор интеграции.
- Название.
- Описание.
- Идентификатор тенанта, к которому относится интеграция.
- Адрес сервера.
- API ключ.
- Организация.
- Категория.
- Конфигурация полей SOAR.
- Дата создания.
- Создал.
- Дата изменения.
- Изменил.

▼ *Поля почтовой интеграции*

- Идентификатор интеграции.

- Название.
- Описание.
- Идентификатор тенанта, к которому относится интеграция.
- Секрет.
- Отправитель.
- Адрес сервера.
- Версия сервера.
- Получатели.
- Дата создания.
- Создал.
- Дата изменения.
- Изменил.

В карточке интеграции находится кнопка **Отправить тестовое оповещение**, при нажатии на которую во внешнюю систему отправляется тестовый запрос для проверки подключения.

Изменение интеграции

Вы можете изменить интеграцию следующими способами:

- через меню действий в карточке интеграции;
- через изменение полей в карточке интеграции.

Включение интеграции

Включить или выключить интеграцию можно с помощью переключателя в левой части строки списка интеграций или из карточки интеграции.

Чтобы включить интеграцию:

1. Перейдите в раздел **Ресурсы** → **Интеграции**. Система отобразит сведения об имеющихся интеграциях.
2. Нажмите на строку интеграции в списке. Система отобразит в правой части экрана карточку этой интеграции с подробной информацией о ней.
3. Установите переключатель в активное положение. Система включит интеграцию и отобразит соответствующее уведомление. Начнется обмен данными согласно настройкам интеграции.

Удаление интеграции

Вы можете удалить интеграцию через меню действий в ее карточке.



Удаление недоступно для интеграций, используемых в других сущностях системы.

Интеграция с R-Vision SOAR

Настройки интеграции позволяют настроить систему на отправку инцидентов в R-Vision SOAR. Когда интеграция включена, создание оповещения в системе приводит к созданию инцидента в R-Vision SOAR.

Интеграция с R-Vision SOAR настраивается в обеих системах.

Настройка R-Vision SOAR

Выполните следующие шаги в R-Vision SOAR для настройки получения событий из системы:

1. Создание пользователя (опционально).
2. Создание токена для отправки запросов.
3. Настройка категории инцидента.

Создание пользователя

Чтобы создать пользователя, выполните следующие шаги в системе R-Vision SOAR:

1. Перейдите в раздел **Настройки системы** → **Общие** → **Пользователи системы** → **Пользователи**.
2. Выберите опцию **Добавить нового пользователя** в выпадающем меню кнопки **Добавить** (⊕). Система отобразит окно добавления пользователя.
3. Введите уникальный логин пользователя.
4. Укажите организации, от которых будут поступать инциденты.
5. Введите описание пользователя.
6. Установите пароль для пользователя.
7. В разделе **Системные роли** нажмите на кнопку **Добавить**, чтобы назначить пользователю соответствующие роли.

   Убедитесь, что у роли есть права на добавление и изменение инцидентов. Роли можно просмотреть в разделе **Настройки системы** → **Общие** → **Роли пользователей**.

8. Нажмите на кнопку **Добавить**, чтобы подтвердить создание пользователя.

Создание токена для отправки запросов

Для пользователя нужно создать токен, который будет использоваться для отправки запросов.

Чтобы создать токен, выполните следующие шаги в системе R-Vision SOAR:

1. Перейдите в раздел **Настройки системы** → **Общие** → **API**.
2. Нажмите на кнопку **Добавить** в заголовке списка токенов. Система отобразит окно выбора пользователя.
3. Выберите пользователя, для которого будет создан токен.
4. Нажмите на кнопку **Выбрать**, чтобы подтвердить выбор пользователя. Токен пользователя будет отображен в списке.

Настройка категории инцидента

Категория инцидента в системе R-Vision SOAR представляет собой набор полей, используемых для описания инцидента. В рамках категории существуют различные типы инцидентов.

Категории позволяют разграничивать инциденты по смыслу. Для разных категорий можно настроить разные поля и разные сценарии реагирования. Например, можно настроить категорию таким образом, что все ее инциденты будут вызывать автоматическую блокировку учетной записи пользователя или изоляцию его хоста в целях обеспечения безопасности.

Для возможности настройки синхронизации с R-Vision SOAR требуется настроить поля категории инцидента.

Чтобы настроить категории инцидента, выполните следующие шаги в системе R-Vision SOAR:

1. Перейдите в раздел **Настройки системы** → **Инциденты** → **Категории**.
2. Выберите требуемую категорию нажатием на ее строку в списке или создайте новую с помощью кнопки **Добавить** (⊕).
3. В секции **Поля** нажмите на кнопку **Изменить**. Система отобразит окно выбора полей.
4. Выберите поля, требуемые для настраиваемой интеграции, и запомните теги этих полей.

   При необходимости нужные поля можно предварительно создать в разделе **Настройки системы** → **Инциденты** → **Поля**.

Настройка системы

Выполните следующие действия в системе для настройки отправки событий в R-Vision SOAR:

1. Перейдите в раздел **Ресурсы** → **Секреты** и создайте секрет типа **Токен**, поместив в него API-ключ (токен пользователя) для подключения к R-Vision SOAR.
2. Перейдите в раздел **Ресурсы** → **Интеграции**. Система отобразит сведения об имеющихся интеграциях.
3. Нажмите на кнопку **Создать** (＋) и выберите **R-Vision SOAR** из выпадающего меню. Система отобразит окно создания интеграции.
4. Заполните поля:
   a. Название.
   b. Описание (опционально).
   c. URL-адрес сервера R-Vision SOAR, например `https://rvision_soar.local`.
5. Выберите секрет типа **Токен**, созданный на шаге 1.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

6. Нажмите на кнопку **Проверить подключение**. Система проверит доступность подключения к системе и отобразит его статус: **Успешное соединение** или **Ошибка подключения**.
7. Выберите организацию из выпадающего списка **Организация** (если в системе существует несколько организаций).

На данном этапе вы можете отправить тестовое оповещение в систему R-Vision SOAR с помощью кнопки **Отправить тестовое оповещение**.

8. Выберите шаблон, по которому будут настраиваться инциденты в R-Vision SOAR.
9. Задайте название категории инцидента из системы R-Vision SOAR.

   Название категории можно также задать с помощью шаблона. Подробности использования шаблонов приведены в разделе Работа с шаблонами полей.

10. При необходимости укажите тип инцидента из системы R-Vision SOAR.
11. Нажмите на кнопку **Создать**. Система создаст интеграцию и отобразит соответствующее уведомление. Новая интеграция появится в списке раздела **Ресурсы** → **Интеграции**.

При отправке оповещений в R-Vision SOAR предусмотрены 3 попытки с интервалом в 100 мс, прежде чем отправка завершится с ошибкой.

Соответствие атрибутов и полей инцидента

При настройке интеграции важно понимать правила соответствия между атрибутами системы и полями инцидентов, отправленных в R-Vision SOAR.

Чтобы задать соответствие полей инцидента из R-Vision SOAR и атрибутов в системе, укажите следующие настройки поля в шаблоне сообщения:

1. **Тег поля SOAR** — тег поля инцидента из системы R-Vision SOAR.
2. **Тип поля** — тип поля инцидента из системы R-Vision SOAR. Позволяет передать в R-Vision SOAR поле нужного типа. Доступные типы поля:
   - **Строка**;
   - **Число**;
   - **Логический тип**;
   - **JSON**.
3. **Значение поля** — значение атрибута системы.

Доступны следующие варианты заполнения значений атрибутов в системе:

- **Константы** — значения, которые всегда передаются в поле инцидента в статичном виде, без изменений. Например: "Системное оповещение".
- **Шаблоны** — значения, которые будут передаваться в поле инцидента на основе шаблона. Подробности использования шаблонов приведены в разделе Работа с шаблонами полей.

Пример заполнения полей

Рассмотрим пример заполнения полей с помощью констант.

- **Тип**: Подозрение на инцидент (событие ИБ)
- **Категория**: Системный инцидент

Пример соответствия между атрибутами и полями инцидента в системах

Имя поля системы	Тег поля SOAR	Имя поля SOAR (RU)	Имя поля SOAR (ENG)
alertId	SystemAlertId	System Alert ID	System Alert ID
statusName	STATUS	Статус инцидента	Incident status
severityLevel	LEVEL	Уровень инцидента	Incident level
correlationEventsLink	correlationEvents	Ссылка на корреляционные события	Correlation events link
correlationEventsCount	correlationEventsCount	Количество корреляционных событий	Correlation events count
correlationRuleName	sense_rule_name	Название правила	Rule name

Так как интеграция настроена на основе констант, то поля инцидентов, отправляемых по этой интеграции в R-Vision SOAR, всегда будут хранить одни и те же указанные данные:

- `SystemAlertId`: `alertId`.
- `STATUS`: `statusName`.
- `LEVEL`: `severityLevel`.
- `correlationEvents`: `correlationEventsLink`.
- `correlationEventsCount`: `correlationEventsCount`.
- `sense_rule_name`: `correlationRuleName`.

Работа с шаблонами инцидентов

Для создания инцидентов в R-Vision SOAR по интеграции необходимо использовать шаблоны. Шаблоны позволяют задавать тег, тип и значение для полей инцидента.

По умолчанию в списке шаблонов отображается стандартный шаблон для инцидентов SOAR. Стандартный шаблон недоступен для изменения или удаления.

▼ *Стандартный шаблон для оповещений SOAR-интеграции*

Поле 1:

- Тег поля SOAR: `LEVEL`
- Тип поля: `string`
- Значение поля:

```
{{#switch alert.severity }}
    {{#case 'SEVERITY_LOW'}} Незначительный {{/case}}
    {{#case 'SEVERITY_MEDIUM'}} Средний {{/case}}
    {{#case 'SEVERITY_HIGH'}} Высокий {{/case}}
    {{#case 'SEVERITY_CRITICAL'}} Критичный {{/case}}
    {{#default 'Средний' }} {{/default}}
{{/switch}}
```

Поле 2:

- Тег поля SOAR: `STATUS`
- Тип поля: `string`
- Значение поля:

```
{{#switch alert.status }}
    {{#case 'STATUS_OPEN'}} Создан {{/case}}
    {{#case 'STATUS_IN_PROGRESS'}} Создан {{/case}}
    {{#case 'STATUS_RESOLVED'}} Закрыт {{/case}}
    {{#case 'STATUS_FALSE_POSITIVE'}} Закрыт {{/case}}
    {{#default 'Создан' }} {{/default}}
{{/switch}}
```

Поле 3:

- Тег поля SOAR: `AlertLink`
- Тип поля: `string`
- Значение поля:

```
{{alert.link}}
```

Поле 4:

- Тег поля SOAR: `DESCRIPTION`
- Тип поля: `string`
- Значение поля:

```
{{{alert.correlationRuleDescription}}}
```

Вы можете добавлять, изменять и удалять шаблоны инцидентов, а также создавать новые шаблоны на основе существующих.

Добавление шаблона инцидента

Чтобы добавить новый шаблон инцидента:

1. Нажмите на кнопку **Добавить шаблон** в окне настройки интеграции с R-Vision SOAR. Отобразится окно добавления шаблона сообщения.
2. Укажите название шаблона.
3. Задайте настройки для каждого поля:
4. **Тег поля SOAR** — укажите тег поля инцидента из системы R-Vision SOAR. Тег может содержать латинские буквы, цифры, подчеркивания. Первый символ должен быть буквой. Длина тега — от 2 до 60 символов.

5. **Тип поля** — выберите из выпадающего списка тип поля инцидента из системы R-Vision SOAR. Позволяет передать в R-Vision SOAR поле нужного типа. Доступные типы поля:
   - **Строка**;
   - **Число**;
   - **Логический тип**;
   - **JSON**.
6. **Значение поля** — укажите значение атрибута из системы.

   В качестве значения можно передавать поля оповещений и корреляционных событий с помощью специальных шаблонов полей. Подробности использования шаблонов приведены в разделе Работа с шаблонами полей.

   Для удобства добавления шаблонов на основе оповещения вы можете воспользоваться функционалом справочника. При нажатии на кнопку **Справочник** в области настроек поля SOAR система отобразит список стандартных полей оповещения. Для каждого поля в списке отображаются его название и тип данных. Чтобы добавить поле оповещения в качестве шаблона, нажмите на него в списке справочника.

   Пример

   При нажатии на поле **Статус оповещения** в справочнике, **Значение поля** заполнится шаблоном `{{alert.status}}`.

   Чтобы добавить новое поле, нажмите на кнопку **Добавить поле**. В окне отобразятся настройки нового поля. Чтобы удалить поле, нажмите на кнопку в правой верхней части его настроек.

7. Нажмите на кнопку **Добавить**. Система создаст шаблон инцидента и отобразит соответствующее уведомление. Новый шаблон появится в выпадающем списке шаблонов.

Изменение шаблона инцидента

Чтобы изменить шаблон инцидента:

1. Откройте выпадающий список **Шаблон** в окне настройки интеграции с R-Vision SOAR.
2. Нажмите на кнопку в строке нужного шаблона. Отобразится окно изменения шаблона сообщения.
3. Внесите требуемые изменения в шаблон инцидента.
4. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в выпадающем списке шаблонов.

Создание шаблона инцидента на основе существующего шаблона

Чтобы добавить новый шаблон инцидента на основе существующего:

1. Откройте выпадающий список **Шаблоны** в окне настройки интеграции с R-Vision SOAR.
2. Нажмите на кнопку + в строке нужного шаблона. Отобразится окно добавления шаблона сообщения, поля которого будут автоматически заполнены данными из выбранного шаблона.
3. Измените название шаблона инцидента.

4. При необходимости измените настройки полей в шаблоне.
5. Нажмите на кнопку **Добавить**. Система создаст шаблон инцидента и отобразит соответствующее уведомление. Новый шаблон появится в выпадающем списке шаблонов.

Удаление шаблона инцидента

Чтобы удалить шаблон инцидента:

1. Откройте выпадающий список **Шаблоны** в окне настройки интеграции с R-Vision SOAR.
2. Нажмите на кнопку в строке нужного шаблона. Отобразится окно подтверждения удаления шаблона.
3. Нажмите на кнопку **Удалить**. Система удалит шаблон инцидента и отобразит соответствующее уведомление. Удаленный шаблон будет исключен из выпадающего списка шаблонов.

Работа с шаблонами полей

В качестве значений полей и названий категорий для инцидентов R-Vision SOAR можно передавать поля оповещений или корреляционных событий, сгенерировавших оповещения. Для этого поля необходимо указать в виде шаблона Handlebars:

- `{{...}}` — для передачи полей с экранированием специальных символов.
- `{{{...}}}` — для передачи полей без экранирования специальных символов.

При построении шаблонов для поля типа **JSON** имеются следующие особенности:

- `{{...}}` — передача JSON-значения как текстовой строки без интерпретации шаблонов.
- `{{{...}}}` — передача JSON-значения как JSON-строки без экранирования символов.

Для корректной передачи JSON-значений необходимо с помощью шаблона Handlebars сформировать валидный JSON-объект или JSON-массив. Например, для этого можно использовать встроенные утилиты (хелперы) преобразования JSON-объекта в JSON-строку.

Примеры ввода значений для поля типа **JSON**:

- Для передачи одного объекта:

```
{{{json <object>}}}
```

- Для передачи нескольких объектов:

```
[{{{json <object_1>}}}, ..., {{{json <object_N>}}}]
```

- Для передачи массива объектов:

```
[{{{json <object>[0]}}}, ..., {{{json <object>[N]}}}]
```

- Для преобразования неформатированной JSON-строки и передачи ее в форматированном виде:

```
{{{json <object> true}}}
```

Здесь:

- `<object>`, `<object_1>`, ..., `<object_N>` — имена JSON-объектов.

Шаблоны могут принимать данные из:

- Оповещений. Для этого используйте шаблон вида: `{{alert.<поле оповещения>}}`.

  Пример шаблона на основе оповещения

  ```
  {{alert.link}}
  ```

  В примере выше шаблон получает данные из поля `link` оповещения. Например, если поле `link` содержит значение "https://1.0.0.0/alert/id", то такое значение и будет передаваться в поле инцидента через этот шаблон.

  Доступные шаблоны системных полей оповещений представлены в таблице ниже.

  ▼ *Доступные шаблоны системных полей оповещений*

Поле оповещения	Тип данных	Шаблон
ID	Строка	`{{alert.id}}`
Название	Строка	`{{alert.name}}`
ID тенанта	Строка	`{{alert.tenantId}}`
Название правила сегментации	Строка	`{{alert.segmentation.ruleName}}`
ID правила сегментации	Строка	`{{alert.segmentation.ruleId}}`
Версия правила сегментации	Строка	`{{alert.segmentation.ruleVersion}}`
Поля группировки	Строка	`{{alert.segmentation.groupBy}}`
Название правила корреляции	Строка	`{{alert.correlationRuleName}}`
ID правила корреляции	Строка	`{{alert.correlationRuleId}}`
Описание правила корреляции	Строка	`{{alert.correlationRuleDescription}}`
Уровень угрозы	Строка	`{{alert.severity}}`
Теги	Строка	`{{alert.tags}}`
Статус	Строка	`{{alert.status}}`
Количество событий	Целое число	`{{alert.correlationEventsCount}}`

Поле оповещения	Тип данных	Шаблон
Дата создания	Строка	`{{alert.createdAt}}`
Дата изменения	Строка	`{{alert.updatedAt}}`
Время первого корреляционного события	Строка	`{{alert.firstEventTs}}`
Время последнего корреляционного события	Строка	`{{alert.lastEventTs}}`
Идентификаторы корреляционных событий	Массив строк	`{{alert.correlationEventsIds}}`
Корреляционные события	Массив объектов	`{{alert.correlationEvents}}`
Идентификатор элемента конвейера, который сгенерировал оповещение. Префикс — идентификатор конвейера	Строка	`{{alert.eventSourceTransformId}}`
Идентификатор хранилища событий	Строка	`{{alert.eventStorageId}}`



Для удобства добавления шаблонов на основе оповещения вы можете воспользоваться функционалом справочника.

- Корреляционных событий, сгенерировавших оповещение. Для этого используйте шаблон вида: `{{correlationEvent.<поле события>}}`.



Данные будут браться только из первого корреляционного события, сгенерировавшего оповещение. Все последующие события, поступающие на то же оповещение, учитываться не будут.

Пример шаблона на основе корреляционного события

```
{{correlationEvent.name}}
```

В примере выше шаблон получает данные из названия корреляционного события. Например, если в качестве названия используется "UsualCorrelationEvent", то такое значение и будет передаваться в поле инцидента через этот шаблон.



Если требуются данные из базового события, то предварительно перенесите соответствующие поля события в блок `on_correlate` правила корреляции.

Построение сложных шаблонов

В системе R-Vision SOAR можно создавать и настраивать пользовательские поля инцидента. По этой причине может возникнуть необходимость гибкой настройки шаблона, обрабатывающего различные значения поля. Для этого, помимо простых шаблонов, в системе можно использовать шаблоны с конструкцией ветвления `switch`.

Общая структура switch-конструкции

```
{{#switch <поле> }}
    {{#case '<значение 1>'}} <результат 1> {{/case}}
    {{#case '<значение 2>'}} <результат 2> {{/case}}
    ...
    {{#default '<значение по умолчанию>' }} <результат по умолчанию> {{/default}}
{{/switch}}
```

Конструкция работает следующим образом:

1. Проверяется поле `<поле>` оповещения или корреляционного события.
2. Содержимое поля сравнивается с заданным списком значений: `<значение 1>`, `<значение 2>`, ..., `<значение по умолчанию>`.
3. В зависимости от значения поля, соответствующее ему поле инцидента, передаваемого в R-Vision SOAR, заполняется требуемым содержимым: `<результат 1>`, `<результат 2>`, ..., `<результат по умолчанию>`.

Рассмотрим пример switch-конструкции, работающей на основе оповещения.

Пример switch-конструкции

```
{{#switch alert.status }}
    {{#case 'STATUS_OPEN'}} Создан {{/case}}
    {{#case 'STATUS_IN_PROGRESS'}} Создан {{/case}}
    {{#case 'STATUS_RESOLVED'}} Закрыт {{/case}}
    {{#case 'STATUS_FALSE_POSITIVE'}} Закрыт {{/case}}
    {{#default 'Создан' }} {{/default}}
{{/switch}}
```

Пример выше проверяет поле `status` оповещения, а затем заполняет соответствующее поле инцидента значением, отвечающим содержимому поля `status`. Например, если поступившее оповещение имеет статус `STATUS_OPEN`, то поле инцидента заполнится значением "Создан".

Почтовые интеграции

В системе оповещения генерируются при обнаружении последовательности событий, соответствующих заданному правилу корреляции. Вы можете настроить отправку оповещений по электронной почте, указав при создании задачи адреса получателей уведомления.

Интеграция с SMTP

Чтобы настроить интеграцию с почтовым сервером SMTP:

1. Если для подключения к почтовому серверу нужна авторизация, перейдите в раздел **Ресурсы** → **Секреты** и создайте секрет типа **Учетные данные**, поместив в него логин и пароль для авторизации в почтовом сервисе.
2. Перейдите в раздел **Ресурсы** → **Интеграции**. Система отобразит сведения об имеющихся интеграциях.

3. Нажмите на кнопку **Создать** (+) и выберите **SMTP** из выпадающего меню. Система отобразит окно создания интеграции.
4. Заполните поля:
   a. Название.
   b. Описание (опционально).
   c. Отправитель.
   d. Адрес сервера.
   e. Порт.
   f. Получатели: введите электронные адреса получателей через запятую.
5. Если для подключения к почтовому серверу нужна авторизация:
   a. Установите переключатель **Требуется авторизация** в активное положение. Отобразится поле выбора секрета.
   b. Выберите секрет типа **Учетные данные**, созданный на шаге 1.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.
6. Выберите шаблон сообщения, которое будет отправляться на почтовый сервер.
7. Нажмите на кнопку **Проверить подключение**. Система проверит доступность подключения к серверу и отобразит его статус: **Успешное соединение** или **Ошибка подключения**.



На данном этапе вы можете отправить тестовое оповещение на почтовый сервер с помощью кнопки **Отправить тестовое оповещение**. Отправка тестового оповещения доступна только после выбора шаблона и успешной проверки подключения.

8. Нажмите на кнопку **Создать**. Система создаст интеграцию и отобразит соответствующее уведомление. Новая интеграция появится в списке раздела **Ресурсы** → **Интеграции**.



При отправке оповещений по электронной почте предусмотрены 3 попытки с интервалом в 100 мс, прежде чем отправка завершится с ошибкой.

Работа с шаблонами сообщения

Для отправки сообщений по SMTP-интеграции необходимо использовать шаблоны. Шаблоны позволяют задавать тему и текст отправляемых сообщений.

По умолчанию в списке шаблонов отображаются стандартные шаблоны. Стандартные шаблоны недоступны для изменения или удаления.

▼ *Стандартный шаблон для оповещений SMTP-интеграции*

Тема сообщения:

```
Сработало правило корреляции {{alert.correlationRuleName}}
```

Текст сообщения:

```
<p><b>Оповещение по правилу корреляции</b></p>
<p>Ссылка на оповещение: {{alert.link}}</p>
<p>System Alert ID: {{alert.id}}</p>
<p>Название правила: {{alert.name}}</p>
<p>Дата первого события: {{alert.firstEventTs}}</p>
<p>Дата последнего события: {{alert.lastEventTs}}</p>
<p>Уровень угрозы: {{#switch alert.severity }}
        {{#case 'SEVERITY_LOW'}} Низкий {{/case}}
        {{#case 'SEVERITY_MEDIUM'}} Средний {{/case}}
        {{#case 'SEVERITY_HIGH'}} Высокий {{/case}}
        {{#case 'SEVERITY_CRITICAL'}} Критический {{/case}}
        {{#default 'Средний' }} {{/default}}
    {{/switch}}</p>
<p>Статус: {{#switch alert.status }}
        {{#case 'STATUS_OPEN'}} Новое {{/case}}
        {{#case 'STATUS_IN_PROGRESS'}} В работе {{/case}}
        {{#case 'STATUS_RESOLVED'}} Закрыто {{/case}}
        {{#case 'STATUS_FALSE_POSITIVE'}} Ложное срабатывание {{/case}}
        {{#default 'Новое' }} {{/default}}
    {{/switch}}</p>
<p>Количество корреляционных событий: {{alert.correlationEventsCount}}</p>
{{#if correlationEvent}}
<p><b>Информация по корреляционному событию</b></p>
{{#each correlationEvent}}
<p>{{@key}}: {{this}}</p>
{{/each }}
{{/if}}
```

▼ *Стандартный шаблон для политики аудита по точкам входа*

Тема сообщения:

```
Сработала политика аудита источников {{monitoringPolicy.name}}
```

Текст сообщения:

```
<p>Перестали поступать события с точки входа <b> {{element.name}} (ID
{{element.id}}) </b></p>
<p>Коллектор ID: {{collector.id}}</p>
<p>Название коллектора: {{collector.name}}</p>
<p>Конвейер ID: {{eventSource.id}}</p>
<p>Название конвейера: {{eventSource.name}}</p>
<p><a href="{{eventSource.link}}"> Ссылка на конвейер </a></p>
<p>Параметры политики: порог {{monitoringPolicy.threshold}} и интервал
{{monitoringPolicy.interval.value}}{{#switch monitoringPolicy.interval.unit
}}{{#case 'UNIT_SECOND'}} Секунда {{/case}}{{#case 'UNIT_MINUTE'}} Минута
{{/case}}{{#case 'UNIT_HOUR'}} Час {{/case}}{{#case 'UNIT_DAY'}} День
```

```
{{/case}}{{#case 'UNIT_WEEK'}} Неделя {{/case}}{{#case 'UNIT_MONTH'}} Месяц
{{/case}}{{#case 'UNIT_QUARTER'}} Квартал {{/case}}{{#case 'UNIT_YEAR'}} Год
{{/case}}{{/switch}} </p>
```

▼ *Стандартный шаблон для политики аудита источников по маске*

Тема сообщения:

```
Сработала политика аудита источников по маске {{monitoringPolicy.name}}
```

Текст сообщения:

```
<p>Перестали поступать события с источника <b> {{source.name}} </b></p>

<p>Параметры политики: порог {{monitoringPolicy.threshold}} и интервал
{{monitoringPolicy.interval.value}}{{#switch monitoringPolicy.interval.unit
}}{{#case 'UNIT_SECOND'}} Секунда {{/case}}{{#case 'UNIT_MINUTE'}} Минута
{{/case}}{{#case 'UNIT_HOUR'}} Час {{/case}}{{#case 'UNIT_DAY'}} День
{{/case}}{{#case 'UNIT_WEEK'}} Неделя {{/case}}{{#case 'UNIT_MONTH'}} Месяц
{{/case}}{{#case 'UNIT_QUARTER'}} Квартал {{/case}}{{#case 'UNIT_YEAR'}} Год
{{/case}}{{/switch}} </p>
```

Вы можете добавлять, изменять и удалять шаблоны сообщений, а также создавать новые шаблоны на основе существующих.

Добавление шаблона сообщения

Чтобы добавить новый шаблон сообщения:

1. Нажмите на кнопку **Добавить шаблон** в окне настройки интеграции с SMTP. Отобразится окно добавления шаблона сообщения.
2. Укажите название шаблона, а также тему и текст сообщения.

   В теме и тексте сообщения можно передавать поля оповещений и корреляционных событий с помощью специальных шаблонов полей.

3. Нажмите на кнопку **Добавить**. Система создаст шаблон сообщения и отобразит соответствующее уведомление. Новый шаблон появится в списке.

Изменение шаблона сообщения

Чтобы изменить шаблон сообщения:

1. Откройте выпадающий список **Шаблон** в окне настройки интеграции с SMTP.
2. Нажмите на кнопку ✎ в строке нужного шаблона. Отобразится окно изменения шаблона сообщения.
3. Внесите требуемые изменения в шаблон сообщения.
4. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее

уведомление. Обновленная информация отобразится в выпадающем списке шаблонов, а также в окне настройки интеграции, если выбран данный шаблон.

Создание шаблона сообщения на основе существующего шаблона

Чтобы добавить новый шаблон сообщения на основе существующего:

1. Откройте выпадающий список **Шаблон** в окне настройки интеграции с SMTP.
2. Нажмите на кнопку + в строке нужного шаблона. Отобразится окно добавления шаблона сообщения, поля которого будут автоматически заполнены данными из выбранного шаблона.
3. Измените название шаблона сообщения.
4. При необходимости измените тему и текст сообщения.
5. Нажмите на кнопку **Добавить**. Система создаст шаблон сообщения и отобразит соответствующее уведомление. Новый шаблон появится в выпадающем списке шаблонов.

Удаление шаблона сообщения

Чтобы удалить шаблон сообщения:

1. Откройте выпадающий список **Шаблон** в окне настройки интеграции с SMTP.
2. Нажмите на кнопку в строке нужного шаблона. Отобразится окно подтверждения удаления шаблона.
3. Нажмите на кнопку **Удалить**. Система удалит шаблон сообщения и отобразит соответствующее уведомление. Удаленный шаблон будет исключен из выпадающего списка шаблонов.

Работа с шаблонами полей

В тексте и теме сообщения SMTP-интеграции можно передавать поля оповещений или корреляционных событий, сгенерировавших оповещения. Для этого поля необходимо указать в двойных фигурных скобках: `{{...}}`.



Данные для корреляционных событий берутся только из первого корреляционного события, сгенерировавшего оповещение. Все последующие события, поступающие на то же оповещение, не учитываются.

Пример шаблона с полем оповещения

```
{{alert.link}}
```

В примере выше шаблон получает данные из поля `link` оповещения. Например, если поле `link` содержит значение "https://1.0.0.0/alert/id", то такое значение и будет передаваться в сообщении с этим шаблоном.

Доступные шаблоны системных полей оповещений представлены в таблице ниже.

▼ *Доступные шаблоны системных полей оповещений*

Поле оповещения	Тип данных	Шаблон
ID	Строка	`{{alert.id}}`
Название	Строка	`{{alert.name}}`
ID тенанта	Строка	`{{alert.tenantId}}`
Название правила сегментации	Строка	`{{alert.segmentation.ruleName}}`
ID правила сегментации	Строка	`{{alert.segmentation.ruleId}}`
Версия правила сегментации	Строка	`{{alert.segmentation.ruleVersion}}`
Поля группировки	Строка	`{{alert.segmentation.groupBy}}`
Название правила корреляции	Строка	`{{alert.correlationRuleName}}`
ID правила корреляции	Строка	`{{alert.correlationRuleId}}`
Описание правила корреляции	Строка	`{{alert.correlationRuleDescription}}`
Уровень угрозы	Строка	`{{alert.severity}}`
Теги	Строка	`{{alert.tags}}`
Статус	Строка	`{{alert.status}}`
Количество событий	Целое число	`{{alert.correlationEventsCount}}`
Дата создания	Строка	`{{alert.createdAt}}`
Дата изменения	Строка	`{{alert.updatedAt}}`
Время первого корреляционного события	Строка	`{{alert.firstEventTs}}`
Время последнего корреляционного события	Строка	`{{alert.lastEventTs}}`
Идентификаторы корреляционных событий	Массив строк	`{{alert.correlationEventsIds}}`
Корреляционные события	Массив объектов	`{{alert.correlationEvents}}`
Идентификатор элемента конвейера, который сгенерировал оповещение. Префикс — идентификатор конвейера	Строка	`{{alert.eventSourceTransformId}}`
Идентификатор хранилища событий	Строка	`{{alert.eventStorageId}}`

Пример шаблона с полем корреляционного события

```
{{correlationEvent.name}}
```

В примере выше шаблон получает данные из названия корреляционного события. Например, если в качестве названия используется "UsualCorrelationEvent", то такое значение и будет передаваться в сообщении с этим шаблоном.



Если требуются данные из базового события, то необходимо предварительно перенести соответствующие поля события в блок `on_correlate` правила

корреляции.

Построение сложных шаблонов полей

При отправке сообщений по SMTP-интеграции может возникнуть необходимость гибкой настройки шаблона, обрабатывающего различные значения одного и того же поля. Для этого, помимо простых шаблонов полей, в системе можно использовать шаблоны с конструкцией ветвления `switch`.

Общая структура switch-конструкции

```
{{#switch <поле> }}
    {{#case '<значение 1>'}} <результат 1> {{/case}}
    {{#case '<значение 2>'}} <результат 2> {{/case}}
    ...
    {{#default '<значение по умолчанию>' }} <результат по умолчанию> {{/default}}
{{/switch}}
```

Конструкция работает следующим образом:

1. Проверяется поле `<поле>` оповещения или корреляционного события.
2. Содержимое поля сравнивается с заданным списком значений: `<значение 1>`, `<значение 2>`, ..., `<значение по умолчанию>`.
3. В зависимости от значения поля, в сообщении передается одно из следующих значений: `<результат 1>`, `<результат 2>`, ..., `<результат по умолчанию>`.

Рассмотрим пример switch-конструкции, работающей на основе оповещения.

Пример switch-конструкции

```
{{#switch alert.status }}
    {{#case 'STATUS_OPEN'}} Создан {{/case}}
    {{#case 'STATUS_IN_PROGRESS'}} Создан {{/case}}
    {{#case 'STATUS_RESOLVED'}} Закрыт {{/case}}
    {{#case 'STATUS_FALSE_POSITIVE'}} Закрыт {{/case}}
    {{#default 'Создан' }} {{/default}}
{{/switch}}
```

Пример выше проверяет поле `status` оповещения, а затем передает в сообщении значение, отвечающее содержимому поля `status`. Например, если поступившее оповещение имеет статус `STATUS_OPEN`, то в сообщении будет передано значение "Создан".

Работа с драйверами БД

Данный раздел описывает процесс работы с драйверами баз данных в системе. Работа осуществляется в разделе **Ресурсы** → **Драйверы БД** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со

статусом **Суперадминистратор**.

О драйверах баз данных

Драйвер БД — это компонент системы, который используется для подключения внешних баз данных и дальнейшей работы с ними. Такие БД можно использовать для сбора данных с помощью точек входа в конфигурации конвейера и последующей их обработки.

В системе доступны драйверы БД следующих типов:

- **Системные драйверы БД** — создаются системой автоматически. Их нельзя изменять или удалять.
- **Пользовательские драйверы БД** — создаются вручную.

Существуют системные драйверы для следующих баз данных:

- ClickHouse;
- PostgreSQL;
- MS SQL;
- MySQL.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (＋) позволяет создать новый драйвер БД.
- Поле **Поиск** предназначено для быстрого поиска драйверов БД в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся драйверов БД. Таблица содержит следующие столбцы:

- **Название** — имя драйвера БД, используемое для его идентификации в системе.
- **Тип** — тип драйвера БД: системный (предустановленный по умолчанию) или пользовательский.
- **Имя файла** — имя файла драйвера БД.
- **Версия** — версия драйвера БД.
- **Описание** — краткое описание функциональности драйвера БД.
- **Дата создания** — дата и время создания драйвера БД.
- **Дата изменения** — дата и время последнего изменения драйвера БД.

При работе с таблицей драйверов БД доступны следующие операции:

- Поиск драйверов по полю **Название**.
- Фильтрация драйверов по полю **Тип**. Тип выбирается из выпадающего списка.
- Сортировка драйверов по полям **Название**, **Версия**, **Дата создания** и **Дата изменения**.

- Настройка отображения таблицы.

При выборе конкретного драйвера БД в правой части рабочей области отображается его карточка с детальной информацией.

Работа с драйвером БД

Доступные операции над драйверами БД:

- Создание драйвера БД
- Просмотр драйвера БД
- Изменение драйвера БД
- Удаление драйвера БД

Создание драйвера БД

Чтобы создать драйвер БД:

1. Перейдите в раздел **Ресурсы** → **Драйверы БД**. Система отобразит сведения об имеющихся драйверах БД.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания драйвера БД.
3. Введите название драйвера подключаемой базы данных. Например: База данных SQLite.
4. При необходимости введите описание драйвера БД.
5. Укажите версию драйвера БД.
6. Нажмите на кнопку **Выбрать файл** и выберите файл драйвера подключаемой базы данных. Допустимый формат файла драйвера — JAR.



- По умолчанию система позволяет загружать файлы драйверов размером не более 100 000 000 байт (приблизительно 95,37 МБ). При необходимости изменить максимальный допустимый размер файла драйвера обратитесь к разделу Настройка ограничений размера файла драйвера БД.
- Если загрузить файл драйвера не удается, обратитесь к разделу Точка входа типа Database: загрузка драйвера прекращается по таймауту.

7. Нажмите на кнопку **Создать**. Система создаст драйвер БД и отобразит соответствующее уведомление. Новый драйвер появится в списке раздела **Ресурсы** → **Драйверы БД**.

Просмотр драйвера БД

Чтобы просмотреть драйвер БД:

1. Перейдите в раздел **Ресурсы** → **Драйверы БД**. Система отобразит сведения об имеющихся драйверах БД.
2. Нажмите на строку драйвера БД в списке. Система отобразит в правой части экрана карточку этого драйвера с подробной информацией о нем.

В верхней части карточки отображается название драйвера БД.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка отображает общие сведения о драйвере БД и содержит следующие поля:

- **ID** — уникальный идентификационный код драйвера БД, генерируемый автоматически при его создании.
- **Название** — имя драйвера БД, используемое для его идентификации в системе.
- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится драйвер БД.
- **Тип** — тип драйвера БД: системный (предустановленный по умолчанию) или пользовательский.
- **Описание** — краткое описание функциональности драйвера БД.
- **Имя файла** — имя файла драйвера БД.



Файл драйвера БД можно скачать из объектного хранилища NATS Object Store.

- **Версия** — версия драйвера БД.
- **Дата создания** — дата и время создания драйвера БД.
- **Создал** — пользователь, инициировавший создание драйвера БД.
- **Дата изменения** — дата и время последнего изменения драйвера БД.
- **Изменил** — пользователь, выполнивший последнее изменение драйвера БД.

Изменение драйвера БД

Вы можете изменить драйвер БД следующими способами:

- через меню действий в карточке драйвера;
- через изменение полей в карточке драйвера.



Изменение доступно только для драйверов БД с типом **Пользовательский**.

Удаление драйвера БД

Вы можете удалить драйвер БД через меню действий в его карточке.



Удаление доступно только для драйверов БД с типом **Пользовательский**, не используемых на конвейерах.

Работа с секретами

Данный раздел описывает процесс работы с секретами в системе. Работа осуществляется в разделе **Ресурсы** → **Секреты** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О секретах

Секрет — это компонент системы, который обеспечивает безопасное хранение чувствительных данных, таких как учетные данные, токены, строки подключения, сертификаты, приватные ключи. Секреты позволяют ссылаться на эти данные и использовать их без явного указания в различных объектах системы.

Секреты можно использовать в конфигурации конвейера для подключения к источникам данных в точках входа и/или внешним системам в конечных точках, а также для подключения к внешним системам при настройке интеграций.



Конвейеры по умолчанию используют те данные секрета, с которыми была установлена их конфигурация. Если в параметры секрета были внесены изменения, то для их принятия на том или ином конвейере необходимо переустановить его конфигурацию.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый секрет.
- Поле **Поиск** предназначено для быстрого поиска секретов в отображаемом списке по названию.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка секретов.

В рабочей области отображается таблица имеющихся секретов. Таблица содержит следующие столбцы:

- **Название** — имя секрета, используемое для его идентификации в системе.
- **Описание** — краткое описание функциональности секрета.
- **Тип** — тип информации, хранящейся в секрете.
- **Статус** — текущий статус секрета: **Включен** или **Выключен**.
- **Дата создания** — дата и время создания секрета.
- **Дата изменения** — дата и время последнего изменения секрета.

При работе с таблицей секретов доступны следующие операции:

- Поиск секретов по полю **Название**.
- Фильтрация секретов по полям **Статус** и **Тип**. Статус и тип выбираются из выпадающих списков.
- Сортировка секретов по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного секрета в правой части рабочей области отображается его карточка с детальной информацией.

Работа с секретом

Доступные операции над секретами:

- Создание секрета
- Просмотр секрета
- Включение секрета
- Изменение секрета
- Удаление секрета

Создание секрета

Чтобы создать секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания секрета.
3. Введите название секрета.
4. При необходимости введите описание секрета.
5. Выберите из выпадающего списка **Тип секрета** тип информации, которая будет храниться в секрете, и задайте настройки выбранного типа:
   - **Учетные данные** — укажите имя пользователя и пароль.
   - **Токен** — укажите токен.
   - **Строка подключения** — введите строку подключения.

     Строка подключения используется в точках входа AMQP и Database. Формат строки отличается в зависимости от типа и настроек точки.

   - **Сертификат** — загрузите файл сертификата. Допустимые форматы: DER, PEM (X.509).
   - **Контейнер PKCS** — контейнер представляет собой архив, в котором хранятся файл сертификата и файл приватного ключа. Чтобы настроить контейнер PKCS:
     - Загрузите файл сертификата. Допустимые форматы: DER, PEM (X.509).
     - Загрузите файл приватного ключа. Допустимые форматы: DER, PEM (PKCS#8).
     - При необходимости укажите пароль для приватного ключа.
   - **Контейнер PKCS#12** — контейнер представляет собой архив, в котором хранится файл сертификата с уже включенным в него приватным ключом. Чтобы настроить контейнер PKCS#12:
     - Загрузите файл сертификата. Допустимый формат: PKCS#12.
     - При необходимости укажите пароль для приватного ключа.
   - **NATS Key** — секрет предназначен для подключения к брокеру сообщений NATS с помощью приватного и публичного ключей. Чтобы настроить NATS Key, введите приватный ключ NKey Seed и публичный ключ NKey User.
6. Нажмите на кнопку **Создать**. Система создаст секрет и отобразит соответствующее

уведомление. Новый секрет появится в списке раздела **Ресурсы** → **Секреты**.

Просмотр секрета

Чтобы просмотреть секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.

В верхней части карточки отображается название секрета.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит следующие поля:

- **ID** — уникальный идентификационный код секрета, генерируемый автоматически при его создании.
- **Название** — имя секрета, используемое для его идентификации в системе.
- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится секрет.
- **Статус** — текущий статус секрета: **Включен** или **Выключен**.
- **Описание** — краткое описание функциональности секрета.
- **Тип секрета** — тип информации, хранящейся в секрете.
- **Дата создания** — дата и время создания секрета.
- **Создал** — пользователь, инициировавший создание секрета.
- **Дата изменения** — дата и время последнего изменения секрета.
- **Изменил** — пользователь, выполнивший последнее изменение секрета.



Можно экспортировать секрет из объектного хранилища NATS Object Store.

Включение секрета

Для того чтобы использовать секрет в системе, его нужно включить.

Чтобы включить секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.
3. Установите переключатель состояния секрета в верхней части карточки в активное положение. Отобразится окно подтверждения включения секрета.
4. Нажмите на кнопку **Включить**. Система включит секрет и отобразит соответствующее уведомление. Секрет станет доступен для добавления в объекты системы.

Если при включении секрета возникли ошибки, система отобразит сообщение об этом. Для

просмотра списка ошибок нажмите на кнопку **Показать**.

Чтобы выключить активный секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.
3. Установите переключатель состояния секрета в верхней части карточки в неактивное положение. Отобразится окно подтверждения выключения секрета.
4. Нажмите на кнопку **Выключить**. Система выключит секрет и отобразит соответствующее уведомление. Секрет перестанет быть доступным для добавления в объекты системы.



Объекты, использующие секрет, продолжат его использовать даже после выключения. Выключение секрета запрещает только его новое добавление в объекты системы.

Изменение секрета

Вы можете изменить секрет следующими способами:

- через меню действий в карточке секрета;
- через изменение полей в карточке секрета.



После изменения секрета необходимо переустановить конфигурацию на тех конвейерах, где требуется использовать обновленные данные секрета.

Удаление секрета

Вы можете удалить секрет через меню действий в его карточке.



Удаление недоступно для секретов, используемых на конвейерах или в интеграциях.

Агенты

В данном разделе описана работа с менеджером агентов в системе. Менеджер позволяет управлять агентами и их группами, а также настраивать шлюзы для подключения агентов к кластерам.

Содержание раздела:

- Работа с агентами
- Работа с группами агентов
  - Настройка конфигурации группы агентов
  - Синтаксис языка XPath
- Работа со шлюзами
- Действия на агентах
  - Ручное снятие самоизоляции узла

Работа с агентами

Данный раздел описывает процесс работы с агентами в системе. Работа осуществляется в разделе **Агенты** → **Агенты** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об агентах

Агенты устанавливаются на конечные устройства и используются для выявления угроз и осуществления реагирования на эти угрозы. Они позволяют осуществлять сетевую изоляцию устройства и производить сбор информации об уязвимостях и угрозах.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Обновить данные** позволяет обновить список агентов.
- Поле **Поиск** предназначено для быстрого поиска агентов в отображаемом списке по имени, группе, IP-адресу или ОС.
- Кнопка **Фильтр** () позволяет открыть в правой части рабочей области панель для настройки фильтрации списка агентов.
- Кнопки команд над выбранными в таблице агентами (становятся доступными после выбора хотя бы одного агента из списка):
  -  — удаляет выбранные агенты;
  -  — отключает выбранные агенты;
  -  — открывает окно для настройки и выполнения специальных действий над выбранными

агентами;

- — создает группу с выбранными агентами;
- — добавляет выбранные агенты в группу;
- — удаляет выбранные агенты из группы.

В рабочей области раздела отображается таблица имеющихся агентов. Таблица содержит следующие столбцы:

- **Название** — название агента, используемое для его идентификации. Слева от названия отображается цветовой индикатор статуса агента:
  - **Выключен** — красный;
  - **Офлайн** — серый;
  - **Онлайн** — зеленый;
  - **Зарегистрирован** — желтый.
- **Группа** — список групп, в которые включен агент.
- **IP-адрес** — список IP-адресов хоста агента.
- **ОС** — операционная система на хосте агента.
- **Версия ОС** — версия операционной системы на хосте.
- **Архитектура** — архитектура процессора на хосте.
- **ID машины** — уникальный идентификационный код компьютера агента.
- **ID агента** — уникальный идентификационный код агента.
- **Версия агента** — версия агента.
- **Дата регистрации** — дата и время регистрации агента.
- **Последняя активность** — дата и время последней активности агента.
- **Тенант** — название тенанта, к которому относится агент.

При работе с таблицей агентов доступны следующие операции:

- Поиск агентов по полям **Название**, **Группа**, **IP-адрес**, **ОС**.
- Фильтрация агентов по полям **Статус**, **Группа**, **Тип ОС**, **Версия агента**, **Дата регистрации**, **Последняя активность**, **Тенант**.
  - Статус, группа, тип ОС, версия агента и последняя активность выбираются из выпадающих списков.
  - Начало и конец диапазона дат регистрации задаются с помощью панели выбора даты и времени.
- Сортировка агентов по полям **Название**, **Группа**, **IP-адрес**, **ОС**, **Архитектура**, **Дата регистрации**, **Последняя активность**.
- Настройка отображения таблицы агентов.

При выборе конкретного агента в правой части рабочей области отображается его карточка с детальной информацией.

В верхнем правом углу карточки расположена кнопка действий ⋮. При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного агента.

Работа с агентом

Доступные операции над агентами:

- Просмотр агента
- Обновление данных о хосте агента
- Отключение агента
- Удаление агента
- Выполнение действий над агентом
- Создание группы на базе агента
- Добавление агента в группу
- Удаление агента из группы

Просмотр агента

Чтобы просмотреть агент:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.

Название агента отображается в верхней части карточки.

Карточка агента содержит следующие вкладки:

- **Общее** — основная информация об агенте.
- **Процессы** — список активных процессов на узле.
- **Порты** — список портов на узле.
- **Контейнеры** — список Docker-контейнеров на агенте.

Вкладка Общее

На вкладке **Общее** отображается основная информация об агенте. Вкладка содержит сведения, сгруппированные по следующим секциям:

- Метрики агента:
  - **CPU** — количество ядер процессора агента.
  - **RAM** — объем оперативной памяти агента.
  - **HDD** — объем жесткого диска агента.
- Основная информация об агенте:
  - **ID агента** — уникальный идентификационный код агента.

- **Название** — имя агента, используемое для его идентификации в системе.
- **ОС** — операционная система на хосте агента.
- **Версия ОС** — версия операционной системы на хосте.
- **Архитектура** — архитектура процессора на хосте.
- **Версия агента** — версия агента.
- **Статус** — текущий статус агента: **Выключен**, **Офлайн**, **Онлайн** или **Зарегистрирован**.
- **IP-адрес** — список IP-адресов хоста агента.
- **Группа** — список групп, в которые включен агент.
- **Тенант** — название тенанта, к которому относится агент.
- **ID машины** — уникальный идентификационный код компьютера агента.
- **Дата регистрации** — дата и время регистрации агента.
- **Последняя активность** — дата и время последней активности агента.

- Список пользователей на агенте. Список представляет собой таблицу со следующими столбцами:

  - **Логин** — логин пользователя.
  - **Последняя активность** — дата и время последнего входа пользователя в систему.

  Для быстрого поиска пользователей в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся пользователи, в логинах или ФИО которых с учетом регистра встречается введенный запрос.

В нижней части вкладки расположена кнопка **Подробнее**, при нажатии на которую открывается более подробная карточка агента.



Подробную карточку также можно открыть, дважды нажав на строку агента в таблице раздела **Агенты** → **Агенты**.

Панель инструментов подробной карточки содержит кнопки команд над агентом:

- — удаляет агент;
- — отключает агент;
- — открывает окно для настройки и выполнения специальных действий над агентом;
- — обновляет инвентарную информацию об агенте;
- — создает группу с агентом;
- — добавляет агент в группу;
- — удаляет агент из группы.

При открытии вкладки **Общее** в подробной карточке агента таблица пользователей на агенте будет содержать следующие колонки:

- **Логин** — логин пользователя.

- **Ф.И.О.** — фамилия, имя и отчество пользователя.
- **Последняя активность** — дата и время последнего входа пользователя в систему.

Вкладка Процессы

На вкладке **Процессы** отображается список активных процессов на узле. Список представляет собой таблицу со следующими столбцами:

- **PID** — уникальный идентификатор процесса.
- **Название** — название процесса.

При открытии вкладки **Процессы** в подробной карточке агента таблица процессов будет содержать следующие столбцы:

- **PID** — уникальный идентификатор процесса.
- **Название** — название процесса.
- **Путь** — подробный путь процесса.

Для быстрого поиска процессов в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся процессы, в полях которых встречается введенный запрос.

Вкладка Порты

На вкладке **Порты** отображается список портов на узле. Список представляет собой таблицу со следующими столбцами:

- **Адрес** — адрес порта.
- **Сервис** — имя сервиса, работающего на порте. При наведении курсора на имя сервиса отображается всплывающее сообщение с его подробной информацией.
- **Протокол** — сетевой протокол, используемый на порте.

При открытии вкладки **Порты** в подробной карточке агента таблица портов будет содержать следующие столбцы:

- **Адрес** — адрес порта.
- **Имя порта** — имя порта.
- **Порт** — порт подключения.
- **Протокол** — сетевой протокол, используемый на порте.

Для быстрого поиска портов в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся порты, в полях которых встречается введенный запрос.

Вкладка Контейнеры

На вкладке **Контейнеры** отображается список Docker-контейнеров на агенте. Список представляет собой таблицу со следующими столбцами:

- **Название** — название контейнера. Слева от названия отображается цветовой индикатор статуса контейнера:

    - **Работает** — зеленый;
    - **Не работает** — серый.
- **Образ** — имя образа контейнера. При наведении курсора на строку с именем образа отображается всплывающее сообщение с его подробной информацией.

Если для контейнера указаны порты, то в правой части его строки отобразится иконка ⇠⇢. При наведении курсора на эту иконку отображается всплывающее сообщение с информацией о портах.

Формат вывода информации о порте

```
<IP-адрес сервера>:<порт сервера> -> <порт контейнера>/<протокол>
```



IP-адрес и порт сервера отображаются только в том случае, если порт контейнера транслируется на сервер.

При открытии вкладки **Контейнеры** в подробной карточке агента таблица контейнеров будет содержать следующие столбцы:

- **Название** — название контейнера.
- **Статус** — текущий статус контейнера: **Работает** или **Не работает**.
- **Команда** — команда, выполняемая при запуске контейнера.
- **Образ** — имя образа контейнера. При наведении курсора на строку с именем образа отображается всплывающее сообщение с его подробной информацией.
- **Порты** — порты, указанные для контейнера.

  Формат вывода информации о порте

  ```
  <IP-адрес сервера>:<порт сервера> -> <порт контейнера>/<протокол>
  ```



IP-адрес и порт сервера отображаются только в том случае, если порт контейнера транслируется на сервер.

Для быстрого поиска контейнеров в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся контейнеры, в полях которых встречается введенный запрос.

Обновление данных о хосте агента

Чтобы обновить инвентарную информацию о хосте агента:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Обновить сведения**. Система отправит запрос на обновление инвентарной информации о хосте выбранного агента и отобразит соответствующее уведомление.

Обновить инвентарную информацию о хосте можно также из подробной карточки агента. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку на панели инструментов. Система отправит запрос на обновление инвентарной информации о хосте выбранного агента и отобразит соответствующее уведомление.

Отключение агента

Чтобы отключить работающий агент:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий в верхнем правом углу карточки и выберите опцию **Отключить**. Отобразится окно подтверждения отключения агента.
4. Нажмите на кнопку **Отключить**. Система отправит запрос на отключение агента и отобразит соответствующее уведомление. Агент перейдет в статус **Выключен**.

Отключить агент можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения отключения агента.
5. Нажмите на кнопку **Отключить**. Система отправит запрос на отключение агента и отобразит соответствующее уведомление. Агент перейдет в статус **Выключен**.

Вы также можете отключить сразу несколько агентов. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. В левом столбце таблицы установите флажки напротив тех агентов, которые вы хотите отключить.
3. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения отключения агентов.

4. Нажмите на кнопку **Отключить**. Система отправит запрос на отключение выбранных агентов и отобразит соответствующее уведомление. Агенты перейдут в статус **Выключен**.

Удаление агента

Вы можете удалить агент следующими способами:

- через меню действий в карточке агента;
- через панель инструментов в подробной карточке агента;
- через массовое удаление агентов.

Выполнение действий над агентом

Чтобы выполнить специальное действие над агентом:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Выполнить действие**. Отобразится окно выбора и настройки действия.
4. Выберите и настройте необходимое действие. Набор настроек отличается в зависимости от выбранного действия.
5. Нажмите на кнопку **Выполнить**. Система отправит команду на выполнение действия на выбранном агенте и отобразит соответствующее уведомление.

Вы также можете выполнить действие сразу над несколькими агентами. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. В левом столбце таблицы установите флажки напротив тех агентов, над которыми вы хотите выполнить действие.
3. Нажмите на кнопку ⚡ на панели инструментов. Отобразится окно выбора и настройки действия.
4. Выберите и настройте необходимое действие. Набор настроек отличается в зависимости от выбранного действия.
5. Нажмите на кнопку **Выполнить**. Система отправит команду на выполнение действия на выбранных агентах и отобразит соответствующее уведомление.

Создание группы на базе агента

В разделе **Агенты** → **Агенты** доступно создание группы на базе агента.

Чтобы создать группу на базе агента:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Создать**

группу. Отобразится окно создания группы.

4. Задайте название группы.
5. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
6. Нажмите на кнопку **Создать**. Система создаст группу с выбранным агентом и отобразит соответствующее уведомление. Созданная группа отобразится в разделе **Агенты → Группы агентов**.



Создать группу на базе агента можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты → Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку + на панели инструментов. Отобразится окно создания группы.
5. Задайте название группы.
6. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
7. Нажмите на кнопку **Создать**. Система создаст группу с выбранным агентом и отобразит соответствующее уведомление. Созданная группа отобразится в разделе **Агенты → Группы агентов**.

Вы также можете создать группу сразу из нескольких агентов. Для этого:

1. Перейдите в раздел **Агенты → Агенты**. Система отобразит сведения об имеющихся агентах.
2. В левом столбце таблицы установите флажки напротив тех агентов, из которых вы хотите сформировать группу.
3. Нажмите на кнопку + на панели инструментов. Отобразится окно создания группы.
4. Задайте название группы.
5. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
6. Нажмите на кнопку **Создать**. Система создаст группу с выбранным агентом и отобразит соответствующее уведомление. Созданная группа отобразится в разделе **Агенты → Группы агентов**.

Добавление агента в группу

Чтобы добавить агент в группу:

1. Перейдите в раздел **Агенты → Агенты**. Система отобразит сведения об имеющихся агентах.

2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Добавить в группу**. Отобразится окно добавления агента в группу.
4. Выберите из выпадающего списка группу, в которую необходимо добавить агент.
5. Нажмите на кнопку **Добавить**. Система добавит агент в выбранную группу и отобразит соответствующее уведомление.



Добавить агент в группу можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку на панели инструментов. Отобразится окно добавления агента в группу.
5. Выберите из выпадающего списка группу, в которую необходимо добавить агент.
6. Нажмите на кнопку **Добавить**. Система добавит агент в выбранную группу и отобразит соответствующее уведомление.

Вы также можете добавить в группу сразу несколько агентов. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. В левом столбце таблицы установите флажки напротив тех агентов, которые вы хотите добавить в группу.
3. Нажмите на кнопку на панели инструментов. Отобразится окно добавления агентов в группу.
4. Выберите из выпадающего списка группу, в которую необходимо добавить агенты.
5. Нажмите на кнопку **Добавить**. Система добавит выделенные агенты в выбранную группу и отобразит соответствующее уведомление.

Удаление агента из группы

Чтобы удалить агент из группы:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Удалить из группы**. Отобразится окно удаления агента из группы.
4. Выберите группу из выпадающего списка, из которой необходимо удалить агент.
5. Нажмите на кнопку **Удалить**. Система удалит агент из выбранной группы и отобразит

соответствующее уведомление.



Удалить агент из группы можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку  на панели инструментов. Отобразится окно удаления агента из группы.
5. Выберите группу из выпадающего списка, из которой необходимо удалить агент.
6. Нажмите на кнопку **Удалить**. Система удалит агент из выбранной группы и отобразит соответствующее уведомление.

Вы также можете удалить из группы сразу несколько агентов. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах.
2. В левом столбце таблицы установите флажки напротив тех агентов, которые вы хотите удалить из группы.
3. Нажмите на кнопку  на панели инструментов. Отобразится окно удаления агентов из группы.
4. Выберите группу из выпадающего списка, из которой необходимо удалить агенты.
5. Нажмите на кнопку **Удалить**. Система удалит выделенные агенты из выбранной группы и отобразит соответствующее уведомление.

Работа с группами агентов

Данный раздел описывает процесс работы с группами агентов в системе. Работа осуществляется в разделе **Агенты** → **Группы агентов** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О группах агентов

Группа агентов объединяют агенты для упрощения выполнения массовых операций над ними.

Для групп агентов доступна настройка конфигурации, которая позволяет задавать политики отслеживания различных действий на агентах в рамках группы.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую группу агентов.
- Поле **Поиск** предназначено для быстрого поиска групп агентов в отображаемом списке по названию и количеству агентов в группе.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка групп.
- Кнопка позволяет удалить выбранные группы агентов. Становится доступной после выбора хотя бы одной группы из списка.

В рабочей области раздела отображается таблица имеющихся групп агентов. Таблица содержит следующие столбцы:

- **Название** — название группы агентов, используемое для ее идентификации
- **Количество агентов** — количество агентов, входящих в группу.
- **Автоназначение** — индикатор включения автоматического назначения группы агенту: **Включено** или **Выключено**.
- **Дата создания** — дата и время создания группы.
- **Дата изменения** — дата и время последнего изменения группы.
- **Тенант** — название тенанта, к которому относится группа.

При работе с таблицей групп агентов доступны следующие операции:

- Поиск групп по полям **Название** и **Количество агентов**.
- Фильтрация групп по полям **Количество агентов**, **Автоназначение**, **Дата создания**, **Дата изменения**, **Тенант**.
  - Начало и конец диапазона количества агентов задаются вручную или с помощью кнопок — и + .
  - Поведение автоназначения выбирается из выпадающего списка.
  - Начало и конец диапазона дат создания и изменения задаются с помощью панели выбора даты и времени.
- Сортировка агентов по полям **Название**, **Количество агентов**, **Дата создания**, **Дата изменения**.
- Настройка отображения таблицы групп.

При выборе конкретной группы в правой части рабочей области отображается ее карточка с детальной информацией.

В верхнем правом углу карточки расположена кнопка действий ⋮ . При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранной группы.

Работа с группой агентов

Доступные операции над группами агентов:

- Создание группы агентов
- Просмотр группы агентов
- Дублирование группы агентов
- Изменение группы агентов
- Удаление группы агентов

Создание группы агентов

Чтобы создать группу агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на кнопку **Создать** (+) на панели инструментов. Система отобразит окно создания группы.



Кнопка доступна, только если ни одна группа агентов из списка не выбрана.

3. Задайте название группы.
4. Если вы работаете в основном тенанте, выберите из выпадающего списка название тенанта, к которому относится группа.



Если у тенанта нет дочерних тенантов, он выбирается системой автоматически. Это поле отредактировать нельзя.

5. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
6. Нажмите на кнопку **Создать**. Система создаст пустую группу и отобразит соответствующее уведомление. Созданная группа отобразится в списке раздела.



Наполнение пустой группы агентами выполняется в разделе **Агенты** → **Агенты**. Для этого доступны следующие операции:

- Добавление агента в группу
- Удаление агента из группы

Просмотр группы агентов

Чтобы просмотреть группу агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.

Название группы агентов отображается в верхней части карточки.

Карточка группы агентов содержит следующие вкладки:

- **Информация** — основная информация о группе агентов.
- **Агенты** — список агентов в группе.

В нижней части карточки содержится кнопка **Конфигурация**, при нажатии на которую открывается страница настройки конфигурации группы агентов.

Вкладка Информация

На вкладке **Информация** отображается основная информация о группе. Вкладка содержит следующие поля:

- **Название** — название группы агентов, используемое для ее идентификации

  С помощью кнопки  название можно изменить.

- **Тенант** — название тенанта, к которому относится группа.
- **Количество агентов** — количество агентов в группе.
- **Автоназначение** — индикатор включения автоматического назначения группы агентам: **Включено** или **Выключено**.
- **Условие** — VRL-выражение для автоматического назначения группы агентам.
- **Дата создания** — дата и время создания группы.
- **Дата изменения** — дата и время последнего изменения группы.

Вкладка Агенты

На вкладке **Агенты** отображается список агентов, входящих в группу. Список представляет собой таблицу со следующими столбцами:

- **Название** — название агента.
- **IP-адрес** — список IP-адресов узлов на агенте.

Для быстрого поиска агентов в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся агенты, в названии или IP-адресе которых встречается введенный запрос.

Чтобы получить подробные сведения об агенте, нажмите на его название в списке. Система откроет раздел **Агенты → Агенты** и отобразит на экране подробную карточку этого агента.

Дублирование группы агентов



Группы дублируются вместе с входящими в них агентами и настроенной конфигурацией.

Чтобы создать копию группы агентов:

1. Перейдите в раздел **Агенты → Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой

группы агентов с подробной информацией о ней.

3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Дублировать**. Отобразится окно создания копии группы.
4. Задайте название копии группы.
5. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
6. Нажмите на кнопку **Дублировать**. Система создаст копию выбранной группы и отобразит соответствующее уведомление. Созданная группа отобразится в списке раздела.



Создать копию группы можно также из экрана настройки ее конфигурации. Для этого:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку **Конфигурация** в нижней части карточки. Откроется страница настройки конфигурации выбранной группы.
4. Нажмите на кнопку на панели инструментов. Отобразится окно создания копии группы.
5. Задайте название копии группы.
6. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
7. Нажмите на кнопку **Дублировать**. Система создаст копию выбранной группы и отобразит соответствующее уведомление. Созданная группа отобразится в списке раздела.

Изменение группы агентов

Вы можете изменить поля группы агентов через ее карточку.

Удаление группы агентов

Вы можете удалить группу агентов следующими способами:

- через меню действий в карточке группы;
- через панель инструментов на экране настройки конфигурации группы;
- через массовое удаление групп.

Настройка конфигурации группы агентов

Конфигурация группы агентов позволяет задать политики отслеживания действий на агентах в рамках группы.

Доступные операции над конфигурацией группы агентов:

- Просмотр конфигурации группы
- Настройка конфигурации группы

Просмотр конфигурации группы

Чтобы просмотреть текущую конфигурацию группы агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку **Конфигурация** в нижней части карточки. Откроется страница настройки конфигурации выбранной группы.



Вы также можете открыть страницу конфигурации при создании группы на базе агента, создании пустой группы, а также дублировании группы путем установки флажка **Перейти к конфигурации**.

Панель инструментов страницы включает в себя следующие компоненты:

- Кнопка  позволяет удалить выбранную группу агентов.
- Кнопка  позволяет создать копию выбранной группы агентов.

На рабочей области страницы отображаются параметры настройки конфигурации, которые сгруппированы по разделам, расположенным слева. Рядом с параметрами конфигурации присутствуют переключатели, позволяющие их активировать.

При активации параметров отображаются их настройки, а также кнопки для сохранения заданных настроек или отмены несохраненных изменений.

Настройка конфигурации группы

Чтобы настроить конфигурацию группы агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку **Конфигурация** в нижней части карточки. Откроется страница настройки конфигурации выбранной группы.



Вы также можете открыть страницу конфигурации при создании группы на базе агента, создании пустой группы, а также дублировании группы путем установки флажка **Перейти к конфигурации**.

4. Задайте настройки конфигурации группы агентов.

Конфигурацию группы агентов следует настраивать аккуратно. Неправильная настройка конфигурации может привести к избыточной нагрузке на агентах и существенно увеличить количество передаваемых событий.

В системе доступна настройка следующих параметров:

- **Автоназначение** — автоматическое назначение группы агенту.
- **Пользователи и процессы**:
  - **Вход/выход пользователей** — отслеживание начала и окончания сеансов пользователей.
  - **Создание процессов** — отслеживание процессов, запущенных пользователями и системой.
  - **Добавление или изменение сервисов** — отслеживание изменений в файлах сервисов.
  - **Управление пользователями** — отслеживание действий по управлению пользователями.
- **Чтение файлов** — сбор данных журнала из файлов и событий Windows.
- **WatchDog** — отслеживание процессов на агенте.
- **Доступ к объектам** — отслеживание действий с объектами в указанных путях.

После настройки любого параметра необходимо нажать кнопку **Сохранить**.

- Если в описании параметра указаны названия ОС, включенный параметр будет применен именно к ним.
- Если в описании параметра не указаны названия ОС, включенный параметр будет применен ко всем ОС.

Автоназначение

Позволяет указать выражение на языке преобразования данных VRL, определяющее условия автоматического назначения группы агенту.

Чтобы использовать автоназначение:

1. Установите переключатель **Автоназначение** в активное положение. Отобразится поле для ввода VRL-выражения.
2. Укажите выражение на языке VRL, по которому группа будет автоматически назначаться агентам. В выражении необходимо обращаться к полям сообщений об инвентарной информации, которые агенты отправляют в менеджер.
3. Нажмите на кнопку **Сохранить**.

Система будет сопоставлять заданное выражение с инвентарными сведениями, собранными с хоста агента. При совпадении данных агент будет включен в текущую группу.

Автоматическое назначение группы происходит только при регистрации новых агентов. Для уже существующих в системе агентов автоназначение не выполняется.

Если инвентарные сведения будут удовлетворять выражению сразу в нескольких группах агентов,

система включит агент во все эти группы.

▼ *Пример автоназначения группы*

- Пример выражения для автоматического добавления в группу всех агентов, установленных на хостах с ОС Linux:

```
.os_info.os_type == 'OS_TYPE_LINUX'
```

- Пример выражения для автоматического добавления в группу всех агентов, установленных на хосте host.example.com:

```
.os_info.hostname == "host.example.com"
```

Пользователи и процессы

Параметры раздела **Пользователи и процессы**:

- Вход/выход пользователей (Linux/macOS)
- Создание процессов (Linux/macOS)
- Добавление или изменение сервисов (Linux)
- Управление пользователями (Linux/macOS)

Вход/выход пользователей (Linux/macOS)

Этот параметр отслеживает начало и окончание сеансов пользователей в системе.



Параметр работает в ОС Linux и macOS.

Чтобы включить параметр, установите переключатель **Вход/выход пользователей (Linux/macOS)** в активное положение и нажмите на кнопку **Сохранить**.

Создание процессов (Linux/macOS)

Этот параметр отслеживает процессы, запущенные пользователями или системой.



Параметр работает в ОС Linux и macOS.

Чтобы включить параметр, установите переключатель **Создание процессов (Linux/macOS)** в активное положение и нажмите на кнопку **Сохранить**.

Добавление или изменение сервисов (Linux)

Этот параметр отслеживает добавление и изменение сервисных файлов `.systemd`, `.timer` и `rc`.



Параметр работает в ОС Linux.

Чтобы включить параметр, установите переключатель **Добавление или изменение сервисов (Linux)**

в активное положение и нажмите на кнопку **Сохранить**.

Управление пользователями (Linux/macOS)

Этот параметр отслеживает изменения пользователей, групп пользователей, прав sudo и настроек окружения (bashrc).



Параметр работает в ОС Linux и macOS.

Чтобы включить параметр, установите переключатель **Управление пользователями (Linux)** в активное положение и нажмите на кнопку **Сохранить**.

Чтение файлов

Этот параметр используется для настройки сбора данных журнала из файлов и событий ОС.

Значимые события (например, ошибки при работе приложений) записываются системой, под которой работает агент, в специальные журналы событий. Также данные можно собирать из файлов в системе агента.

При настройке параметра требуется указать, каким способом и по какому пути агенты текущей группы должны собирать данные.

Чтобы настроить параметр:

1. Установите переключатель **Чтение файлов** в активное положение. Система отобразит настройки сбора данных журнала.
2. Задайте настройки сбора данных.
3. Нажмите на кнопку **Сохранить**. Система сохранит заданные настройки сбора данных с агентов.

Добавление настройки сбора данных

Чтобы добавить настройку сбора данных в параметре **Чтение файлов**:

1. Нажмите на кнопку **Добавить настройку** (+). Отобразится строка с набором полей для настройки нового типа журнала.
2. Выберите тип журнала из выпадающего списка и задайте его настройки.

   Доступные типы журналов:

   - **audit** — используется для событий из инструмента Linux Audit Daemon (auditd). Объединяет последовательные журналы с одинаковым идентификатором в одно событие.
   - **eventchannel** — используется для журналов событий Windows, получает события в формате JSON, контролирует все каналы, указанные в файле конфигурации, и показывает все включенные в них поля. Может использоваться для мониторинга стандартных журналов событий Windows, а также журналов приложений и служб.

     При выборе этого типа становятся доступны следующие настройки:

     - **Имя журнала** — имя журнала событий Windows.

- **Фильтр** — запрос на языке XPath, позволяющий указать критерии, по которым будут отфильтрованы собираемые данные.

  ▼ *Примеры XPath-запросов для фильтрации*

  - Для сбора только событий с идентификатором 5145:

    ```
    *[System[EventID = 5145]]
    ```

  - Для сбора всех событий, кроме событий с идентификатором 5145 или 5196:

    ```
    *[System[(EventID!=5145 and EventID!=5196)]]
    ```

- **eventfile** — используется для сбора событий Windows из файлов формата EVTX.

  При выборе этого типа становятся доступна настройка **Фильтр** — запрос на языке XPath, позволяющий указать критерии, по которым будут отфильтрованы собираемые данные.

  ▼ *Примеры XPath-запросов для фильтрации*

  - Для сбора только событий с идентификатором 5145:

    ```
    *[System[EventID = 5145]]
    ```

  - Для сбора всех событий, кроме событий с идентификатором 5145 или 5196:

    ```
    *[System[(EventID!=5145 and EventID!=5196)]]
    ```

- **eventlog** — используется для классического формата журнала событий Windows.
- **syslog** — используется для простых текстовых файлов в формате, подобном системному журналу.

Для каждого типа журнала, кроме **eventchannel**, нужно указать путь, используемый системой при сборе данных на агенте, в поле **Путь**.

Удаление настройки сбора данных

Чтобы удалить настройки журнала, нажмите на кнопку справа от его строки настроек.

WatchDog

Этот параметр запускает встроенный функционал WatchDog, позволяющий:

- автоматически перезапускать агент в случае его остановки;
- предотвращать несанкционированное завершение работы агента;
- информировать менеджер о попытках остановки или завершения контролируемых процессов.

Чтобы включить и настроить параметр:

1. Установите переключатель **WatchDog** в активное положение.
2. По умолчанию при активации WatchDog операции остановки и перезапуска агента могут выполняться без ограничений.

   Чтобы остановка или перезапуск агента происходили по паролю, установите флажок **Пароль** и введите необходимый пароль в появившемся поле. Пароль может содержать:

   - строчные латинские буквы (a–z);
   - заглавные латинские буквы (A–Z);
   - цифры (0–9);
   - специальные символы:

     ```
     ~"!@#$%^&*_-+=`(){}[]:;'<>,.?/|\
     ```

3. Нажмите на кнопку **Сохранить**. Система сохранит заданные настройки WatchDog.

Доступ к объектам (Linux)

Этот параметр отслеживает действия с объектами в указанных путях. Отслеживаются чтение, запись и запуск объектов, а также изменение прав доступа к ним.



Политика работает в ОС Linux.

Чтобы включить и настроить параметр:

1. Установите переключатель **Доступ к объектам (Linux)** в активное положение.
2. Задайте перечень путей к объектам. Для этого:
   a. Укажите путь и задайте права доступа к объектам, хранящимся по этому пути, посредством установки соответствующих флажков:
      - **w (Writing)** — запись;
      - **r (Reading)** — чтение;
      - **x (Execution)** — выполнение;
      - **a (Attributes)** — атрибуты.
   b. Чтобы добавить следующий путь, нажмите на кнопку **Добавить путь** (+). Появится строка настройки нового пути.
   c. Чтобы удалить ненужные пути, нажмите на кнопку справа от их строк.
3. Задайте при необходимости шаблоны исключений в секции **Исключения**:
   - **По пути** — исключают из рассмотрения указанные пути.
   - **По имени файлов** — исключают из рассмотрения файлы с указанными именами.
   - **По пути родительского процесса** — исключают из рассмотрения указанные пути родительского процесса.

   При задании шаблонов исключений поддерживается использование символа-подстановки (*).

Например, исключение по имени файлов вида `*.log` позволит исключить из рассмотрения все файлы с расширением `.log`.

4. Нажмите на кнопку **Сохранить**. Система сохранит заданные настройки доступа к объектам.

Синтаксис языка XPath

При настройке конфигурации групп агентов в параметре **Чтение файлов** доступно задание фильтров на языке XPath.

- Операторы и специальные символы
- Коллекции
- Операторы сравнения
- Логические выражения
- Выражение набора
- Функции
- Оси

Операторы и специальные символы

Обозначение	Описание
`/`	Дочерний элемент.
`/descendant-or-self::node()` `//`	Рекурсивный спуск.
`self::node()` `.`	Текущий контекст.
`parent::node()` `..`	Родитель узла текущего контекста.
`child::*` `*`	Все элементы.
`@`	Префикс имени атрибута.
`attribute::*` `@*`	Все атрибуты.
`:`	Разделитель пространства имен.
`( )`	Группировка операций.
`[ ]`	Шаблон фильтра.
`+`	Сложение.
`-`	Вычитание.
`div`	Деление с плавающей запятой.

Обозначение	Описание
`*`	Умножение.
`mod`	Остаток от деления.

Коллекции

Обозначение	Описание
`author[1]`	Первый элемент `<author>`.

Операторы сравнения

Обозначение	Описание
`<`	Меньше.
`<=`	Меньше или равно.
`=`	Равно.
`>`	Больше.
`>=`	Больше или равно.

Логические выражения

Обозначение	Описание
`and`	Логическое "И".
`or`	Логическое "ИЛИ".
`not()`	Отрицание.

Выражение набора

Обозначение	Описание
`|`	Операция множества для объединения узлов.

Функции

- Функции для наборов узлов
- Строковые функции
- Логические функции
- Численные функции

Функции для наборов узлов

Обозначение	Описание
`count()`	Получение количества узлов в множестве.
`id()`	Выбор элемента с уникальным идентификатором.

Обозначение	Описание
`last()`	Получение номера последнего элемента в множестве.
`local-name()`	Получение имени первого узла в множестве.
`name()`	Получение полного имени узла в множестве.
`namespace-uri()`	Получение URI-кода пространства имен.
`position()`	Получение позиции элемента в множестве.

Строковые функции

Обозначение	Описание
`concat(string, string, string*)`	Сцепление строк
`contains(string, string)`	Определение, есть ли в заданной строке искомая строка.
`normalize-space(string)`	Удаление начальных, конечных и повторяющихся пробелов в строке.
`starts-with(string, string)`	Определение, есть ли в начале строки искомая строка.
`string(object)`	Преобразование объекта в строку.
`string-length(object)`	Получение числа символов в строке.
`substring(string, number, number?)`	Получение подстроки.
`substring-after(string, string)`	Получение подстроки, которая следует за вторым аргументом строки.
`substring-before(string, string)`	Получение подстроки, которая следует до второго аргумента строки.
`translate(string, string, string)`	Получение строки, в которой символы из второго аргумента заменены на символы третьего аргумента с учетом позиции.

Логические функции

Обозначение	Описание
`boolean()`	Преобразование аргумента в логическое значение.
`false()`	Получение значения `false`.
`lang(string)`	Определение, совпадает ли атрибут `xml:lang` со строкой аргумента.
`not()`	Отрицание логического значения.
`true()`	Получение значения `true`.

Численные функции

Обозначение	Описание
`ceiling()`	Округление в большую сторону.

Обозначение	Описание
`floor()`	Округление в меньшую сторону.
`number()`	Преобразование аргумента в число.
`round()`	Округление до ближайшего целого числа.
`sum()`	Получение суммы всех узлов в множестве.

Оси

Обозначение	Описание
`ancestor::`	Предки узла контекста.
`ancestor-or-self::`	Узел контекста и его предки.
`@ attribute`	Атрибуты контекстного узла.
`child::`	Дочерние элементы узла контекста.

`.//`
 	
`descendant`
	Потомки контекстного узла.

`descendant-or-self::`	Узел контекста и его потомки.
`following::`	Все узлы, расположенные на дереве после узла контекста.
`following-sibling::`	Все следующие элементы узла контекста с общим родителем.
`namespace::`	Узлы пространства имен узла контекста.

`..`
 	
`parent`
	Родитель узла контекста.

`preceding::`	Все узлы, расположенные на дереве перед узлом контекста.
`preceding-sibling::`	Все предшествующие элементы узла контекста с общим родителем.

`.`
 	
`self`
	Только сам контекстный узел.

Работа со шлюзами

Данный раздел описывает процесс работы со шлюзами в системе. Работа осуществляется в разделе **Агенты** → **Шлюзы** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О шлюзах

Шлюз представляет собой промежуточный уровень в сети между агентами, установленными на

конечных устройствах, и центральным кластером.

Шлюзы предназначены для управления потоками данных между агентами и центральным кластером. Они позволяют настраивать маршрутизацию, агрегацию и фильтрацию данных.



Подробности подключения агента к кластеру с использованием шлюзов описаны в разделе Подключение агента к системе.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый шлюз.
- Кнопка  позволяет обновить список шлюзов.
- Кнопка  позволяет удалить выбранные шлюзы.
- Поле **Поиск** предназначено для быстрого поиска шлюзов в отображаемом списке по названию и пространству.
- Кнопка **Фильтр** () позволяет открыть в правой части рабочей области панель для настройки фильтрации списка шлюзов.

В рабочей области раздела отображается таблица имеющихся шлюзов. Таблица содержит следующие столбцы:

- **Название** — название шлюза, используемое для его идентификации.
- **Статус** — текущий статус шлюза:
  - **В очереди** — шлюз находится в очереди на запуск.
  - **Запускается** — шлюз находится в процессе запуска.
  - **Запущен** — шлюз запущен и готов к подключениям.
  - **Останавливается** — шлюз находится в процессе остановки.
  - **Остановлен** — шлюз остановлен и недоступен для взаимодействия.
  - **Ошибка** — ошибка размещения шлюза в пространстве.
  - **Неизвестно** — неизвестное состояние шлюза.
- **Пространство** — название подключения к пространству, в котором создан шлюз.

При работе с таблицей шлюзов доступны следующие операции:

- Поиск шлюзов по полям **Название** и **Пространство**.
- Фильтрация шлюзов по полю **Пространство**. Пространство выбирается из выпадающего списка.
- Сортировка шлюзов по полям **Название** и **Пространство**.
- Настройка отображения таблицы шлюзов.

При выборе конкретного шлюза в правой части рабочей области отображается его карточка с детальной информацией.

Работа со шлюзом

Доступные операции над шлюзом:

- Создание шлюза
- Просмотр шлюза
- Изменение шлюза
- Удаление шлюза
- Запуск шлюза
- Остановка шлюза

Создание шлюза

Чтобы создать шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на кнопку **Создать** (＋) на панели инструментов. Система отобразит окно создания шлюза.
3. Задайте название шлюза.
4. Выберите подключение к пространству, в котором необходимо создать шлюз.
5. Укажите IP- или FQDN-адрес для подключения к шлюзу.

   ▼ *Особенности задания адресов для шлюза*

   Для подключения к шлюзу необходимо использовать адрес узла кластера, где создается шлюз. Так, например:

   - Если шлюз создается в основном экземпляре системы, то используйте IP-адрес любого узла центрального кластера.
   - Если шлюз создается в сателлите, используйте IP-адреса любого узла кластера сателлита.

   FQDN-адрес можно использовать, только если в инфраструктуре существует DNS-сервер и администратор настроил прямую зону (forward zone). Проверить, настроена ли такая зона, можно с помощью утилиты nslookup:

   ```
   nslookup <domain_name>
   ```

   Здесь:

   - `<domain_name>` — доменное имя.

   Обратные DNS-запросы для определения FQDN-адресов по IP-адресам возможны, если на DNS-сервере настроена обратная зона (reverse zone). В ней хранятся записи, сопоставляющие IP-адреса с доменными именами. Проверить, настроена ли такая зона, можно также с помощью утилиты nslookup:

```
nslookup <IP_address>
```

Здесь:

- `<IP_address>` — IP-адрес узла.

Если обратная зона не настроена, свяжитесь с сетевым администратором.

Если ваш компьютер имеет доступ к кластеру с установленным файлом конфигурации `kubeconfig`, вы можете получить сведения обо всех IP-адресах узлов кластера. Для сбора IP-адресов всех узлов в кластере и их вывода в виде строки используйте команду:

```
kubectl get node -o json
--output='jsonpath={.items[*].status.addresses[0].address}'
```

6. Укажите порт для подключения к шлюзу. Допустимые значения: 30 000—32 767.



Порт должен быть уникальным в рамках инсталляции системы.

7. Задайте максимальный размер буфера в МБ для хранения событий ИБ.



Эффективный размер буфера будет равен 80% от указанного значения.

8. При необходимости введите описание шлюза.
9. При необходимости шифрования потока данных между агентом и шлюзом через TLS установите переключатель **Включить TLS** в активное положение.

   При включении TLS необходимо выбрать секрет типа **Контейнер PKCS**, хранящий данные сертификата TLS.



Выбранный секрет запоминается системой и сохраняется в шлюзе. Таким образом, в случае удаления секрета шлюз продолжит работать без ошибок, даже если будет выключен и включен заново.



Все шлюзы в системе должны быть либо с TLS, либо без TLS. В ситуациях, когда для одной части шлюзов настроено TLS-шифрование, а для другой — нет, корректное подключение агентов к кластеру не гарантируется.

10. Нажмите на кнопку **Создать**. Система создаст шлюз и отобразит соответствующее уведомление. Созданный шлюз отобразится в списке раздела и будет запущен.

Просмотр шлюза

Чтобы просмотреть шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на строку шлюза в списке. Система отобразит в правой части экрана карточку этого

шлюза с подробной информацией о нем.

Название шлюза отображается в верхней части карточки.

Карточка содержит следующие сведения о шлюзе:

- **ID шлюза** — уникальный идентификационный код шлюза, генерируемый автоматически при его создании.
- **Название** — имя шлюза, используемое для его идентификации в системе.
- **Описание** — краткое описание функциональности шлюза.
- **Статус** — текущий статус шлюза:
  - **В очереди**;
  - **Запускается**;
  - **Запущен**;
  - **Останавливается**;
  - **Остановлен**;
  - **Ошибка**;
  - **Неизвестно**.
- **IP-адрес или FQDN** — IP- или FQDN-адрес для подключения к шлюзу.
- **Порт** — порт для подключения шлюза к другим системам.
- **Пространство** — название подключения к пространству, в котором создан шлюз.
- **Размер стрима для событий ИБ** — максимальный размер буфера для хранения событий ИБ. Если размер буфера меньше 1024 МБ, то значение отображается в МБ, иначе — в ГБ.
- **Шифрование** — текущий статус шифрования через TLS. Доступные значения:
  - **Включено** — шифрование соединения к шлюзу через TLS включено.
  - **Выключено** — шифрование соединения к шлюзу через TLS выключено.
- **Секрет** — секрет, использующийся для TLS-шифрования.

  Чтобы просмотреть секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Изменение шлюза

Вы можете изменить поля шлюза через его карточку.

Удаление шлюза

Вы можете удалить шлюз следующими способами:

- через кнопку в карточке шлюза;
- через массовое удаление шлюзов.

Запуск шлюза

Чтобы запустить остановленный шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на строку шлюза в списке. Система отобразит в правой части экрана карточку этого шлюза с подробной информацией о нем.
3. Установите переключатель в верхней части вкладки в активное положение. Система запустит шлюз и отобразит соответствующее уведомление. Шлюз перейдет в статус **Запущен**.

Остановка шлюза

Чтобы остановить запущенный шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на строку шлюза в списке. Система отобразит в правой части экрана карточку этого шлюза с подробной информацией о нем.
3. Установите переключатель в верхней части вкладки в неактивное положение. Система остановит шлюз и отобразит соответствующее уведомление. Шлюз перейдет в статус **Остановлен**.

   Если для агента включена возможность переключения между шлюзами и агент остается в статусе **Онлайн**, то после остановки шлюза агент автоматически подключится к кластеру системы через другой шлюз и останется в том же статусе. В противном случае после остановки шлюза агент перейдет в статус **Офлайн**.

   Подробные сведения о данном механизме приведены в описании параметра `gateway_discovery_enabled` файла конфигурации `rpoint.conf`.

Действия на агентах

В разделе **Агенты** → **Агенты** доступно выполнение специальных действий на агентах.



При выполнении специальных действий на агентах система автоматически генерирует соответствующие события аудита.

Настройка действий

В системе доступны следующие специальные действия на агенте:

- **Удаление файла** () — удаление файла или директории на узле.
- **Самоизоляция узла** () — блокировка всех сетевых соединений, кроме исключений.
- **Снятие самоизоляции узла** () — разблокировка всех сетевых соединений.
- **Остановка процесса** () — остановка выполнения указанного процесса на агенте.
- **Отправка файла** () — отправка файла на указанный URL.
- **Добавление записи в hosts (DNS Sinkholing)** () — добавление записи в файл `hosts` для блокировки доступа к веб-сайтам или IP-адресам с помощью техники DNS Sinkholing.

- **Удаление записи из hosts (DNS Sinkholing)** () — удаление записи из файла `hosts` для разблокировки доступа к веб-сайтам или IP-адресам.

Удаление файла

Действие позволяет удалить указанный файл или директорию на узле.



- При удалении директории удаляются все содержащиеся в ней файлы.
- Если файл или директория заняты процессами, агент останавливает эти процессы и удаляет файл или директорию. События остановки процессов записываются в журнал агента.

Чтобы выполнить действие:

1. Укажите путь к файлу или директории, которые требуется удалить.
2. Если требуется удалить директорию, установите переключатель **Удалить директорию** в активное положение.
3. Нажмите на кнопку **Выполнить**.

Самоизоляция узла

Действие **Самоизоляция узла** () позволяет блокировать все входящие и исходящие сетевые соединения по всем портам, кроме:

- Соединений с управляющим сервером (менеджером), указанным в файле конфигурации nats-leafnode.conf.
- Соединений с IP-адресами из белого списка, заданного в настройках менеджера агентов.
- Портов из следующего списка:
  - 53 — DNS.
  - 67 — DHCP-сервер.
  - 68 — DHCP-клиент.
  - 88 — Kerberos/Active Directory Authentication.
  - 123 — NTP.
  - 389 — LDAP-сервер.
  - 546 — DHCPv6-клиент.
  - 547 — DHCPv6-сервер.
  - 647 — DHCP failover.
  - 847 — DHCP failover.
  - 853 — DNS over TLS.
  - 5353 — Multicast DNS (mDNS).
  - 5355 — Link-Local Multicast Name Resolution (LLMNR).

Действие **Снятие самоизоляции узла** () отменяет эту изоляцию.



Если отменить самоизоляцию узла через команду не удается, обратитесь к разделу Ручное снятие самоизоляции узла.

Чтобы выполнить действие, нажмите на кнопку **Выполнить**.

Остановка процесса

Действие позволяет остановить процесс, выполняющийся на узле.

Чтобы выполнить действие:

1. Выберите способ идентификации процесса, который требуется остановить, из выпадающего списка, и укажите соответствующий идентификатор:
   - **PID** — PID процесса.
   - **Путь** — путь до объекта процесса на узле.
   - **Имя** — имя бинарного файла процесса.
2. Нажмите на кнопку **Выполнить**.

Отправка файла

Действие позволяет отправить файл с агента на указанный URL.

Чтобы выполнить действие:

1. Укажите путь к файлу на агенте, который требуется отправить.
2. Укажите URL, на который требуется отправить файл.
3. Выберите метод отправки: POST или PUT.
4. Если проверять валидность SSL-сертификата при подключении не требуется, установите переключатель **Игнорирование SSL** в активное положение.
5. Укажите заголовок токена.



Заголовок токена определяется производителем системы назначения.

6. Укажите токен API для аутентификации при отправке файла.
7. Укажите тип отправляемого файла.



Тип файла определяется производителем системы назначения.

8. Нажмите на кнопку **Выполнить**.

Добавление записи в hosts (DNS Sinkholing)

Действие позволяет добавить запись в файл `hosts` для блокировки доступа к веб-сайтам или IP-адресам с помощью техники DNS Sinkholing.

Чтобы выполнить действие:

1. Укажите IP-адрес сервера, доступ к которому требуется заблокировать.
2. Укажите домен, доступ к которому требуется заблокировать.
3. При необходимости введите поясняющий комментарий к действию.
4. Нажмите на кнопку **Выполнить**.

Удаление записи из hosts (DNS Sinkholing)

Действие позволяет удалить запись из файла `hosts` для разблокировки доступа к веб-сайтам или IP-адресам.

Чтобы выполнить действие:

1. Укажите IP-адрес сервера, доступ к которому требуется разблокировать.
2. Укажите домен, доступ к которому требуется разблокировать.
3. Нажмите на кнопку **Выполнить**.

Ручное снятие самоизоляции узла

При работе с агентами возможны ситуации, при которых нельзя отправить команду на снятие самоизоляции узла с IP-адресов, находящихся в белом списке. Это связано с невозможностью передачи команды от менеджера к агенту, например, из-за отключения шлюза или изменения его IP-адреса.

В таких ситуациях снятие самоизоляции можно произвести вручную. Алгоритм ручного снятия самоизоляции узла зависит от используемой ОС:

- Снятие самоизоляции в ОС Linux
- Снятие самоизоляции в macOS



Ручное снятие самоизоляции в ОС Windows недоступно.

Снятие самоизоляции в ОС Linux

Чтобы отключить самоизоляцию узла в ОС Linux, последовательно выполните следующие действия на узле:

1. Удалите фильтры агента с помощью утилиты iptables:

```
sudo iptables -F rpoint-ar-self-isolation
```

2. Удалите фильтры агента с помощью утилиты nft:

```
sudo nft delete table inet rpoint-ar-self-isolation
```

3. Удалите фильтры агента с помощью утилиты ufw. Для этого:
   a. Получите список всех правил фильтрации, выполнив команду:

```
sudo ufw status numbered
```

b. Для каждого правила с комментарием `rpoint-ar-self-isolation` выполните следующую команду:

```
sudo ufw delete <line_number>
```

Здесь:

- `<line_number>` — номер строки правила.

Снятие самоизоляции в macOS

Чтобы отключить самоизоляцию узла в macOS, выполните следующие действия на узле:

1. Очистите содержимое файла `/etc/pf.anchors/rpoint-ar-self-isolation`.
2. Перезапустите файрвол Packet Filter:

```
sudo pfctl -f /etc/pf.conf
```

Экспертиза

В данном разделе описаны инструменты и методы работы с элементами экспертизы в системе, включая схемы RObject и язык преобразования данных VRL.

Содержание раздела:

- Работа с элементами экспертизы
- Правила корреляции
  - Построение правила корреляции в конструкторе
  - Императивные правила
  - Декларативные правила
- Правила нормализации
- Правила агрегации
- Правила сегментации
- Схемы активных списков
- Схемы дашбордов
- Схемы шаблонов отчетов
- Таблицы обогащения
- Шаблоны поиска
- Глобальные функции
- Тестирование правил экспертизы с помощью языка VRL
- Оценка покрытия техник MITRE ATT&CK правилами экспертизы

Работа с элементами экспертизы

Данный раздел описывает процесс работы с элементами экспертизы в системе. Работа осуществляется в разделе **Экспертиза** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об элементах экспертизы

Экспертиза — модуль системы, позволяющий создавать объекты на языке RObject для специализированных задач. RObject представляет собой синтаксическую надстройку над YAML. Объекты, создаваемые в экспертизе, называются **элементами экспертизы**.

Система позволяет работать со следующими элементами экспертизы:

- Правила корреляции

- Правила нормализации
- Правила агрегации
- Правила сегментации
- Схемы активных списков
- Схемы дашбордов
- Схемы шаблонов отчетов
- Таблицы обогащения
- Шаблоны поиска
- Глобальные функции

В системе доступны элементы экспертизы следующих типов:

- **Системные элементы экспертизы** — создаются системой автоматически. Их нельзя переносить между каталогами, изменять, удалять или экспортировать.
- **Пользовательские элементы экспертизы** — создаются вручную или в результате импорта.

Если при установке системы были импортированы пакеты экспертизы, в разделе **Экспертиза** будут доступны элементы из этих пакетов.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый элемент экспертизы. По этой кнопке открывается выпадающий список для выбора типа создаваемого элемента.
- Кнопка позволяет удалить выбранные элементы экспертизы.
- Кнопка позволяет перенести выбранные из списка элементы экспертизы в другой каталог.
- Кнопка позволяет импортировать элементы экспертизы в систему.
- Кнопка позволяет экспортировать выбранные элементы экспертизы из системы в формате ROC.
- Кнопка позволяет обновить версии выбранных правил экспертизы на конвейерах.
- Переключатель **Каталоги** позволяет показать или скрыть в левой части рабочей области панель с деревом каталогов.
- Поле **Поиск** предназначено для быстрого поиска элементов экспертизы в отображаемом списке.
- Кнопка **Фильтр** (▽) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка элементов экспертизы.

Панель **Каталоги** содержит корневые каталоги и вложенные каталоги, относящиеся к ним. При нажатии на кнопку ⋮ в верхнем правом углу панели открывается выпадающее меню с кнопкой **Создать каталог** для добавления нового корневого каталога.

При наведении курсора на название вложенного каталога в строке появляется кнопка ⋮. При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного каталога.

При выборе каталога система отображает список элементов экспертизы в этом каталоге. Список представлен в виде таблицы со следующими столбцами:

- **Тип элемента** — тип элемента экспертизы.
- **ID** — уникальный идентификационный код, который автоматически присваивается системой при создании элемента.
- **Название** — имя элемента экспертизы, используемое для его идентификации в системе.
- **Версия** — версия элемента экспертизы, представленная в формате SemVer v2.0. Неопубликованные элементы экспертизы (черновики) имеют значение **Черновик**.
- **Теги** — пользовательские теги, присвоенные данному элементу экспертизы.
- **Источник данных** — список совместимых с элементом экспертизы источников данных, присвоенных элементу в поле `data_source`.
- **Дата изменения** — дата и время последнего изменения элемента экспертизы.
- **Уровень угрозы** — уровень угрозы, ассоциированный с элементом экспертизы: **Низкий**, **Средний**, **Высокий**, **Критический** или **Не указан**.
- **Статус** — текущий статус элемента экспертизы: **Включен** или **Выключен**.
- **Изменил** — пользователь, выполнивший последнее изменение элемента экспертизы.

При работе с таблицей элементов экспертизы доступны следующие операции:

- Поиск элементов экспертизы. Поиск осуществляется по полям **ID** и **Название**, а также по коду текущих версий элементов.
- Фильтрация элементов экспертизы. В параметрах фильтрации доступны следующие опции:
  - **Тип элемента** — тип элемента экспертизы. Выбирается из выпадающего списка. Доступен выбор нескольких вариантов.
  - **Теги** — пользовательские теги. Вводятся вручную и применяются с помощью клавиши ENTER.
  - **Источники данных** — список источников данных. Вводится вручную и применяется с помощью клавиши ENTER.
  - **Статус** — текущий статус элемента экспертизы. Выбирается из выпадающего списка.
  - **Уровень угрозы** — уровень угрозы, ассоциированный с элементом экспертизы. Выбирается из выпадающего списка.
  - **Тип создания** — тип создания элемента экспертизы: системный или пользовательский. Выбирается из выпадающего списка.
  - **Дата изменения** — временной диапазон, в котором произошло изменение элемента экспертизы. Начало и конец диапазона можно ввести вручную или задать с помощью панели выбора даты и времени.
  - **Изменил** — пользователь, выполнивший последнее изменение элемента экспертизы. Выбирается из выпадающего списка.



Для удобства работы с разделом заданные условия фильтрации отображаются над списком элементов. Условие фильтрации можно сбросить с помощью кнопки ✕ справа от него. Чтобы сбросить сразу все условия, используйте

кнопку **Сбросить все** (✕) справа от списка условий.



Перейти к списку правил корреляции, отфильтрованных по тегу какого-либо приема, который используют злоумышленники, можно также из раздела **Покрытие MITRE ATT&CK**.

- Сортировка элементов экспертизы по полям **Название** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного элемента экспертизы в правой части рабочей области отображается его карточка с детальной информацией.

Работа с каталогами

Элементы экспертизы хранятся в каталогах и подкаталогах. В левой части рабочей области интерфейса представлена иерархическая структура каталогов. Корневые каталоги содержат общий перечень элементов экспертизы, включая элементы из подкаталогов. При выборе конкретного каталога в древовидной структуре, содержащиеся в нем элементы экспертизы отображаются в таблице в центральной части рабочей области.

Доступные операции над каталогами:

- Создание каталога
- Переименование каталога
- Перемещение каталога
- Удаление каталога

Создание каталога

Вы можете создавать корневые или вложенные каталоги в разделе **Экспертиза**.

Чтобы создать корневой каталог:

1. Перейдите в раздел **Экспертиза**. В левой части рабочей области находится панель **Каталоги**, которая содержит перечень доступных в системе каталогов.
2. Нажмите на кнопку ⋮ справа от названия панели и выберите опцию **Создать каталог**. Отобразится окно создания каталога.
3. Введите название каталога и нажмите на кнопку **Создать**. Система создаст каталог и отобразит соответствующее уведомление. Новый каталог появится в списке.



Имена корневых каталогов должны быть уникальными с учетом регистра.

Чтобы создать вложенный каталог:

1. Выберите в дереве каталог, в котором требуется создать вложенный каталог.
2. Нажмите на появившуюся кнопку действий ⋮ напротив названия каталога и выберите опцию **Создать каталог**. Отобразится окно создания каталога.

3. Введите название каталога и нажмите на кнопку **Создать**. Система создаст каталог и отобразит соответствующее уведомление. Новый каталог появится в дереве внутри выбранного каталога.



Имена вложенных каталогов в пределах родительского каталога должны быть уникальными с учетом регистра.

Переименование каталога



Переименовать можно только вложенные каталоги. Имена каталогов в пределах родительского каталога должны быть уникальными с учетом регистра.

Чтобы переименовать каталог:

1. Перейдите в раздел **Экспертиза**. В левой части рабочей области находится панель **Каталоги**, которая содержит перечень доступных в системе каталогов.
2. Нажмите на появившуюся кнопку действий ⋮ напротив названия каталога и выберите опцию **Переименовать**. Отобразится окно переименования каталога.
3. Измените название каталога и нажмите на кнопку **Переименовать**. Система переименует каталог и отобразит соответствующее уведомление. Каталог отобразится в дереве под новым именем.

Перемещение каталога



Переместить можно только вложенные каталоги. Имена каталогов в пределах родительского каталога должны быть уникальными с учетом регистра.

Чтобы переместить каталог:

1. Перейдите в раздел **Экспертиза**. В левой части рабочей области находится панель **Каталоги**, которая содержит перечень доступных в системе каталогов.
2. Нажмите на появившуюся кнопку действий ⋮ напротив названия каталога и выберите опцию **Переместить**. Отобразится окно перемещения каталога.
3. В окне перемещения каталога раскройте дерево каталогов и выберите новый родительский каталог для текущего каталога.
4. Нажмите на кнопку **Переместить**. Система переместит каталог со всеми вложенными каталогами и элементами в выбранное местоположение и отобразит соответствующее уведомление.

Удаление каталога



Удалить можно только вложенные каталоги, которые не содержат пользовательских элементов экспертизы.

1. Перейдите в раздел **Экспертиза**. В левой части рабочей области находится панель **Каталоги**, которая содержит перечень доступных в системе каталогов.
2. Нажмите на появившуюся кнопку действий ⋮ напротив названия каталога и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.

3. Нажмите **Удалить**. Система удалит каталог и отобразит соответствующее уведомление. Удаленный каталог исчезнет из дерева.

Работа с элементами экспертизы

К элементам экспертизы относятся:

- Правила корреляции
- Правила нормализации
- Правила агрегации
- Правила сегментации
- Схемы активных списков
- Схемы дашбордов
- Схемы шаблонов отчетов
- Таблицы обогащения
- Шаблоны поиска
- Глобальные функции

Доступные операции над элементами экспертизы:

- Версионирование элемента экспертизы
- Создание элемента экспертизы
- Изменение элемента экспертизы
- Работа в окне настройки
- Тестирование элемента экспертизы
- Просмотр элемента экспертизы
- Включение элемента экспертизы
- Выключение элемента экспертизы
- Обновление элемента экспертизы на конвейерах
- Перенос элемента экспертизы
- Создание копии элемента экспертизы
- Копирование кода элемента экспертизы
- Удаление элемента экспертизы
- Экспорт элементов экспертизы
- Импорт элементов экспертизы

Версионирование элемента экспертизы

Работа с элементом экспертизы зависит от типа его версии. Доступны следующие типы:

- **Опубликованная версия** — полноценный рабочий элемент экспертизы. При публикации элемента

экспертизы система автоматически запускает тесты для валидации его конфигурации.

Для опубликованной версии доступны все возможные операции над элементом экспертизы, в том числе включение и выключение. Включенные элементы экспертизы можно использовать в элементах конвейера.

При первичной публикации элемента экспертизы требуется указать его версию в формате SemVer v2.0 в поле `version`. При повторной публикации элемента экспертизы необходимо вручную увеличивать его версию. Историю изменения версий можно просмотреть в окне просмотра конфигурации опубликованного элемента экспертизы.

- **Черновик** — предварительный вариант элемента экспертизы, не предназначенный для полноценного использования в системе. Черновик позволяет изменять конфигурацию элемента экспертизы и предварительно сохранять внесенные изменения без публикации элемента и без валидации его настроек.

  В отличие от опубликованной версии, черновик нельзя включить, а следовательно, он недоступен для использования в элементах конвейера. Также для черновика недоступна возможность экспорта конфигурации.

Создание элемента экспертизы

Чтобы создать элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** (+) и выберите из выпадающего списка нужный тип элемента.

   Доступные типы элементов экспертизы:

   - Правила корреляции
   - Правила нормализации
   - Правила агрегации
   - Правила сегментации
   - Схемы активных списков
   - Схемы дашбордов
   - Схемы шаблонов отчетов
   - Таблицы обогащения
   - Шаблоны поиска
   - Глобальные функции

   Система отобразит окно создания элемента экспертизы выбранного типа.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите

требуемый пример из выпадающего списка.

3. Заполните поля элемента экспертизы, чтобы определить логику его работы.



Идентификатор и название элемента экспертизы должны быть уникальными в рамках системы.

4. Нажмите на кнопку **Опубликовать версию**. Система создаст элемент экспертизы и отобразит соответствующее уведомление. Новый элемент появится в списке.



Чтобы элемент экспертизы стал доступен для использования в системе, его необходимо включить.

Вы можете создать черновик элемента экспертизы без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик элемента и отобразит соответствующее уведомление. Новый черновик появится в списке элементов экспертизы.



Представленная инструкция применима к элементам экспертизы всех типов, кроме правил корреляции.

Декларативные правила корреляции можно создавать как через RObject-конфигурацию, так и с помощью специального конструктора. Подробности приведены в разделе Создание правила корреляции.

Изменение элемента экспертизы

Вы можете изменить элемент экспертизы через меню действий в его карточке.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.



Изменение доступно только для элементов экспертизы с типом создания **Пользовательский**.



При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.



Представленная инструкция применима к элементам экспертизы всех типов, кроме правил корреляции.

Декларативные правила корреляции можно изменять как через RObject-конфигурацию, так и с помощью специального конструктора. Подробности приведены в разделе Изменение правила корреляции.

Работа в окне настройки

При создании или изменении элемента экспертизы в окне настройки его конфигурации доступны:

- Использование горячих клавиш
- Вставка сниппетов в элемент экспертизы
- Тестирование элемента экспертизы

Для удобной работы с кодом предусмотрен полноэкранный режим. Чтобы развернуть окно на всю страницу браузера, нажмите в правом верхнем углу окна на кнопку **На весь экран** (↙↗).

Чтобы вернуть обычный режим отображения, нажмите в правом верхнем углу кнопку **Свернуть** (↘↖).



Чтобы скопировать код элемента, не открывая редактор, нажмите на кнопку действий ⋮ в верхнем правом углу карточки элемента и выберите опцию **Скопировать код**.



В VRL-программе для доступа к данным события используется символ точки (.), а к метаданным события — символ процента (%). В корреляционных событиях метаданные включают данные исходных событий.

Использование горячих клавиш

При работе в окне настройки доступно использование следующих горячих клавиш:

Функция	Клавиши и их комбинации
Сместить табуляцию на шаг вправо	TAB
Сместить табуляцию на шаг влево	SHIFT+TAB
Вызвать подсказки кода Чтобы вставить выбранную подсказку, нажмите TAB или ENTER.	CTRL+SPACE
Искать по коду	CTRL+F
Включить или выключить поиск с учетом регистра	ALT+C
Включить или выключить поиск слова целиком	ALT+W
Включить или выключить поиск с регулярными выражениями	ALT+R
Выделить текущее слово.	CTRL+D Чтобы выделить следующее идентичное значение и добавить курсор после него, удерживая CTRL, нажмите D.
Добавить курсор на следующие строки	CTRL+ALT+↓
Добавить курсор на предыдущие строки	CTRL+ALT+↑
Выделить всю строку	CTRL+L

Функция	Клавиши и их комбинации
Выделить следующую строку	SHIFT+↓
Выделить предыдущую строку	SHIFT+↑
Выделить весь код	CTRL+A
Копировать выделенное в буфер обмена	CTRL+C
Вырезать выделенное	CTRL+X
Вставить из буфера обмена	CTRL+V
Переместить текущую строку вниз	ALT+↓
Переместить текущую строку вверх	ALT+↑
Дублировать текущую строку и переместить курсор вниз	SHIFT+ALT+↓
Дублировать текущую строку и оставить курсор на текущей строке	SHIFT+ALT+↑
Отменить последнее действие	CTRL+Z
Повторить отмененное действие	CTRL+Y

Вставка сниппетов в элемент экспертизы

В окне настройки конфигурации вы можете вставить фрагмент VRL-кода из сниппета.

Чтобы вставить сниппет:

1. Установите курсор в то место программы, куда необходимо вставить сниппет.
2. Нажмите на кнопку **Сниппеты** () в левой нижней части окна. Система отобразит панель со списком доступных сниппетов. В верхней части панели расположено поле для быстрого поиска сниппетов по названию.
3. Выберите сниппет из списка. Система вставит фрагмент кода из сниппета в то место, где расположен курсор.

Тестирование элемента экспертизы

В окне настройки конфигурации вы можете запустить тесты из блока тестирования `tests`. Для этого нажмите кнопку **Запустить тесты** (▷) в левой нижней части окна.

Тесты также запускаются автоматически при создании и публикации элемента экспертизы.



Если в конфигурации элемента экспертизы имеются ошибки, их требуется предварительно исправить перед запуском тестирования или публикацией элемента. Публикация элемента с ошибками в конфигурации недоступна.

Если тесты в блоке `tests` заданы, то система запустит тестирование и отобразит в нижней части окна панель с подробным описанием результатов тестов.



Если в правиле корреляции настроена корреляция по полю с типом `DateTime`, например `timestamp`, то значение этого поля в блоке `tests` необходимо задавать в

формате `t'<DateTime-значение>'`. Например: `t'15.05.2024 06:28:15'`. Это важно при корреляции на отсутствие событий.

В противном случае при запуске тестов система обработает поле в виде строки и корреляция по дате-времени будет работать некорректно.

При необходимости вы можете свернуть панель результатов тестирования с помощью кнопки ⌄ в правой части ее заголовка. Чтобы отобразить свернутую панель, нажмите на кнопку ⌃ в правой части ее заголовка.

Обновление элемента экспертизы на конвейерах

Чтобы обновить версию правила нормализации, корреляции, агрегации или сегментации в рамках одного или нескольких конвейеров:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила в списке. Система отобразит в правой части экрана карточку с подробной информацией о правиле.
3. Откройте в карточке вкладку **Конвейеры** и нажмите на кнопку **Обновить версию** в нижней части вкладки. Отобразится дерево выбранного правила с коллекторами и конвейерами, на которых оно установлено.



Кнопка **Обновить версию** отображается, если правило установлено хотя бы на одном конвейере.

4. Выберите конвейеры, на которых необходимо обновить версию правила, установив флажки слева от их названий.



Для выбора сразу всех конвейеров установите флажок слева от варианта **Все**.

5. Нажмите на кнопку **Обновить версию**. Система запустит обновление правила на выбранных конвейерах до последней версии и отобразит окно с результатами обновления.

Вы также можете обновить версии сразу нескольких правил. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. В левом столбце таблицы установите флажки напротив тех правил, версии которых вы хотите обновить.
3. Нажмите на кнопку ⟳ на панели инструментов. Отобразится дерево выбранных правил с коллекторами и конвейерами, на которых они установлены.
4. Для каждого правила установите флажки слева от названий конвейеров, на которых необходимо обновить версию.



Для выбора всех конвейеров по конкретному правилу установите флажок слева от названия этого правила.

> Для выбора сразу всех конвейеров установите флажок слева от варианта **Все**.

5. Нажмите на кнопку **Обновить**. Система запустит обновление правил на выбранных конвейерах до последних версий и отобразит окно с результатами обновления.

Окно результатов обновления

Окно результатов обновления правил содержит таблицу со следующими столбцами:

- **Коллектор** — коллектор, в котором расположен конвейер.
- **Конвейер** — конвейер, на котором произошла попытка обновить версию правила.
- **Результат** — результат обновления: **Успешно** или **Ошибка**.
- **Ошибка** — краткое описание ошибки, если попытка обновления была неудачной.

Под таблицей отображаются поля:

- **Успешно обновлено** — количество успешно обновленных правил.
- **Ошибок обновления** — количество неудачных попыток обновления.

Просмотр элемента экспертизы

Чтобы просмотреть элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.

В верхней части карточки отображается название элемента экспертизы. Справа от названия расположена кнопка ⋮, при нажатии на которую открывается выпадающее меню с действиями, доступными для выбранного элемента экспертизы.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Если элемент экспертизы имеет неопубликованные изменения, то в верхней части карточки отображается предупреждение.

В нижней части карточки расположена кнопка просмотра или изменения элемента экспертизы:

- Если элемент экспертизы не опубликован и имеет статус черновика, то отображается кнопка **Изменить**, предназначенная для изменения его текущей конфигурации.
- Если элемент экспертизы опубликован, то отображается кнопка **Просмотр**, предназначенная для просмотра его текущей конфигурации.

Карточка элемента экспертизы содержит следующие вкладки:

- **Информация** — общие сведения об элементе.
- **Связанные элементы** — элементы экспертизы, связанные с текущим элементом.

- **Конвейеры** — конвейеры, на которых используется элемент (только для правил).

Вкладка Информация

По умолчанию при выборе элемента экспертизы открывается вкладка **Информация**. На вкладке отображается информация о выбранном элементе экспертизы.

Вкладка содержит следующие параметры:

- **Уровень угрозы** — уровень угрозы, ассоциированный с элементом экспертизы: **Низкий**, **Средний**, **Высокий**, **Критический** или **Не указан**.
- **Название** — имя элемента экспертизы, используемое для его идентификации в системе.
- **ID** — уникальный идентификационный код элемента экспертизы, генерируемый автоматически при создании элемента.
- **Тип элемента** — тип элемента экспертизы.
- **Тип создания** — тип создания элемента экспертизы: системный или пользовательский.
- **Описание** — краткое описание функциональности элемента экспертизы.
- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится элемент экспертизы.
- **Автор** — разработчик или создатель элемента экспертизы.
- **Версия** — версия элемента экспертизы, представленная в формате SemVer v2.0. Для неопубликованных элементов экспертизы (черновиков) поле не содержит значений.
- **Теги** — пользовательские теги, присвоенные элементу экспертизы в поле `tags`.
- **Источник данных** — список совместимых с элементом экспертизы источников данных, присвоенных элементу в поле `data_source`.
- **Каталог** — родительский каталог, в котором расположен элемент экспертизы.
- **Статус** — текущий статус элемента экспертизы: **Включен** или **Выключен**.
- **Дата создания** — дата и время создания элемента экспертизы.
- **Создал** — пользователь, инициировавший создание элемента экспертизы.
- **Дата изменения** — дата и время последней модификации элемента экспертизы.
- **Изменил** — пользователь, внесший последние изменения в элемент экспертизы.

Если элемент экспертизы имеет неопубликованные изменения, то в верхней части вкладки **Информация** отображается предупреждение.

Чтобы просмотреть конфигурацию элемента экспертизы:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части вкладки **Информация**. Система отобразит окно просмотра текущей конфигурации.

  В верхнем правом углу открывшегося окна отображается текущая версия элемента экспертизы. Справа от текущей версии расположена кнопка **История**, при нажатии на которую открывается выпадающий список версий элемента экспертизы. Вы можете просмотреть конфигурацию конкретной версии, выбрав ее из списка.

- Для черновика: нажмите на кнопку **Изменить** в нижней части вкладки **Информация**. Система отобразит окно редактирования текущей конфигурации.



Окно конфигурации черновика открывается только в режиме редактирования.

Вкладка Связанные элементы

Вкладка **Связанные элементы** отображает список названий элементов экспертизы, к которым обращается текущий элемент или которые обращаются к текущему элементу. Состав списка зависит от типа элемента:

- Для правил нормализации и корреляции: схемы активных списков и таблицы обогащения, к которым обращается правило.
- Для схем активных списков и таблиц обогащения: правила нормализации и корреляции, в которых используется активный список или таблица.
- Для глобальных функций:
  - схемы активных списков и таблицы обогащения, к которым обращается функция;
  - правила нормализации и корреляции, в которых используется функция.

При нажатии на название связанного элемента откроется окно просмотра его конфигурации.

Вкладка Конвейеры

Вкладка **Конвейеры** отображает список конвейеров, на которых используется выбранное правило нормализации, корреляции, агрегации или сегментации, с возможностью поиска.

Карточка каждого конвейера содержит следующие параметры:

- Название конвейера.
- Идентификатор конвейера.
- Статус конфигурации конвейера с цветовым индикатором:
  - зеленый — конфигурация установлена;
  - серый — конфигурация не установлена;
  - красный — ошибка установки.
- Идентификатор коллектора, в котором расположен конвейер.
- Название коллектора.

При нажатии на кнопку **Конфигурация конвейера** откроется диаграмма конфигурации конвейера, в элементах которого содержится выбранное правило.



Чтобы выйти из диаграммы конфигурации и вернуться на вкладку **Конвейеры** в карточке элемента экспертизы, нажмите на кнопку **Назад** в левой верхней части экрана.

Если правило установлено хотя бы на одном конвейере, в нижней части вкладки отображается кнопка **Обновить версию**, при нажатии на которую открывается окно обновления версии правила на

конвейерах.

Включение элемента экспертизы

Чтобы опубликованный элемент экспертизы стал доступен на конвейерах событий, его необходимо включить.



Для черновиков возможность включения недоступна.

Чтобы включить элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Убедитесь, что в карточке элемента экспертизы открыта вкладка **Информация**.
4. Установите переключатель состояния элемента экспертизы в верхней части карточки в активное положение. Отобразится окно подтверждения включения элемента экспертизы.
5. Нажмите на кнопку **Включить**. Система включит элемент экспертизы и отобразит соответствующее уведомление. Элемент станет доступным на конвейерах событий.

Если при включении элемента экспертизы возникли ошибки, система отобразит сообщение об этом. Для просмотра списка ошибок нажмите на кнопку **Показать**.

Выключение элемента экспертизы

Чтобы выключить опубликованный элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Убедитесь, что в карточке элемента экспертизы открыта вкладка **Информация**.
4. Установите переключатель в верхней части элемента экспертизы в неактивное положение. Отобразится окно подтверждения выключения элемента экспертизы.
5. Нажмите на кнопку **Выключить**. Система обновит статус элемента экспертизы и отобразит уведомление о выключении. Элемент экспертизы перестанет быть доступным при добавлении элементов конвейера.

Перенос элемента экспертизы

Элементы экспертизы можно переносить между каталогами.



Перенос доступен только для элементов экспертизы с типом создания **Пользовательский**.

Чтобы перенести элемент или группу элементов из списка:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите перенести, в левом столбце таблицы.
3. Нажмите на кнопку **Переместить в каталог** () на панели инструментов. Отобразится окно перемещения каталога.
4. В отобразившемся окне раскройте дерево каталогов и выберите каталог для перемещения.
5. Нажмите на кнопку **Переместить**. Система переместит элемент экспертизы в выбранный каталог и отобразит соответствующее уведомление. Элемент отобразится в выбранном каталоге.

Создание копии элемента экспертизы

Чтобы создать копию элемента экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Создать копию**. Отобразится окно настроек копии элемента экспертизы.



В окне отображаются последние сохраненные для элемента экспертизы настройки конфигурации вне зависимости от того, были ли они опубликованы.

4. Внесите требуемые изменения в конфигурацию копии элемента экспертизы.
5. Нажмите на кнопку **Опубликовать версию**. Система создаст копию элемента и отобразит соответствующее уведомление. Новый элемент экспертизы появится в списке.

   Вы можете создать копию элемента экспертизы в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Система сохранит конфигурацию для дальнейшего редактирования и отобразит соответствующее уведомление. В верхней части карточки элемента будет отображаться предупреждение о наличии неопубликованных изменений.



При создании копии элемента экспертизы требуется изменить его идентификатор в поле `id`.

Копирование кода элемента экспертизы

Чтобы скопировать код элемента экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию

Скопировать код. Система скопирует код элемента в буфер обмена и отобразит соответствующее уведомление.

Удаление элемента экспертизы

Вы можете удалить элемент экспертизы следующими способами:

- через меню действий в карточке элемента;
- через массовое удаление элементов.



Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**, не используемых в других сущностях системы.

Экспорт элементов экспертизы



Экспорт доступен только для элементов экспертизы с типом создания **Пользовательский**.

Чтобы экспортировать элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите экспортировать, в левом столбце таблицы.
3. Нажмите на кнопку **Экспортировать** на панели инструментов. Система запросит подтверждение выполнения экспорта.
4. Нажмите на кнопку **Экспортировать**. Система загрузит элемент экспертизы на устройство пользователя в виде файла с расширением .roc и отобразит соответствующее уведомление.



Неопубликованные изменения и черновики в экспорте не участвуют.



Конфигурацию таблиц обогащения также можно экспортировать из объектного хранилища NATS Object Store.

Импорт элементов экспертизы

Для загрузки элементов экспертизы в систему применяется функционал импорта, поддерживающий загрузку файлов с расширением .roc. Организация каталогов внутри ROC-файла определяет иерархию элементов экспертизы в системе после импорта. Файлы, содержащие элементы экспертизы (такие как правила и списки), должны быть в формате с расширением .roc.



- При импорте архива система автоматически создает каталоги и элементы экспертизы в соответствии со структурой каталогов в архиве.
- Если в системе уже существуют элементы с такими же названиями, как в импортируемом архиве, их содержание обновляется.
- Если в системе уже существуют элементы экспертизы с такими же ID, как в импортируемом архиве, они будут заменены на импортируемые.

- Пути к файлам в системе обновляются согласно новой конфигурации, определенной в архиве.

Чтобы импортировать элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Импортировать** на панели инструментов. Система отобразит окно импорта элементов экспертизы.
3. Нажмите **Выбрать файл**. Откроется окно проводника операционной системы.
4. В окне выберите файл для импорта и нажмите на кнопку **Открыть**. Файл отобразится в окне импорта элементов экспертизы.
5. Нажмите на кнопку **Импортировать**. Элемент экспертизы будет загружен в систему. Система отобразит окно с итогами импорта.

В случае возникновения ошибок в ходе импорта, система отобразит список ошибок с указанием ID, типа и описания ошибки.

Правила корреляции

В данном разделе представлен обзор структуры, принципов работы, а также процесса создания и расширения конфигураций правил корреляции.

О правилах корреляции

Правила корреляции в системе являются основным компонентом коррелятора. Они отвечают за обнаружение взаимосвязанных событий, которые соответствуют заданным критериям и паттернам и также могут инициировать действия после обнаружения, например, создание корреляционных событий и оповещений.

Правило корреляции представляет собой структурированный объект данных в формате RObject, который содержит конфигурацию для автоматизированного распознания определенных последовательностей обрабатываемых событий ИБ. Оно определяет параметры для фильтрации, группировки, и сопоставления событий, а также содержит логику для пост-обработки корреляционных событий, генерируемых правилом.

В системе поддерживаются два типа корреляционных правил: императивные и декларативные.

Императивные правила корреляции, основанные на алгоритмическом подходе, определяют точную последовательность операций для обработки и анализа данных. Эти правила детально регламентируют процесс, который необходимо выполнить для достижения конкретного результата.



Коллектор, настроенный для работы в режиме распределенной корреляции, не поддерживает работу императивных правил корреляции.



Поддержка императивных правил корреляции будет прекращена в ближайших версиях системы. При разработке новых правил корреляции рекомендуется использовать декларативный подход.

Декларативные правила корреляции ориентированы на описание желаемого результата без указания конкретных шагов для его достижения. Эти правила фокусируются на определении условий, при которых должен произойти определенный вывод, позволяя системе самостоятельно определять путь достижения результата. Декларативные правила обеспечивают более высокий уровень абстракции и упрощают процесс определения корреляционных связей.

Правила корреляции создаются с использованием языка преобразований и корреляции VRL, который позволяет формулировать сложные логические условия и трансформации данных для анализа и корреляции событий.

Работа с правилом корреляции

Доступные операции над правилом корреляции:

- Создание правила корреляции
- Изменение правила корреляции
- Удаление правила корреляции

Создание правила корреляции

Создать правило корреляции можно через его настройку в конструкторе или через задание его RObject-конфигурации.



Создание через конструктор доступно только для декларативных правил корреляции. Создание императивных правил выполняется только через задание их RObject-конфигурации.

Создание правила корреляции через конструктор

Конструктор правил корреляции позволяет визуально построить правило корреляции в интерактивном режиме без настройки его конфигурации.

Чтобы добавить правило корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило корреляции**. Система отобразит окно выбора способа создания правила.
3. Выберите способ создания правила через конструктор и нажмите на кнопку **Создать**. Система отобразит окно конструктора правил корреляции.
4. Заполните поля правила корреляции в конструкторе, чтобы определить логику работы правила.
5. Нажмите на кнопку **Опубликовать**. Система создаст правило корреляции и отобразит соответствующее уведомление. Новое правило отобразится в таблице раздела **Экспертиза**.



Чтобы правило корреляции стало доступно на конвейерах событий, его необходимо включить.

Вы можете создать черновик правила корреляции без публикации с помощью кнопки **Сохранить**

как черновик. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки правила будет отображаться предупреждение о наличии неопубликованных изменений.

Создание RObject-конфигурации правила корреляции

Чтобы добавить правило корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило корреляции**. Система отобразит окно выбора способа создания правила.
3. Выберите способ создания правила через код и нажмите на кнопку **Создать**. Система отобразит окно задания RObject-конфигурации правила корреляции.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

4. Заполните поля правила корреляции, чтобы определить логику его работы.
5. Нажмите на кнопку **Опубликовать версию**. Система создаст правило корреляции и отобразит соответствующее уведомление. Новое правило появится в таблице раздела **Экспертиза**.



Чтобы правило корреляции стало доступно на конвейерах событий, его необходимо включить.

Вы можете создать черновик правила корреляции без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик правила и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение правила корреляции

Изменить правило корреляции можно с помощью изменения его настроек в конструкторе или через редактирование его RObject-конфигурации.



Изменение через конструктор доступно только для декларативных правил корреляции. Изменение императивных правил выполняется только через редактирование их RObject-конфигурации.

Изменение правила корреляции в конструкторе

Конструктор правил корреляции позволяет визуально построить правило корреляции в интерактивном режиме без настройки его конфигурации.

Чтобы изменить правило корреляции в конструкторе:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах

экспертизы.

2. Нажмите на строку правила корреляции в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Изменить в конструкторе**. Система отобразит окно конструктора правил корреляции.



Вы также можете открыть окно конструктора с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек правила. Нажмите на кнопку **Изменить в конструкторе** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки. Система отобразит окно настройки конфигурации правила. Перейдите в режим конструктора с помощью переключателя в правом верхнем углу окна.

4. Внесите требуемые изменения в настройки правила корреляции в конструкторе.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию правила корреляции с измененной конфигурацией и отобразит соответствующее уведомление.



При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию правила корреляции в виде черновика без публикации с помощью кнопки **Сохранить как черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки правила будет отображаться предупреждение о наличии неопубликованных изменений.

Изменение RObject-конфигурации правила корреляции

Чтобы изменить RObject-конфигурацию правила корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила корреляции в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий ⋮ в верхнем правом углу карточки и выберите опцию **Изменить код**. Отобразится окно настроек RObject-конфигурации правила.



Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек правила. Нажмите на кнопку **Изменить в коде** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

Система отобразит окно настройки конфигурации правила.

4. Внесите требуемые изменения в конфигурацию правила корреляции.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию правила корреляции с измененной конфигурацией и отобразит соответствующее уведомление.



При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию правила корреляции в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки правила будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление правила корреляции

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Построение правила корреляции в конструкторе

Конструктор правил корреляции позволяет визуально построить правило корреляции в интерактивном режиме без необходимости ручной настройки его RObject-конфигурации. Конструктор автоматически создает правило нужной конфигурации на основе заданных настроек.



Построение через конструктор доступно только для декларативных правил корреляции. Создание или изменение императивных правил выполняется только через настройку их RObject-конфигурации.

Построение правила корреляции с помощью конструктора состоит из следующих этапов:

1. Заполнение общей информации о правиле
2. Задание переменных
3. Настройка фильтрации событий
4. Настройка объединения событий
5. Настройка корреляционных событий
6. Настройка тестов
7. Сохранение правила

Заполнение общей информации о правиле

На вкладке **Общая информация** в конструкторе заполняются поля общей информации о правиле корреляции.

Чтобы задать общую информацию о правиле, заполните следующие поля:

1. **ID** — введите уникальный идентификатор правила корреляции. Указанное значение будет

сохранено в поле `id` в RObject-конфигурации правила.

2. **Версия** — введите версию правила корреляции в формате Semantic Versioning. Указанное значение будет сохранено в поле `version` в RObject-конфигурации правила.
3. **Название** — введите название правила корреляции. Указанное значение будет сохранено в поле `name` в RObject-конфигурации правила.
4. **Описание** (опционально) — введите описание правила корреляции, содержащее дополнительную информацию о его назначении и функциональности. Указанное значение будет сохранено в поле `description` в RObject-конфигурации правила.
5. **Автор** (опционально) — введите имя и контактные данные автора правила корреляции. Указанное значение будет сохранено в поле `author` в RObject-конфигурации правила.
6. **Статус** (опционально) — введите статус разработки варианта правила. Указанное значение будет сохранено в поле `status` в RObject-конфигурации правила.
7. **Дата** — при необходимости выберите другую дату создания или изменения правила. По умолчанию установлена текущая дата. Указанное значение будет сохранено в поле `date` в RObject-конфигурации правила.
8. **Уровень угрозы** — выберите уровень угрозы корреляционного события, испускаемого правилом корреляции. Доступные значения:
   - Критический (`critical`);
   - Высокий (`high`);
   - Средний (`medium`);
   - Низкий (`low`).

   Выбранное значение будет сохранено в поле `severity` в RObject-конфигурации правила.
9. **Источники данных** (опционально) — введите список источников данных, с которыми совместимо правило корреляции. Данный список используется для классификации и быстрого поиска правил в системе. Указанное значение будет сохранено в поле `data_source` в RObject-конфигурации правила.
10. **Референсы** (опционально) — введите ссылки на подробный материал. Указанное значение будет сохранено в поле `reference` в RObject-конфигурации правила.
11. **Известные ложные срабатывания** (опционально) — введите возможные ложноположительные срабатывания. Указанное значение будет сохранено в поле `known_false_positives` в RObject-конфигурации правила.
12. **Теги** (опционально) — введите список тегов для правила корреляции. Указанное значение будет сохранено в поле `tags` в RObject-конфигурации правила.
13. **Максимальное количество корреляционных окон** (опционально) — введите максимально допустимое количество корреляционных окон, которое можно запустить одновременно. При превышении система отключает правило корреляции. Указанное значение будет сохранено в поле `max_correlation_windows` в RObject-конфигурации правила.
14. **Ограничения генерации корреляционных событий** (опционально) — установите переключатель в активное положение, чтобы настроить параметры для ограничения генерации корреляционных событий. При превышении ограничений система отключает правило корреляции. Указанные значения будут сохранены в полях группы `rate_limit` в RObject-конфигурации правила.

Доступны следующие параметры:

- **Количество событий** — максимально допустимое количество корреляционных событий, генерируемых за указанный период времени. Соответствует полю `max_rate` группы `rate_limit` в RObject-конфигурации правила.
- **Интервал для генерации событий** — промежуток времени в секундах, в течение которого допустима генерация не более указанного количества корреляционных событий. Соответствует полю `duration_secs` группы `rate_limit` в RObject-конфигурации правила.
- **Время ожидания восстановления работы правила** — промежуток времени в секундах, в течение которого ожидается восстановление нормального режима работы правила. Если режим работы правила не восстановится, система отключит его. Соответствует полю `grace_period_secs` группы `rate_limit` в RObject-конфигурации правила.



Пусть установлены следующие ограничения генерации корреляционных событий:

- **Количество событий** — 10.
- **Интервал для генерации событий** — 60 сек.
- **Время ожидания восстановления работы правила** — 60 сек.

Предположим, правило сгенерировало более 10 событий за 60 секунд. Если за последующие 60 секунд также будет сгенерировано более 10 событий, система отключит правило корреляции.

Задание переменных

На вкладке **Переменные** в конструкторе можно задать переменные — выражения VRL, которые могут использоваться для вычисления значений в других разделах конструктора.

Чтобы задать переменную:

1. Нажмите на кнопку **Добавить переменную**.
2. Введите название переменной из латинских букв и цифр. Первый символ в названии должен быть буквой.
3. Введите выражение на языке VRL.

Чтобы удалить переменную, нажмите на кнопку  в строке ее заголовка.

Настройка фильтрации событий

На вкладке **Фильтры событий** в конструкторе настраивается фильтр, определяющий условия, при которых правило корреляции будет применяться к событиям.

Чтобы настроить фильтрацию событий:

1. Добавьте псевдонимы событий.
2. Задайте критерии фильтрации событий.
3. При необходимости настройте группировку событий для псевдонимов.

Настройка псевдонимов событий

Псевдонимы потоков событий используются для их идентификации в рамках правила корреляции, а также для их разделения и фильтрации. В RObject-конфигурации правила корреляции настройки псевдонимов задаются в поле `aliases`.

Чтобы добавить псевдоним событий в конструкторе:

1. Нажмите на кнопку **Добавить событие** (+). Отобразятся настройки псевдонима событий.
2. При необходимости измените название псевдонима. По умолчанию псевдонимы создаются с названием "event_<номер псевдонима>".
3. Задайте критерии фильтрации событий.
4. При необходимости настройте группировку событий для псевдонима.

Чтобы удалить псевдоним событий, нажмите на кнопку в строке его заголовка.



На вкладке **Фильтры событий** должен быть задан хотя бы один псевдоним. Удаление последнего псевдонима недоступно.

Настройка фильтрации

Фильтрация может задаваться как для конкретных псевдонимов событий, так и для всего потока событий:

- Фильтрация псевдонима настраивается с помощью условий, задаваемых в области настройки псевдонима. Для каждого псевдонима должно быть задано хотя бы одно условие фильтрации.
- Фильтрация всего потока событий выполняется с помощью общего фильтра. Чтобы включить общий фильтр, необходимо установить переключатель **Общий фильтр** в активное положение. В RObject-конфигурации правила корреляции общий фильтр задается в поле `filter`.

Чтобы добавить условие фильтрации для псевдонима событий или общего фильтра:

1. Добавьте поля настроек условия. Способ добавления настроек отличается в зависимости от того, является ли добавляемое условие первым в псевдониме или общем фильтре:
   - Если вы добавляете первое условие, нажмите на кнопку **Добавить условие** (+) и выберите тип условия. Отобразятся поля настроек выбранного типа условия.
   - Если псевдоним или общий фильтр уже содержит условия, новые условия добавляются с использованием логической связи с существующими условиями. Чтобы добавить условие с логической связью, нажмите на кнопку необходимой логической связи в строке нужного условия и выберите тип условия.

     Доступные кнопки логической связи:

     - **И** (⊕) — позволяет добавить новое условие к выбранному условию и связать их через логическое "И".

       Если выбранное условие связано с другими условиями на том же уровне через логическое "ИЛИ", то новое условие создаст подгруппу вместе с выбранным условием. Условия внутри созданной подгруппы будут связаны через логическое "И". Сама группа

будет связана с условиями на своем уровне через логическое "ИЛИ".

Если выбранное условие связано с другими условиями на том же уровне через логическое "И", то новое условие будет добавлено на тот же уровень без создания подгруппы.

- **ИЛИ** (⊕) — позволяет добавить новое условие к выбранному условию и связать их через логическое "ИЛИ".

  Если выбранное условие связано с другими условиями на том же уровне через логическое "И", то новое условие создаст подгруппу вместе с выбранным условием. Условия внутри созданной подгруппы будут связаны через логическое "ИЛИ". Сама группа будет связана с условиями на своем уровне через логическое "И".

  Если выбранное условие связано с другими условиями на том же уровне через логическое "ИЛИ", то новое условие будет добавлено на тот же уровень без создания подгруппы.



Нельзя добавить более трех уровней условий. Добавление новых условий к группе на третьем уровне выполняется через тип связи, уже установленный в этой группе. Например, если в группе на третьем уровне условия связаны между собой через логическое "И", то новое условие в этой группе может быть добавлено только со связью "И".

2. Задайте настройки условия. Настройки отличаются в зависимости от типа условия.



При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы удалить условие фильтрации, нажмите на кнопку 🗑 в его строке.



Для каждого псевдонима событий должно быть задано хотя бы одно условие фильтрации.

Настройка условий разных типов

Доступные типы условий фильтрации:

- Сравнение полей
- Запрос в активный список
- Запрос в таблицу обогащения
- Глобальная функция
- Блок кода

Настройка условия Сравнение полей

Условие типа **Сравнение полей** позволяет сравнивать значение поля в событии со значением другого поля или с каким-либо заданным значением.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Сравнение полей**:

1. Выберите из выпадающего списка поле события.

На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

2. При необходимости выберите из выпадающего списка тип данных выбранного поля.

Изменение типа данных для полей универсальной модели события и универсальной модели 2.0 позволяет переопределить их системный тип данных в рамках заданного условия.

3. Выберите оператор сравнения из выпадающего списка:
   - `=` — равно.
   - `!=` — не равно.
   - `Like` (только для строк и полей `JSON`) — поиск строк, содержащих заданное значение.

Система ищет точное вхождение значения, символы подстановки не поддерживаются.

Заданное значение должно быть строкой, числом или логическим.

Чтобы проверить вхождение объекта или массива, задайте его как строку одним из способов:

- Укажите в значении строковое представление в формате JSON. Поля в представлении объекта должны быть упорядочены по алфавиту, иначе фильтр не сработает.
- Создайте переменную, где значение преобразуется в строку с помощью функции `encode_json`, и укажите переменную как значение.

- `In` (только для строк) — поиск подстрок. Строка, в которой будет осуществляться поиск, должна быть указана в формате JSON.
- `=null` — не имеет значений (пустое поле).
- `!=null` — имеет значение (непустое поле).

4. Если на предыдущем шаге был выбран оператор сравнения, отличный от `=null` и `!=null`, то необходимо задать значение поля. Для этого выберите тип значения из выпадающего списка:
   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.

     Значение исходного поля будет сравниваться с указанным значением.

     Если выбран оператор `In`, экранировать специальные символы не требуется, за исключением двойных кавычек внутри строковых значений.

     Например, для строки `["a", "b"]` дополнительное экранирование не требуется, а для строки `["a", "\"b"\"]` требуется экранировать кавычки, в которые заключен символ `b`.

   - **Значение из поля** — потребуется выбрать поле события из выпадающего списка.

     На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

     Чтобы вручную добавить поле в выпадающий список:

     i. Введите название поля.

     ii. Выберите опцию **Создать**.

     Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

     Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

     Значение исходного поля будет сравниваться со значением указанного поля.

   - **Переменная** — потребуется выбрать одну из добавленных ранее переменных.

     Значение исходного поля будет сравниваться с вычисленным значением переменной.

Настройка условия Запрос в активный список

Условие типа **Запрос в активный список** позволяет дополнить поток событий в псевдониме записями активного списка, которые могут использоваться для обогащения корреляционных событий.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Запрос в активный список**:

1. Выберите нужный активный список из выпадающего списка. В области настроек условия отобразятся все поля ключей выбранного активного списка.
2. Выберите условие из выпадающего списка:
   - **Есть запись, для которой выполняются все условия** — условие истинно, если в активном списке существует запись, удовлетворяющая всем заданным условиям.
   - **Нет записей, для которых выполняются все условия** — условие истинно, если в активном списке нет записей, удовлетворяющих всем заданным условиям.
3. Для каждого поля ключа активного списка выберите тип значения из выпадающего списка:
   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.

     Значение поля ключа будет сравниваться с указанным значением.
   - **Значение из поля** — потребуется выбрать поле события из выпадающего списка.

     На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

     Чтобы вручную добавить поле в выпадающий список:

     i. Введите название поля.

     ii. Выберите опцию **Создать**.

     Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

     Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

     Значение поля ключа будет сравниваться со значением указанного поля события.
   - **Переменная** — потребуется выбрать одну из добавленных ранее переменных.

     Значение поля ключа будет сравниваться с вычисленным значением переменной.

Настройка условия Запрос в таблицу обогащения

Условие типа **Запрос в таблицу обогащения** позволяет дополнить поток событий в псевдониме записями таблицы обогащения, которые могут использоваться для обогащения корреляционных событий.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Запрос в таблицу обогащения**:

1. Выберите нужную таблицу обогащения из выпадающего списка.
2. Выберите условие из выпадающего списка:
   - **Есть запись, для которой выполняются все условия** — условие истинно, если в таблице обогащения существует запись, удовлетворяющая всем заданным условиям.
   - **Нет записей, для которых выполняются все условия** — условие истинно, если в таблице обогащения нет записей, удовлетворяющих всем заданным условиям.
3. Добавьте условия для таблицы обогащения.

   Чтобы добавить условие:

   a. Нажмите на кнопку **Добавить поле** (＋). Добавится строка настройки условия.

   По умолчанию при выборе таблицы обогащения уже отображается одна строка настройки условия.

   b. Выберите из выпадающего списка поле таблицы обогащения, для которого будет задаваться условие.

   c. Выберите тип значения поля из выпадающего списка:

   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.

     Значение поля таблицы будет сравниваться с указанным значением.

   - **Значение из поля** — потребуется выбрать поле события из выпадающего списка.

     На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

     Чтобы вручную добавить поле в выпадающий список:

     i. Введите название поля.

     ii. Выберите опцию **Создать**.

     Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

     Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

     Значение поля таблицы будет сравниваться со значением указанного поля события.

- **Переменная** — потребуется выбрать одну из добавленных ранее переменных.

Значение поля таблицы будет сравниваться с вычисленным значением переменной.

Чтобы удалить условие, нажмите на кнопку в его строке.



Для таблицы обогащения должно быть задано хотя бы одно условие. Удаление последнего условия недоступно.

Настройка условия Глобальная функция

Условие типа **Глобальная функция** позволяет использовать глобальные функции для фильтрации событий.



При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Глобальная функция**:

1. Выберите нужную глобальную функцию из выпадающего списка. В области настроек условия отобразятся все параметры выбранной глобальной функции.
2. Для каждого параметра глобальной функции выберите тип значения из выпадающего списка:
   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных параметра.
   - **Значение из поля** — потребуется выбрать поле события из выпадающего списка.



На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

   - **Переменная** — потребуется выбрать одну из добавленных ранее переменных.

Настройка условия Блок кода

Условие типа **Блок кода** позволяет задать условие в виде кода на языке VRL.



При сохранении правила корреляции через конструктор все заданные условия

преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Блок кода**, задайте VRL-код условия.



В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

Настройка группировки событий

Группировка позволяет объединять события в группы. В RObject-конфигурации правила корреляции группировка задается в поле `group`.

Чтобы добавить группировку:

1. Нажмите на кнопку **Добавить группировку** (+). Отобразятся настройки группировки.
2. Задайте количество событий в группе вручную или с помощью кнопок — и +. Допустимый диапазон: 2—100.
3. Выберите нужные поля события из выпадающего списка.



На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

4. Если необходимо учитывать только уникальные события в рамках указанных полей группировки, установите флажок **Уникальность событий в рамках группировки**.



Вы можете добавить не более двух группировок. При этом одна группировка должна быть с уникальностью событий, другая — без уникальности событий.

Чтобы удалить группировку, нажмите на кнопку 🗑 в строке ее заголовка.

Настройка объединения событий

На вкладке **Объединение** задаются правила объединения двух или более потоков событий. В RObject-конфигурации правила корреляции настройки объединения задаются в поле `select`.

Чтобы настроить объединения:

1. Укажите время действия (TTL) корреляционного окна в секундах. В RObject-конфигурации правила корреляции этой настройке соответствует поле `ttl`.
2. Если на вкладке **Фильтры событий** добавлено более одного псевдонима событий:
   a. Выберите из выпадающего списка один из псевдонимов.
   b. Добавьте объединение для выбранного псевдонима событий.
   c. Если на вкладке **Фильтры событий** добавлено более двух псевдонимов событий, добавьте объединения для оставшихся псевдонимов.

Настройка объединений



Добавление объединений недоступно, если на вкладке **Фильтры событий** добавлен только один псевдоним событий.

В RObject-конфигурации правила корреляции настройка объединения выполняется в поле `join`.

Чтобы добавить объединение:

1. Нажмите на кнопку **Добавить объединение** (+). Отобразятся настройки объединения.
2. Выберите тип объединения для определения условий формирования корреляционного события:
   - **Получение события** — за указанный таймаут должно быть получено соответствующее событие. В RObject-конфигурации правила корреляции этой настройке соответствует поле `absent` со значением `false`.
   - **Отсутствие события** — за указанный таймаут не должно быть получено соответствующее событие. В RObject-конфигурации правила корреляции этой настройке соответствует поле `absent` со значением `true`.
3. Введите время ожидания (таймаут) получения соответствующего события в секундах. В RObject-конфигурации правила корреляции этой настройке соответствует поле `wait_for`.
4. Выберите из выпадающего списка один из псевдонимов событий, добавленных на вкладке **Фильтры событий**.



На выбор доступны только те псевдонимы, которые ранее не использовались на вкладке **Объединение**.

5. Добавьте условия сопоставления полей.

Чтобы удалить объединение, нажмите на кнопку  в строке заголовка объединения.

Настройка сопоставления полей

Сопоставление полей позволяет сравнить определенные поля в событии исходного псевдонима и в событии другого псевдонима. В зависимости от итогов этого сравнения за указанный таймаут и от заданных условий формирования корреляционного события, по итогам выполнения корреляции будет создаваться корреляционное событие.

Чтобы добавить условие сопоставления полей в объединении:

1. Нажмите на кнопку **Добавить условие** (+). Отобразится строка для задания условия

сопоставления полей.

2. Выберите из выпадающего списка псевдоним, поля событий которого требуется сопоставлять с полями событий исходного псевдонима.



- На выбор доступны только те псевдонимы, которые ранее уже использовались на вкладке **Объединение** для других объединений.
- Для первого объединения автоматически устанавливается псевдоним, выбранный в верхней части вкладки **Объединение**.

3. Выберите из выпадающего списка поле события.



На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

Указанные поля в событиях выбранных псевдонимов будут участвовать в сопоставлении полей.

Чтобы удалить условие сопоставления полей в объединении, нажмите на кнопку в строке условия.



В объединении должно быть хотя бы одно условие сопоставления полей. Удаление последнего условия в объединении недоступно.

Настройка корреляционных событий

На вкладке **Настройка события корреляции** определяются действия, которые должны выполняться при успешной корреляции событий, настраиваются правила формирования корреляционных событий, а также задаются прочие дополнительные условия корреляции. В RObject-конфигурации правила корреляции эти настройки задаются в поле `on_correlate`.

Чтобы настроить поведение корреляционных событий:

1. При необходимости введите время троттлинга в секундах. Время троттлинга определяет временной интервал, который указывает, как часто могут генерироваться корреляционные события для каждой группы, заданной на вкладке **Объединение**. Если групп нет, время троттлинга указывается глобально на все правило. Указанное значение будет сохранено в поле `throttle_time_sec` в RObject-конфигурации правила.

2. Определите, должны ли создаваться оповещения по результатам корреляции:
    - **Включено** — оповещение будет создаваться при каждом успешном случае корреляции, соответствующем условиям правила. В RObject-конфигурации правила этой настройке соответствует поле `trigger_alert` со значением `true`.
    - **Выключено** — оповещения создаваться не будут, даже если корреляция происходит. В RObject-конфигурации правила этой настройке соответствует поле `trigger_alert` со значением `false`.
3. Сформируйте список полей корреляционного события.
4. При необходимости введите правила формирования текста сообщения в корреляционном правиле на языке VRL.
5. При необходимости введите дополнительные условия корреляции на языке VRL.



При сохранении правила корреляции через конструктор все заданные настройки корреляции преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Формирование полей корреляционного события

Чтобы добавить поле в корреляционное событие:

1. Нажмите на кнопку **Добавить поле** (+). Отобразится строка настройки нового поля.
2. Выберите из выпадающего списка поле события.



На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

3. Выберите из выпадающего списка тип источника, из которого поле будет брать значение.
4. Задайте настройки источника. Настройки отличаются в зависимости от типа источника.

Чтобы удалить поле, нажмите на кнопку в строке его настроек.

Настройка источников разных типов

Доступные типы источников:

- Значение
- Значение из поля
- Запрос в активный список
- Запрос в таблицу обогащения
- Переменная

Настройка источника типа Значение

Источник типа **Значение** позволяет задать определенное значение, которое будет присвоено полю корреляционного события.

Чтобы настроить источник типа **Значение**, укажите значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.

Настройка источника типа Значение из поля

Источник типа **Значение из поля** позволяет задать поле события, значение которого будет использоваться для значения поля в корреляционном событии.

Чтобы настроить источник типа **Значение из поля**:

1. Выберите из выпадающего списка псевдоним события. На выбор доступны псевдонимы, заданные на вкладке **Фильтры событий**.
2. Выберите из выпадающего списка поле события.



На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

Настройка источника типа Переменная

Источник типа **Переменная** позволяет задать выражение на языке VRL, результат которого будет присвоен полю корреляционного события.

Чтобы настроить источник типа **Переменная**, выберите из выпадающего списка название переменной. На выбор доступны переменные, заданные на вкладке Переменные.

Настройка источника типа Запрос в активный список

Источник типа **Запрос в активный список** позволяет найти запись в активном списке и указать поле записи, значение которого будет использоваться для значения поля в корреляционном событии.

Чтобы настроить источник типа **Запрос в активный список**:

1. Выберите нужный активный список из выпадающего списка. В области настроек источника отобразятся все поля ключей выбранного активного списка.
2. Для каждого поля ключа активного списка выберите тип значения из выпадающего списка:
   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.
   - **Значение из поля** — потребуется выбрать псевдоним события. На выбор доступны псевдонимы, заданные на вкладке **Фильтры событий**.

     Затем необходимо выбрать поле события из выпадающего списка.

     На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

     Чтобы вручную добавить поле в выпадающий список:

     i. Введите название поля.

     ii. Выберите опцию **Создать**.

     Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

     Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

     Значение выбранного поля в выбранном событии будет использоваться в качестве значения поля активного списка.

   - **Переменная** — потребуется выбрать название переменной. На выбор доступны переменные, заданные на вкладке **Переменные**.
3. Выберите из выпадающего списка поле активного списка, значение которого будет использоваться для значения поля в корреляционном событии.

Настройка источника типа Запрос в таблицу обогащения

Источник типа **Запрос в таблицу обогащения** позволяет найти запись в таблице обогащения и указать поле записи, значение которого будет использоваться для значения поля в корреляционном событии.

1. Выберите нужную таблицу обогащения из выпадающего списка.

2. Добавьте поля таблицы обогащения, по которым будет осуществляться поиск записи в таблице.

   Чтобы добавить поле:

   a. Нажмите на кнопку **Добавить поле** (+). Добавится строка настройки поля.

   b. Выберите из выпадающего списка поле таблицы обогащения.

   c. Выберите тип значения поля из выпадающего списка:

   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.
   - **Значение из поля** — потребуется выбрать псевдоним события. На выбор доступны псевдонимы, заданные на вкладке **Фильтры событий**.

     Затем необходимо выбрать поле события из выпадающего списка.

     На выбор доступны только поля универсальной модели события и универсальной модели 2.0, а также поля, ранее добавленные вручную.

     Чтобы вручную добавить поле в выпадающий список:

     i. Введите название поля.

     ii. Выберите опцию **Создать**.

     Чтобы обратиться ко вложенному полю внутри поля типа `JSON`, используйте последовательность атрибутов, разделенных точкой. Например, `data.network.src_ip`.

     Чтобы в карточке события отображались разобранные и структурированные данные JSON, их нужно обработать на конвейере. Например, используйте VRL-трансформацию, которая содержит пример JSON с нужной структурой.

     Значение выбранного поля в выбранном событии будет использоваться в качестве значения поля таблицы обогащения.

   - **Переменная** — потребуется выбрать название переменной. На выбор доступны переменные, заданные на вкладке **Переменные**.

   Чтобы удалить поле, нажмите на кнопку в его строке.

   Для таблицы обогащения должно быть задано хотя бы одно поле. Удаление последнего поля недоступно.

3. Выберите из выпадающего списка поле таблицы обогащения, значение которого будет использоваться для значения поля в корреляционном событии.

Настройка тестов

На вкладке **Тесты** опционально задаются тесты для проверки и верификации правила корреляции. Они помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Тесты, заданные на вкладке **Тесты**, заполняют блоки `tests` и `setup_tests` в RObject-конфигурации правила корреляции.

Чтобы добавить тесты:

1. Введите набор тестов на языке VRL.
2. Нажмите на кнопку **Запустить** для запуска выполнения программы.

   Если выполнение программы прошло успешно, обработанное событие отобразится на панели **Результат**.



Если в элементе экспертизы используются внешние зависимости, то есть активные списки, таблицы обогащения или глобальные функции, обязательно заполните поле `setup_tests`.

Сохранение правила

Вы можете сохранить созданное правило корреляции по одной из следующих кнопок в правом нижнем углу конструктора:

- **Опубликовать** — правило будет сохранено и опубликовано. Система создаст правило корреляции и отобразит соответствующее уведомление. Новое правило отобразится в таблице раздела **Экспертиза**. Опубликованные правила можно включить и использовать в конвейерах событий.
- **Сохранить как черновик** — правило будет сохранено в виде черновика без публикации. Система создаст черновик правила корреляции и отобразит соответствующее уведомление. Черновик отобразится в таблице раздела **Экспертиза**. В отличие от опубликованных версий, черновики не могут быть использованы в конвейерах событий.

Императивные правила

В данном разделе представлен обзор структуры, принципа работы, процесса создания и расширения конфигурации императивного правила корреляции. Управление правилами корреляции в системе осуществляется в разделе **Экспертиза**.



Поддержка императивных правил корреляции будет прекращена в ближайших версиях системы. При разработке новых правил корреляции рекомендуется использовать декларативный подход.

Об императивных правилах корреляции

Императивное правило корреляции в системе представляет собой объект данных в формате RObject (.ro), содержащий детально определенные команды и процедуры для анализа и корреляции событий ИБ. Такие правила характеризуются точной спецификацией действий, определенным порядком операций, контрольными структурами и явным управлением состоянием.



Коллектор, настроенный для работы в режиме распределенной корреляции, не поддерживает работу императивных правил корреляции.

Императивное правило корреляции состоит из набора полей, которые можно разделить на два типа:

- Набор общих полей, которые предоставляют основную информацию о правиле и его назначении, задают контекст для корреляции и определяют основные характеристики правила. Этот набор общих полей остается неизменным независимо от типа корреляционного правила.
- Набор дополнительных полей, который изменяется в зависимости от выбранного режима работы окна корреляции.

Императивные правила корреляции поддерживают три режима работы окна корреляции (`window_mode`):

- Параллельный (`parallel`) — это режим работы корреляционного окна, при котором создаваемые правилом окна могут открываться друг за другом и работать одновременно, накладываясь друг на друга во времени. Таким образом, одно событие может присутствовать сразу в нескольких окнах одновременно. Этот метод позволяет системе обнаруживать сложные атаки и аномалии, так как события могут анализироваться одновременно в рамках различных временных контекстов.



Если поле `start_when` не указано, окно будет открываться для каждого события, прошедшего через фильтр. В таком случае событие также будет добавлено в каждое открытое корреляционное окно.

- Без перекрывания (`non-overlapping`) — это режим работы корреляционного окна, при котором правило имеет одновременно не более одного активного окна, таким образом открываемые правилом окна не пересекаются во времени. Окно создается при получении первого события, которое удовлетворяет условиям открытия окна. Последующее окно может быть открыто только после закрытия текущего окна. Таким образом, каждое событие может находиться не более чем в одном окне одновременно. Этот подход предотвращает дублирование анализа событий и обеспечивает предсказуемое распределение ресурсов для обработки данных.
- Без состояния (`stateless`) — это режим работы корреляционного окна, при котором создаваемые правилом окна хранят данные только об одном событии. На каждое событие, удовлетворившее условию, создается индивидуальное корреляционное окно, которое сразу закрывается, генерируя корреляционное событие. Таким образом, каждое событие анализируется независимо от предыдущего, и их контекст не учитывается при анализе. Этот режим обеспечивает простоту реализации и минимальную нагрузку на систему, но ограничивает способность обнаружения сложных атак и аномалий, требующих анализа последовательности событий в контексте периода времени.

Правила корреляции создаются с использованием языка преобразований и корреляции VRL, который позволяет формулировать сложные логические условия и трансформации данных для анализа и корреляции событий.

Режим работы окна корреляции определяется в поле `window_mode`. Каждое правило может функционировать только в одном выбранном режиме. Если режим не задан, система ожидает декларативный подход к написанию правила.

Структура правила корреляции

Правила корреляции содержат набор обязательных и опциональных полей. Состав полей и логика их работы отличаются в зависимости от заданного режима работы корреляционного окна.

Общие поля

Поля верхнего уровня:

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила корреляции.	строка	да
`name`	Название правила, которое обозначает его цель или функцию.	строка	да
`version`	Версия правила корреляции, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора правила корреляции.	строка	нет
`window_mode`	Режим окна корреляции, определяющий каким образом система будет обрабатывать корреляционные события. Доступные режимы: `parallel`, `non-overlapping`, `stateless`. Если значение не задано, система ожидает декларативный подход к написанию правила.	строка	да
`description`	Описание правила корреляции, где можно указать дополнительную информацию о его назначении и функциональности.	строка	нет
`data_source`	Список источников данных, с которыми совместимо правило корреляции. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных правилу корреляции.	массив строк	нет
`group_by`	События группируются по этим полям и каждая группа коррелируется независимо. Значения полей группы автоматически попадают в поля корреляционного события.	массив строк	нет
`type`	Тип правила. Всегда имеет значение `correlation_rule` для правила этого типа.	строка	да
`severity`	Уровень угрозы корреляционного события, испускаемого корреляционным правилом. Доступны значения: `low`, `medium`, `high`, `critical`.	строка	да
`filter`	Блок правил на языке VRL, определяющий условия (фильтр), при которых правило корреляции будет применяться к событиям.	строка	нет
`on_correlation`	Блок правил на языке VRL, определяющий логику обработки корреляционного события после его создания.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`throttle_time_sec`	Указывает интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group_by`.	целое число	нет
`tests`	Блок тестирования правила корреляции, содержащий массив тестовых событий и условия для проверки срабатывания правила. Обязательные поля: `name`, `events`, `assertion`.	объект	нет

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста. Позволяет идентифицировать конкретный тест в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, поступающих в тестируемый элемент конвейера.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Дополнительные поля режима `parallel`

Поле	Описание	Тип данных	Обязательное поле
`on_event`	Блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий.	строка	да
`correlate_when`	Блок правил на языке VRL, определяющий условие закрытия корреляционного окна и генерации корреляционного события.	строка	да
`start_when`	Блок правил на языке VRL, определяющий условие открытия корреляционного окна. Если условие не задано, новое окно будет открываться на каждое событие, прошедшее через фильтр.	строка	нет
`expire_after_sec`	Время жизни корреляционного окна в секундах, если не происходит обновление его состояния.	целое число	нет
`windows_limit`	Максимально допустимое количество корреляционных окон, которое можно открыть одновременно.	целое число	нет

Дополнительные поля режима `non-overlapping`

Поле	Описание	Тип данных	Обязательное поле
`on_event`	Блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий.	строка	да
`correlate_when`	Блок правил на языке VRL, определяющий условие закрытия корреляционного окна и испускания корреляционного события.	строка	да
`expire_after_sec`	Указывает время автоматического закрытия корреляционного окна (по умолчанию 60 сек).	целое число	нет

Дополнительные поля режима `stateless`

Поле	Описание	Тип данных	Обязательное поле
`on_event`	Блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий. Если событие не обновляет окно, необходимо вызвать инструкцию `abort`.	строка	нет
`correlate_when`	Блок правил на языке VRL, определяющий условие закрытия корреляционного окна и испускания корреляционного события.	строка	да

Составление правила

Чтобы составить правило корреляции:

1. Заполните общие поля метаданных корреляционного правила:
   - `id`: задайте уникальный идентификатор правила.
   - `name`: задайте название правила.
   - `window_mode` (обязательное): определите режим работы корреляционного окна (`parallel`, `non-overlapping` или `stateless`).
   - `type` (обязательное): укажите `correlation_rule` в поле, чтобы определить тип правила.
   - `severity` (обязательное): задайте уровень угрозы испускаемого события. Доступны значения: `low`, `medium`, `high`, `critical`.
   - `version`: укажите версию правила в формате Semantic Versioning.
   - `author` (опциональное): укажите имя и контактные данные автора.
   - `description` (опциональное): предоставьте краткое описание назначения и функциональности правила.
   - `tags` (опциональное): добавьте теги для классификации и поиска правила.
2. Заполните общие поля корреляционного правила:
   - `filter` (опциональное): создайте блок правил на языке VRL, который определит, к каким событиям будет применяться правило.
   - `group_by` (опциональное): определите поля, по которым будет производиться группировка событий для корреляции.
   - `on_correlation` (опциональное): создайте блок правил на языке VRL, которые необходимо выполнить после корреляции событий.
   - `throttle_time_sec` (опциональное): установите интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group_by`.
3. Заполните специфические поля в зависимости от выбранного режима корреляции (`window_mode`):

   Для правила `parallel`:

- `start_when` (опциональное): создайте блок правил на языке VRL, определяющий условие открытия корреляционного окна. Если условие не задано, новое окно будет открываться на каждое событие прошедшее через фильтр.
- `on_event` (обязательное): создайте блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий. Если событие не обновляет окно, необходимо вызвать инструкцию `abort`.
- `correlate_when` (обязательное): создайте блок правил на языке VRL, определяющий условие закрытия корреляционного окна и генерации корреляционного события. Если условие удовлетворено, окно будет закрыто, и по нему будет сформировано корреляционное событие.
- `expire_after_sec` (обязательное): задайте период времени, по истечении которого корреляционное окно будет закрыто автоматически.
- `windows_limit` (опциональное): ограничьте количество корреляционных окон, которые можно открыть правилом (по умолчанию 100 000).

Для правила `non-overlapping`:

- `on_event` (обязательное): создайте блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий. Если событие не обновляет окно, необходимо вызвать инструкцию `abort`.
- `correlate_when` (обязательное): создайте блок правил на языке VRL, определяющий условие закрытия корреляционного окна и генерации корреляционного события. Если условие удовлетворено, окно будет закрыто, и по нему будет сформировано корреляционное событие.
- `expire_after_sec` (обязательное): задайте период времени, по истечении которого корреляционное окно будет закрыто автоматически.

Для правила `stateless`:

- `on_event` (опциональное): создайте блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий. Если событие не обновляет окно, необходимо вызвать инструкцию `abort`.
- `correlate_when` (обязательное): создайте блок правил на языке VRL, определяющий условие закрытия корреляционного окна и генерации корреляционного события. Если условие удовлетворено, окно будет закрыто, и по нему будет сформировано корреляционное событие.

4. Создайте блок условий тестирования работы правила корреляции в поле `tests`.

Тестирование правила

Тесты в правиле корреляции позволяют проверить корректность работы правила и его способность реагировать на заданные типы событий.

Тестирование правила корреляции гарантирует, что правило корректно обрабатывает данные, точно идентифицирует события и правильно реагирует на них, обеспечивая надежную работу системы.

В случае, если правило корреляции использует данные из активных списков, необходимо инициализировать искусственные данные через поле `setup_tests`. Это позволит тестировать правило в условиях, максимально приближенных к реальной эксплуатации, без необходимости доступа к продуктивным данным.

Поле `setup_tests`



Для тестирования правил, которые используют внешние зависимости (например, взаимодействуют с активными списками или таблицами обогащения), использование поля `setup_tests` является обязательным.

Поле `setup_tests` в конфигурации правила корреляции предназначено для определения тестового окружения, включая предварительную настройку искусственных данных для активных списков и таблиц обогащения, используемых в правиле. Это обеспечивает проверку правила на корректность выполнения и взаимодействие с данными внешних зависимостей до его применения в операционной среде.

`setup_tests` используется для:

- Имитации взаимодействия правила с активными списками и таблицами обогащения.
- Проверки логики правила с учетом ожидаемых данных обогащения.

Процедура настройки `setup_tests`:

1. В корне конфигурации правила указывается секция `setup_tests`.
2. Внутри `setup_tests` через ключ `enrichment_tables` или `active_lists` задаются искусственные таблицы обогащения и активные списки, используемые правилом.
3. Для каждого искусственного элемента указывается его структура и набор искусственных данных, которые будут использоваться при тестировании.
4. Искусственные данные должны соответствовать ожидаемым данным из реальных таблиц обогащения или активных списков и позволять полноценно проверить работу правила.

Структура `setup_tests` для активного списка:

```
setup_tests:
  active_lists:
    <имя_списка>:
      <поле>:
        - [<значение_поля1>]
        - [<значение_поля2>]
```

Структура `setup_tests` для таблицы обогащения:

```
setup_tests:
  enrichment_tables:
    <имя_таблицы>:
      fields:
        - <поле1>
        - <поле2>
      data:
        - [<значение1_поля1>, <значение1_поля2>]
        - [<значение2_поля1>, <значение2_поля2>]
```

Поле `tests`

Поле `tests` включает набор тестов для проверки и верификации правила корреляции. Эти тесты помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Чтобы создать тестовый сценарий, в поле `tests` определите следующие параметры:

- В `name` укажите название теста.
- В `events` создайте массив событий, которые будут использоваться для тестирования. Эти события должны имитировать поток реальных исходных событий, которые поступают на коррелятор.
- В `assertion` установите условие или набор условий, которые должны быть выполнены в результате обработки тестовых событий правилом. Это может включать проверку конкретных полей в итоговых агрегированных событиях или подсчет количества таких событий.

Тестовые события

Тестовые события должны имитировать данные, которые в реальности поступают на коррелятор. Они должны соответствовать формату и структуре событий, обрабатываемых правилом корреляции и содержать поля, которые используются в правиле.



Если в правиле настроена корреляция по полю с типом `DateTime`, например `timestamp`, то значение этого поля в `tests` необходимо задавать в формате `t'<DateTime-значение>'`. Например: `t'15.05.2024 06:28:15'`. Это важно при корреляции на отсутствие событий.

В противном случае при запуске тестов правила система обработает поле в виде строки и корреляция по дате-времени будет работать некорректно.

Условия проверки

Условия должны точно отражать ожидаемый результат применения правила к тестовым событиям. Это может быть проверка наличия определенных полей в событии, соответствие их значений ожидаемым или сравнение количества событий с предполагаемым исходом. Для более сложных правил могут потребоваться более сложные утверждения, которые могут включать логические операции и сравнения.

Пример:

```
tests:
  - name: "Тестирование обнаружения DDoS атаки"
    events:
      - { source_ip: "192.168.1.1", event_type: "traffic_anomaly", traffic_volume:
"10000" }
    assertion: !vrl |
        assert_eq!(.alert_type, "DDoS", "Ожидается обнаружение DDoS атаки")
```

Обогащение через внешние зависимости

Обогащение корреляционных событий представляет собой процесс дополнения событий, сгенерированных в ходе корреляции, дополнительными данными. В декларативных правилах корреляции можно настроить логику обогащения в полях `on_event` и `on_correlation`.

В рамках языка VRL обогащение корреляционных событий включает в себя:

- применение специфических функций и выражений для динамического добавления или модификации данных в событиях;
- извлечение информации из активных списков, которые могут предоставлять как статическую, так и динамическую информацию в зависимости от настроек схемы активного списка;
- взаимодействие с таблицами обогащения, которые предоставляют статическую информацию, например, сведения о геолокации, данные организаций, соответствующие спискам разрешенных и заблокированных ресурсов.

Активные списки

Активные списки представляют собой структуры данных, которые используются для хранения и управления как динамически изменяющимися наборами информации, так и статическими списками и таблицами, такими как IP-адреса, доменные имена, идентификаторы устройств или пользователей. В контексте императивных правил корреляции, поля `on_correlation` и `on_event` могут включать вызовы функций для взаимодействия с активными списками, используя язык VRL. Это позволяет добавлять, извлекать и удалять записи из активных списков на основе логики обработки корреляционных событий.

Функции для взаимодействия с активными списками:

Функция	Описание	Параметры
`active_record_excluded(active_list, key)`	Проверяет, есть ли запись с указанным ключом в активном списке.	• `active_list`: название схемы активного списка, в который добавляется запись. • `key`: ключ записи.
`add_active_record(list, key, value)`	Добавляет запись в активный список. Если запись с таким ключом уже существует, значение обновляется. В противном случае создается новая запись.	• `list`: название схемы активного списка, в который добавляется запись. • `key`: уникальный ключ для записи. • `value`: значение записи.
`get_active_record(list, key)`	Извлекает значение из активного списка по указанному ключу. Возвращает значение, ассоциированное с ключом, или `null`, если запись отсутствует.	• `list`: название схемы активного списка, из которого нужно извлечь запись. • `key`: уникальный ключ записи.

Функция	Описание	Параметры
`remove_active_record (list, key)`	Удаляет запись из активного списка. Если запись с таким ключом отсутствует, ничего не происходит.	• `list`: название схемы активного списка, из которого необходимо удалить запись. • `key`: уникальный ключ записи, которую необходимо удалить.

Пример использования в поле `on_correlation` правила корреляции:

```
on_correlation: !vrl |
  if .threat_level == "High" {
    add_active_record!("suspicious_ips", { "ip": .source_ip, "is_suspicious": true })
  }

  if .event_type == "authentication_failure" {
    failure_count = get_active_record!("auth_failure_counts", { "user_id": .user_id
}).failure_count || 0
    add_active_record!("auth_failure_counts", { "user_id": .user_id, {"failure_count":
failure_count + 1 })
  }

  if .event_type == "authentication_success" {
    remove_active_record!("auth_failure_counts", { "user_id": .user_id })
  }
```

В этом примере:

- IP-адреса с высоким уровнем угрозы добавляются в активный список подозрительных IP-адресов.
- Считается количество неудачных попыток аутентификации для пользователей и обновляется соответствующая запись в активном списке.
- При успешной аутентификации запись пользователя удаляется из списка неудачных попыток аутентификации, что может свидетельствовать об окончании атаки или восстановлении доступа пользователя.

Правило корреляции с активным списком, схема RObject

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule

# Название правила.
name: DDoSDetectionRule

# Режим работы корреляционного окна.
window_mode: parallel

# Тип правила.
type: correlation_rule
```

```
# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика и взаимодействия с активным
списком подозрительных IP.

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .event_type == "HTTP_REQUEST" &&
  .source_ip != null &&
  .request_path != null

# Логика группировки событий.
group_by:
  - source_ip

# Время жизни корреляционного окна в секундах.
expire_after_sec: 600

# Частота генерации корреляционных событий в рамках каждой группы.
throttle_time_sec: 60

# Логика обновления состояния окна корреляции при получении событий.
on_event: !vrl |
  if int!(%state.request_count) >= 10 {
    abort
  }
  %state.request_count = int(%state.request_count) + 1

# Условие открытия корреляционного окна.
```

```
start_when: !vrl |
  .event_type == "HTTP_REQUEST" && .source_ip != null

# Условие закрытия корреляционного окна и генерации корреляционного события.
correlate_when: !vrl |
  int!(%state.request_count) >= 10

# Логика, применяемая после генерации корреляционного события.
on_correlation: !vrl |
  let is_suspicious = get_active_record("suspicious_ips", .source_ip)
  if is_suspicious != null {
    .message = "Обнаружена атака DDoS с источника: " + .source_ip + ". IP помечен как
подозрительный."
  } else {
    .message = "Обнаружена атака DDoS с источника: " + .source_ip + ". IP не найден в
списке подозрительных."
  }

# Инициализация искусственных данных для активных списков.
setup_tests:
  active_lists:
    suspicious_ips:
      - source_ip: "192.0.2.1"
        is_suspicious: true
      - source_ip: "198.51.100.10"
        is_suspicious: false

# Тесты для проверки работы правила.
tests:
  - name: "Detect DDoS from suspicious IP"
    events:
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
    assertion: !vrl |
      assert_eq!(.message, "Обнаружена атака DDoS с источника: 192.0.2.1. IP помечен
как подозрительный.")
   - name: "Detect DDoS from non-suspicious IP"
     events:
       - { "event_type": "HTTP_REQUEST", "source_ip": "198.51.100.10", "request_path":
"/api" }
     assertion: !vrl |
       assert_eq!(.message, "Обнаружена атака DDoS с источника: 198.51.100.10. IP не
найден в списке подозрительных.")
```

Логика правила описывается в блоках `on_event`, `start_when`, `correlate_when`, `on_correlation`, где:

- `on_event` учитывает количество HTTP-запросов от каждого источника IP.
- `start_when` инициирует корреляционное окно при поступлении HTTP-запроса.
- `correlate_when` активирует генерацию корреляционного события, когда количество запросов достигает порога в 10.

- `on_correlation` дополняет корреляционное событие информацией о том, был ли источник IP помечен как подозрительный в активном списке.

Секция `setup_tests` с искусственными данными для активного списка `suspicious_ips` позволяет проверить правило в условиях, максимально приближенных к реальной эксплуатации, без необходимости доступа к продуктивным данным. Это обеспечивает возможность тестирования правила на взаимодействие с активным списком.

Тесты `Detect DDoS from suspicious IP` и `Detect DDoS from non-suspicious IP` демонстрируют работу правила в различных условиях, проверяя его способность корректно реагировать на атаки с подозрительных и не подозрительных IP-адресов соответственно. Успешное прохождение этих тестов подтверждает, что правило корреляции обнаруживает DDoS-атаки и правильно взаимодействует с активными списками для обогащения корреляционных событий дополнительной информацией об источнике атаки.

Таблицы обогащения

Таблицы обогащения представляют собой структурированные наборы данных, используемые для добавления контекстной информации к событиям в процессе их корреляции. Это может включать данные о геолокации, информацию об устройствах, базы данных угроз и другие внешние источники данных. В контексте императивных правил корреляции, поля `on_correlation` и `on_event` могут включать вызовы функции VRL для запроса и извлечения данных из таблиц обогащения, тем самым обогащая события дополнительными деталями для более глубокого анализа.

Функции для взаимодействия с таблицами обогащения:

- `find_enrichment_table_records(table, condition, [select, case_sensitive])`: позволяет выполнить поиск записей в таблице обогащения, которые соответствуют заданным условиям. Возвращает массив записей, удовлетворяющих условиям поиска.
- `get_enrichment_table_record(table, condition, [select, case_sensitive])`: используется для извлечения одной записи из таблицы обогащения, которая точно соответствует заданному условию. В случае, если запись не найдена или найдено более одной записи, функция возвращает ошибку.

Пример использования в поле `on_correlation`:

```
on_correlation: !vrl |
  let geo_info = find_enrichment_table_records("geoip", {"ip": .source_ip}, select:
["country_name", "city_name"])
  if !geo_info.is_empty() {
    .source_country = geo_info[0].country_name
    .source_city = geo_info[0].city_name
  }

  let threat_info = get_enrichment_table_record("threat_intelligence", {"ip":
.source_ip}, select: ["category", "score"])
  if !is_null(threat_info) {
    .threat_category = threat_info.category
    .threat_score = threat_info.score
```

```
}
```

В этом примере:

- Выполняется поиск геолокационной информации по IP-адресу источника события в таблице `geoip`, и результаты добавляются к событию как `.source_country` и `.source_city`.
- Затем производится запрос к таблице `threat_intelligence` для извлечения информации о категории угрозы и оценке уровня угрозы, ассоциированных с IP-адресом источника. Эти данные добавляются к событию как `.threat_category` и `.threat_score`.

Тестирование правила с внешними зависимостями

Тестирование правил корреляции, которые обращаются к таблицам обогащения и/или активным спискам, требует предварительной настройки тестового окружения с помощью поля `setup_tests`, которое позволяет определить искусственные данные для внешнего элемента.

Поле `setup_tests` позволяет инициализировать необходимые данные для таблиц обогащения и активных списков перед запуском тестов правила корреляции. Это обеспечивает возможность проверить не только синтаксическую корректность правила, но и его логику обработки данных при взаимодействии с внешними элементами.

Процесс тестирования

1. Определение искусственных данных: для начала тестирования необходимо определить искусственные данные для всех активных списков и таблиц обогащения, с которым взаимодействует тестируемое правило. Эти данные должны быть репрезентативными и охватывать различные сценарии использования списка в правиле.
2. Конфигурация `setup_tests`: в корне конфигурации правила корреляции необходимо добавить секцию `setup_tests`, где указываются все используемые таблицы обогащения и активные списки с их искусственными данными. Имя каждого внешнего элемента в секции должно точно совпадать с именем, используемым в теле правила.
3. Заполнение искусственных данных: внутри внешнего элемента в секции `setup_tests` указывается список полей и соответствующих им искусственных данных. Данные задаются в формате, соответствующем ожидаемым запросам к активному списку и/или таблице обогащения в рамках теста.

Примеры:

Структура искусственных данных для активного списка:

```
setup_tests:
  active_lists:
    suspicious_ips:
      - source_ip: "192.168.1.1"
        is_suspicious: true
      - source_ip: "10.10.10.10"
        is_suspicious: false
    auth_failure_counts:
      - user_id: "user1"
        failure_count: 1
```

```
      - user_id: "user2"
        failure_count: 2
```

Структура искусственных данных для таблицы обогащения:

```
  enrichment_tables:
    geoip:
      fields:
        - ip
        - country
        - city
      data:
        - ["8.8.8.8", "US", "Mountain View"]
        - ["9.9.9.9", "GB", "London"]
```

Создание и выполнение тестов

После настройки искусственных данных следующим шагом является создание набора тестов, которые будут проверять взаимодействие правила корреляции с активными списками. Каждый тест должен включать в себя:

1. Описание теста (`name`): краткое описание цели теста.
2. События (`events`): список событий, которые будут подаваться на вход правила в процессе тестирования.
3. Утверждения (`assertions`): логика проверки результата работы правила на основе предоставленных событий и искусственных данных активных списков.

Пример теста:

```
tests:
  - name: "Test for Suspicious IP Detection"
    events:
      - { source_ip: "192.168.1.1", event_type: "login_attempt" }
    assertion: !vrl |
        let is_suspicious = get_active_record("suspicious_ips", .source_ip);
        assert_eq!(is_suspicious, true, "IP should be marked as suspicious")
  - name: "Authentication Failure Count Test"
    events:
      - { user_id: "user2", event_type: "login_failure" }
    assertion: !vrl |
        let failure_count = get_active_record("auth_failure_counts", .user_id);
        assert_eq!(failure_count, 2, "User2 should have 2 authentication failures")
```

В разделе событий (`events`) указываются события, которые подаются на вход правила корреляции для проверки его срабатывания. В первом тесте это попытка входа с IP-адреса, который должен быть помечен как подозрительный. Во втором тесте — неудачная попытка входа от пользователя, для которого уже зафиксированы подобные инциденты.

Утверждения (`assertions`) содержат логику проверки, выполняемую после обработки событий правилом. В первом тесте проверяется, что IP-адрес из события корректно идентифицирован как подозрительный на основе данных активного списка `suspicious_ips`. Во втором — что количество неудачных попыток входа для пользователя соответствует ожидаемому значению, основанному на данных активного списка `auth_failure_counts`.

Если утверждение (`assertion`) выполнено успешно, это означает, что правило корреляции корректно обрабатывает входные данные и правильно взаимодействует с активными списками. В противном случае, если результат не соответствует ожидаемому, это указывает на проблемы в логике правила или в его взаимодействии с активными списками.



- Убедитесь, что искусственные данные адекватно отражают разнообразие реальных сценариев, с которыми правило будет сталкиваться в продуктивной среде.
- Тщательно проверяйте соответствие имен активных списков и таблиц обогащения в `setup_tests` именам, используемым в правиле корреляции.
- Используйте результаты тестирования для уточнения и оптимизации логики правила корреляции и взаимодействия с внешним элементом.

Пример правила корреляции

Правило корреляции, схема RObject

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule

# Название правила.
name: DDoSDetectionRule

# Режим работы корреляционного окна.
window_mode: parallel

# Тип правила.
type: correlation_rule

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила корреляции.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
```

```
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .event_type == "HTTP_REQUEST" &&
  .source_ip != null &&
  .request_path != null

# Логика группировки событий.
group_by:
  - source_ip

# Время жизни корреляционного окна в секундах.
expire_after_sec: 600

# Частота генерации корреляционных событий в рамках каждой группы.
throttle_time_sec: 60

# Логика обновления состояния окна корреляции при получении событий.
on_event: !vrl |
  if int!(%state.request_count) >= 10 {
    abort
  }
  %state.request_count = int!(%state.request_count) + 1

# Условие открытия корреляционного окна.
start_when: !vrl |
  .event_type == "HTTP_REQUEST" && .source_ip != null

# Условие закрытия корреляционного окна и генерации корреляционного события.
correlate_when: !vrl |
  int!(%state.request_count) >= 10

# Логика, применяемая после генерации корреляционного события.
on_correlation: !vrl |
  .message = "Обнаружена атака DDoS с источника: " + string!(.event.source_ip)
  .additional_info = "Обнаружено " + string!(%state.request_count) + " запросов с
одного IP за короткий промежуток времени"

# Тесты для проверки работы правила.
tests:
  - name: DDoS Attack Detected
    events:
```

```
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.3", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.2", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.4", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
    assertion: !vrl |
      assert_eq!(.message, "Обнаружена атака DDoS с источника: 192.0.2.1")
```

Интерпретация

Правило корреляции `DDoSDetectionRule` разработано с целью выявления DDoS-атак на основе анализа входящего сетевого трафика.

Метаданные:

- `id`: `example/ddos_detection_rule` — уникальный идентификатор правила.
- `name`: `DDoSDetectionRule` — идентификационное название правила.
- `version`: `1.0.0` — версия правила.
- `author`: `Ivan Ivanov <ivanov@example.ru>` — создатель или команда, отвечающая за правило.
- `description`: уточняет цель правила — обнаружение DDoS-атак, основанное на анализе трафика.
- `tags`: помечено как `ddos` и `security`, что подсказывает об основной функциональности и сфере применения правила.

Параметры корреляции:

- `window_mode`: `parallel` — режим корреляции, где события анализируются параллельно.
- `expire_after_sec`: `600` — время жизни окна корреляции в секундах (10 минут). Окно закроется не

сгенерировав корреляционное событие, если в течение указанного времени не произойдет обновления состояния.

- `throttle_time_sec`: `60` — интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group_by`.
- `type`: `correlation_rule` — тип правила.
- `severity`: `high` — уровень серьезности, присеваемый правилу.

Логика правила:

- `filter`: правило активируется только для событий с типом `HTTP_REQUEST`, имеющих определенный `source_ip` и `request_path`.
- `group_by`: события группируются по полю `source_ip`, что позволяет отслеживать активность каждого отдельного IP-адреса.
- `on_event`: если количество запросов (`request_count`) для данного источника IP достигает или превышает 10, то выполнение правила прерывается с помощью директивы `abort`. Это означает, что атака DDoS уже обнаружена, и состояние перестает меняться. В противном случае значение `request_count` увеличивается на 1.
- `start_when`: определяет условие, при котором начинается новое корреляционное окно. В данном случае это любое событие `HTTP_REQUEST` с непустым полем `source_ip`.
- `correlate_when`: корреляционное событие генерируется, когда количество запросов от одного IP достигает 10.
- `on_correlation`: после обнаружения атаки DDoS формируется корреляционное событие со следующими аттрибутами:

```
{
  "type": "correlation_rule",
  "severity": "high",
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена атака DDoS с источника: 192.168.1.1",
  "additional_info": "Обнаружено 10 запросов с одного IP за короткий промежуток
времени."
}
```

Тестирование правила:

Тест `DDoS Attack Detected` имитирует поток событий, включая 13 HTTP-запросов. Из них 10 запросов исходят от одного IP-адреса 192.168.1.1. После обработки этого потока событий правило должно генерировать одно корреляционное событие с сообщением о DDoS-атаке от IP-адреса 192.168.1.1.

Таким образом, тест `DDoS Attack Detected` проверяет, что правило корреляции правильно реагирует на атаку DDoS, а именно, что оно обнаруживает атаку, когда количество запросов от одного IP-адреса достигает 10 и более.

Таким образом, правило корреляции `DDoSDetectionRule` предназначено для отслеживания атаки

DDoS на основе количества HTTP-запросов от одного источника IP. Как только количество запросов достигает 10 или более за короткий промежуток времени (10 минут), правило обнаруживает атаку и генерирует соответствующее корреляционное событие.

Обработка событий правилом корреляции

Продолжим анализ работы правила `DDoSDetectionRule` на примере исходного события и его обработки.

Предположим, что в течение 10 минут на коррелятор, на котором установлено правило `DDoSDetectionRule`, поступило 10 JSON-событий с источника 192.168.1.1 и они имеют следующий вид:

Исходное событие, JSON

```
{
  "event_type": "HTTP_REQUEST",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:34:56Z",
  "request_path": "/api",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/93.0.4472.124 Safari/537.36"
}
```

При обработке событий происходит срабатывание правила корреляции `DDoSDetectionRule`. Правило создает корреляционное окно каждый раз, когда событие проходит его фильтр. Как только для источника 192.168.1.1 накапливается 10 событий, правило генерирует корреляционное событие, которое будет иметь следующий вид:

Корреляционное событие, генерируемое правилом `DDoSDetectionRule`

```
{
  "type": "correlation_rule",
  "severity": "high",
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена атака DDoS с источника: 192.168.1.1",
  "additional_info": "Обнаружено 10 запросов с одного IP за короткий промежуток
времени."
}
```

Таким образом, правило отслеживает аномально высокую активность отдельных IP-адресов на основе входящих HTTP-запросов.



Чтобы правило могло участвовать в обработке поступающих в систему событий, его необходимо добавить в коррелятор соответствующего коллектора.

Расширение конфигурации существующего правила

Рассмотрим сценарий, в котором необходимо расширить конфигурацию существующего правила

корреляции DDoSDetectionRule с помощью дополнительного условия. Данное условие будет учитывать количество HTTP-запросов, поступающих с одного источника, и ранжировать эти источники в зависимости от этого параметра. В зависимости от количества коррелируемых событий, корреляционному событию будет присвоен один из четырех уровней угрозы: "low" (низкий), "medium" (средний), "high" (высокий) или "critical" (критический).

▼ *Исходное правило корреляции "DDoSDetectionRule", схема RObject*

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule

# Название правила.
name: DDoSDetectionRule

# Режим работы корреляционного окна.
window_mode: parallel

# Тип правила.
type: correlation_rule

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила корреляции.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .event_type == "HTTP_REQUEST" &&
  .source_ip != null &&
  .request_path != null
```

```
# Логика группировки событий.
group_by:
  - source_ip

# Время жизни корреляционного окна в секундах.
expire_after_sec: 600

# Частота генерации корреляционных событий в рамках каждой группы.
throttle_time_sec: 60

# Логика обновления состояния окна корреляции при получении событий.
on_event: !vrl |
  if int!(%state.request_count) >= 10 {
    abort
  }
  %state.request_count = int!(%state.request_count) + 1

# Условие открытия корреляционного окна.
start_when: !vrl |
  .event_type == "HTTP_REQUEST" && .source_ip != null

# Условие закрытия корреляционного окна и генерации корреляционного события.
correlate_when: !vrl |
  int!(%state.request_count) >= 10

# Логика, применяемая после генерации корреляционного события.
on_correlation: !vrl |
  .message = "Обнаружена атака DDoS с источника: " + string!(.event.source_ip)
  .additional_info = "Обнаружено " + string!(%state.request_count) + " запросов с
одного IP за короткий промежуток времени"

# Тесты для проверки работы правила.
tests:
  - name: DDoS Attack Detected
    events:
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.3", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.2", "request_path":
"/api" }
```

```
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.4", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
    assertion: !vrl |
      assert_eq!(.message, "Обнаружена атака DDoS с источника: 192.0.2.1")
```

Конфигурация расширенного правила, схема RObject

```
# Уникальный идентификатор правила.
id: example/enhanced_ddos_detection_rule

# Название правила.
name: EnhancedDDoSDetectionRule

# Режим работы корреляционного окна.
window_mode: parallel

# Тип правила.
type: correlation_rule

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Расширенное правило обнаружения атаки DDoS с классификацией уровней
угрозы

# Имя и контактные данные автора правила корреляции.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
```

```
 - ddos
 - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .event_type == "HTTP_REQUEST" &&
  .source_ip != null &&
  .request_path != null

# Логика группировки событий.
group_by:
  - source_ip

# Время жизни корреляционного окна в секундах.
expire_after_sec: 600

# Частота генерации корреляционных событий в рамках каждой группы.
throttle_time_sec: 60

# Логика обновления состояния окна корреляции при получении событий.
on_event: !vrl |
  if int!(%state.request_count) >= 10 {
    abort
  }
  %state.request_count = int!(%state.request_count) + 1

# Условие открытия корреляционного окна.
start_when: !vrl |
  .event_type == "HTTP_REQUEST" && .source_ip != null

# Условие закрытия корреляционного окна и генерации корреляционного события.
correlate_when: !vrl |
  int!(%state.request_count) >= 10

# Логика, применяемая после генерации корреляционного события.
on_correlation: !vrl |
  .message = "Обнаружена атака DDoS с источника: " + .event.source_ip

  # Расширяем правило для классификации уровней угрозы в зависимости от количества
запросов
  .severity = select(
    int!(%state.request_count) >= 10 => "critical",
    int!(%state.request_count) >= 6 => "high",
    int!(%state.request_count) >= 3 => "medium",
    true => "low"
  )
  .alert_type = "DDoS Attack"
  .additional_info = "Обнаружено " + string!(%state.request_count) + " запросов с
одного IP за короткий промежуток времени"

# Тесты для проверки работы правила.
```

```
tests:
 - name: DDoS Attack Detected
   events:
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.3", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.2", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.4", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
   assertion: !vrl |
     assert_eq!(.message, "Обнаружена атака DDoS с источника: 192.0.2.1")
```

Интерпретация

В поле `severity` расширенного правила устанавливается один из следующих уровней серьезности в зависимости от значения `%state.request_count`:

- Если значение `%state.request_count` больше или равно 10, то уровень серьезности будет установлен как `critical`.
- Если значение `%state.request_count` больше или равно 6, но меньше 10, то уровень серьезности будет установлен как `high`.
- Если значение `%state.request_count` больше или равно 3, но меньше 6, то уровень серьезности будет установлен как `medium`.
- В противном случае, если ни одно из вышеперечисленных условий не выполняется (`true`), то уровень серьезности будет установлен как `low`.

Предположим, что в течение 10 минут на правило `EnhancedDDoSDetectionRule` поступает 10 событий в формате JSON с источника 192.168.1.1, и они имеют следующий вид:

Исходное событие, JSON

```
{
  "event_type": "HTTP_REQUEST",
  "source_ip": "192.168.1.1",
  "datetime": "2023-08-15T12:34:56Z",
  "request_path": "/api/resource",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/91.0.4472.124 Safari/537.36"
}
```

Тогда в результате срабатывании правила корреляции `EnhancedDDoSDetectionRule`, генерируемое корреляционное событие будет иметь следующий вид:

Корреляционное событие, генерируемое правилом `EnhancedDDoSDetectionRule`

```
{
  "type": "correlation_rule",
  "severity": "critical", # Уровень угрозы установлен в значение `critical` согласно
расширенному условию классификации уровней угрозы.
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена атака DDoS с источника: 192.168.1.1",
}
```

Улучшенное правило `EnhancedDDoSDetectionRule` представляет собой расширенную версию исходного правила `DDoSDetectionRule`. Оно расширяет конфигурацию исходного правила так, чтобы при обнаружении DDoS-атак использовалось дополнительное условие, которое учитывает количество HTTP-запросов, приходящих от источников. Условие ранжирует источники по количеству запросов, чтобы присвоить событию один из 4 уровней угрозы: `low`, `medium`, `high` или `critical`. Таким образом, улучшенное правило классифицирует источники атаки DDoS на основе их активности.

Декларативные правила

В данном разделе представлен обзор структуры, принципа работы, процесса создания и расширения конфигурации декларативного правила корреляции. Управление корреляционными правилами в системе осуществляется в разделе **Экспертиза**.

О декларативных правилах корреляции

Декларативное правило корреляции в системе представляет собой структурированное описание желаемых корреляций между событиями безопасности, представленное в виде объекта данных в формате RObject (.ro). Это описание определяет условия корреляции на высоком уровне, фокусируясь на желаемом результате корреляции, а не на точной последовательности операций для его достижения.

Структура декларативного правила корреляции подразумевает определение псевдонимов (`aliases`) для событий и их взаимосвязей. Псевдонимы позволяют группировать события в отдельные потоки и связывать их в логические цепочки (`join`). Условия корреляции и агрегации задаются отдельно для

каждой группы событий, что позволяет системе объединять их в единые корреляционные события.

Процесс корреляции состоит из последовательности операций сопоставления, где каждая операция связывает события на основе определенных критериев. Корреляция завершается, когда событие проходит через всю цепочку операций и соответствует всем условиям. Если задана группировка, событие может быть агрегировано с другими по установленным критериям. Дополнительно корреляционное событие можно обогатить, используя правила, написанные на языке VRL.

Структура правила корреляции

Правила корреляции содержат набор обязательных и опциональных полей.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила корреляции.	строка	да
`name`	Название правила.	строка	да
`type`	Тип элемента. Всегда имеет значение `correlation_rule` для элементов экспертизы этого типа.	строка	да
`version`	Версия правила корреляции, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора правила корреляции.	строка	нет
`description`	Описание правила корреляции, содержащее дополнительную информацию о его назначении и функциональности.	строка	нет
`data_source`	Список источников данных, с которыми совместимо правило корреляции. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`status`	Статус разработки текущей версии правила. Может принимать любые текстовые значения. Примеры значений: • `debug` — версия в разработке; • `experimental` — экспериментальная версия; • `test` — версия для тестирования; • `stable` — стабильная версия.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`date`	Дата создания или изменения правила. Задается в формате `YYYY-MM-DD`, то есть "год-месяц-день". Поле не связано с полями **Дата создания** и **Дата изменения** в карточке элемента экспертизы.	строка	нет
`reference`	Список ссылок на дополнительную информацию об атаках, которые пытается обнаружить правило.	массив строк	нет
`known_false_positives`	Список возможных ложноположительных срабатываний правила. Позволяет определить, было ли правило отработано ложно или нет.	массив строк	нет
`severity`	Уровень угрозы корреляционного события, испускаемого правилом. Допустимые значения: • `critical` — критический; • `high` — высокий; • `medium` — средний; • `low` — низкий.	строка	да
`tags`	Список тегов, присвоенных правилу корреляции для классификации и быстрого поиска.	массив строк	нет
`rate_limit`	Параметры для ограничения генерации корреляционных событий. При превышении ограничений система приостанавливает работу правила корреляции.	объект	нет
`max_correlation_windows`	Максимально допустимое количество корреляционных окон. При превышении ограничения система останавливает работу правила корреляции.	число	нет
`filter`	Блок условий, определяющий события, к которым будет применяться правило корреляции. Условия задаются на языке VRL. При создании правила в конструкторе условия автоматически сохраняются с использованием синтаксиса !expression, но логика работы фильтра остается идентичной.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`aliases`	Псевдонимы потоков событий, которые используются для их идентификации в рамках правила.	объект	да
`select`	Определяет структуру корреляции, включая псевдонимы и условия в `join`. Указывает начальный псевдоним потока событий, с которого начинается процесс корреляции.	объект	да
`group`	Параметры для группировки результатов корреляции.	массив	нет
`ttl`	Максимальное общее время хранения промежуточных корреляционных данных в секундах для всей цепочки событий. При отсутствии индивидуального значения `wait_for` для конкретного `join`, используется значение из поля `ttl`. Является обязательным, если в правиле используются операции соединения (`join`) и группировки (`group`).	число	нет
`on_correlate`	Логика обогащения корреляционного события, выполняемая после успешного соответствия корреляционным условиям. Логика задается на языке VRL. При создании правила в конструкторе условия автоматически сохраняются с использованием синтаксиса !expression, но логика работы блока остается идентичной.	строка	нет
`trigger_alert`	Определяет, должно ли создаваться оповещение по результатам корреляции событий. По умолчанию, если не указано, считается `true`.	логический	нет
`setup_tests`	Настройка тестового окружения для правил, включая предварительную конфигурацию всех используемых в правиле внешних зависимостей, таких как активные списки. Это поле позволяет определить и инициализировать все необходимые ресурсы для выполнения тестов правила, включая искусственные данные для активных списков.	объект	нет (если нет внешних зависимостей)
`throttle_time_sec`	Указывает интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group`.	число	нет

Поле	Описание	Тип данных	Обязательное поле
`tests`	Блок тестирования правила корреляции, содержащий массив тестовых событий и условия для проверки срабатывания правила.	объект	нет



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

В блоках VRL используются правила экранирования строк VRL.

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `filter`

Поле	Описание	Тип данных	Обязательное поле
`filter`	Определяет условия для событий, к которым применяется правило корреляции. Условия задаются на языке VRL. Если правило создается через конструктор, условия сохраняются с использованием синтаксиса `!expression`, где выражения представляют собой вложенные структуры условий. При этом логика работы фильтра остается неизменной.	строка	нет

Синтаксис `filter` в режиме кода

Если правило создается в режиме кода, условия фильтрации задаются на языке VRL.

Пример фильтра, созданного в режиме кода

```
filter: !vrl |
  .sourceIp != null &&
  (.sourceIp == "192.168.0.1" || includes(["192.168.0.2"], .sourceIp)) &&
  .duser != "admin"
```

Синтаксис `filter` в режиме конструктора

При сохранении правила через конструктор фильтр автоматически записывается с использованием синтаксиса `!expression`, при этом логика работы фильтра остается неизменной. Такая структура описывает условия фильтрации в виде вложенных объектов.



Максимальный уровень вложенности для условий фильтрации, созданных в конструкторе, ограничен тремя уровнями. Это ограничение применимо для всех операторов, включая `$and`, `$or`, `$eq`, `$ne`, `$active_list` и `$enrichment_table`.

Если требуется создать более сложные условия, рекомендуется создавать RObject-конфигурацию правила корреляции вручную в режиме кода с использованием синтаксиса VRL.

Основные элементы синтаксиса:

- `$and`: логическое "И" для объединения нескольких условий.
- `$or`: логическое "ИЛИ" для объединения нескольких условий.
- `$eq`: условие равенства поля и значения.
- `$ne`: условие неравенства поля и значения.
- `$active_list`: проверка наличия значения в активном списке.
- `$enrichment_table`: проверка наличия значения в таблице обогащения.
- `$const`: задание фиксированного значения для использования в условиях.
- `$get`: извлечение значения из поля события по указанному пути.

Пример фильтра, созданного в конструкторе

```
filter: !expression |
  $and:
    - $or:
        - $eq:
            path: sourceIp
            value:
              $const: "192.168.0.1"
        - $active_list:
            key:
              - path: sourceIp
                value:
                  $const: "192.168.0.2"
            name: "SuspiciousIPs"
            operator: in
    - $ne:
        path: duser
        value:
          $const: "admin"
```

В этом примере:

- `$and`: объединяет два условия, которые должны одновременно выполняться.
- `$or`: объединяет два условия, одно из которых должно выполняться:
  - `$eq`: проверяет, что поле `sourceIp` равно `"192.168.0.1"`.
  - `$active_list`: проверяет, что `sourceIp` содержится в активном списке `SuspiciousIPs`.
- `$ne`: проверяет, что поле `duser` не равно `"admin"`.

Структура `rate_limit`

Поле	Описание	Тип данных	Обязательное поле
`max_rate`	Максимально допустимое количество корреляционных событий, генерируемых за `duration_secs` секунд.	число	да

Поле	Описание	Тип данных	Обязательное поле
`duration_secs`	Промежуток времени в секундах, в течение которого допустима генерация не более `max_rate` корреляционных событий.	число	да
`grace_period_secs`	Промежуток времени в секундах, в течение которого ожидается восстановление нормального режима работы правила. При нормальном режиме за `duration_secs` секунд правило генерирует не более `max_rate` корреляционных событий. Если при превышении `max_rate` за `duration_secs` количество событий не возвращается к нормальным значениям в течение периода `grace_period_secs`, правило корреляции отключается.	число	да

Структура `aliases`

Поле	Описание	Тип данных	Обязательное поле
Имя_псевдонима	Уникальное имя, идентифицирующее поток событий.	строка (ключ в объекте `aliases`)	да
`filter`	Условие на языке VRL, которое определяет критерии фильтрации событий.	строка	да

Структура `select`

Поле	Описание	Тип данных	Обязательное поле
`alias`	Определяет начальный псевдоним потока событий для корреляции.	строка	да
`join`	Описывает последовательность операций соединения других потоков событий с исходным, указанным в `alias`.	объект	да
`on`	Условия, по которым происходит соединение потоков событий.	массив	да
`absent`	Указывает, должно ли отсутствие события из потока `alias` рассматриваться как условие для перехода к следующему `join`. Если указано `absent: true`, то для этого `join` поле `wait_for` является обязательным. По умолчанию, если не указано, считается `false`.	булево	нет

Поле	Описание	Тип данных	Обязательное поле
`wait_for`	Время жизни для промежуточных результатов корреляции. Если не задано, используется значение из `ttl` верхнего уровня. Если указано `absent: true`, то `wait_for` определяет время ожидания для проверки условия `absent`.	число	нет (но обязательно, если `absent: true`)

Структура `join`

Поле	Описание	Тип данных	Обязательное поле
`join`	Определяет последующие псевдонимы, которые присоединяются к корреляции.	объект	да
`on`	Условия, по которым происходит соединение между псевдонимами потоков событий.	массив	да
`absent`	Указывает, должно ли отсутствие события из потока `alias` рассматриваться как условие для перехода к следующему `join`.	логический	нет
`wait_for`	Время жизни для промежуточных результатов корреляции. Если не задано, используется значение из `ttl` верхнего уровня. Если указано `absent: true`, то `wait_for` определяет время ожидания для проверки условия `absent`.	число	нет (но обязательно, если указано `absent: true`)

Структура `group`

Поле	Описание	Тип данных	Обязательное поле
`alias`	Идентифицирует псевдоним потока событий, по атрибутам которого будет производиться группировка.	строка	да
`by`	Поля, по которым будет производиться группировка.	массив строк	да
`count`	Количество элементов в группе.	число	да
`unique`	Указывает, должна ли группировка выполняться по уникальным значениям полей, указанных в `by`. По умолчанию, если не указано, считается `false`.	логический	нет

Структура `on_correlate`

Поле	Описание	Тип данных	Обязательное поле
`on_correlate`	Определяет логику обогащения корреляционного события, выполняемую после успешного соответствия условиям правила. Логика задается на языке VRL. Если правило создается через конструктор, она сохраняется с использованием синтаксиса `!expression`, при этом логика работы блока остается неизменной.	строка	нет

Синтаксис on_correlate в режиме кода

Если правило создается в режиме кода, логика обогащения задается на языке VRL.

Пример блока обогащения, созданного в режиме кода

```
on_correlate: !vrl |
  if .event_count > 50 {
    .alert_type = "DDoS"
  } else if .event_count > 10 {
    .alert_type = "brute_force"
  } else {
    .alert_type = "suspicious_activity"
  }

  .severity = "high"

  if exists(.event1.sourceIp) {
    .detected_ip = .event1.sourceIp
  } else {
    .detected_ip = "неизвестен"
  }

  if exists(.event2.duser) {
    .detected_user = .event2.duser
  } else {
    .detected_user = "неизвестен"
  }

  .message = "Обнаружена угроза типа " + .alert_type + " от IP-адреса " + .detected_ip
+
             ". Пользователь: " + .detected_user
```

В этом примере:

- `.event_count`: используется для определения типа угрозы.
  - Если значение больше 50, устанавливается `.alert_type` как `"DDoS"`.
  - Если значение больше 10, но не превышает 50, устанавливается `.alert_type` как

"brute_force".

    - В остальных случаях `.alert_type` устанавливается как `"suspicious_activity"`.

- `.severity`: статически задается как `"high"`.
- `.detected_ip`: извлекается из поля `.event1.sourceIp`. Если поле отсутствует, задается как `"неизвестен"`.
- `.detected_user`: извлекается из поля `.event2.duser`. Если поле отсутствует, задается как `"неизвестен"`.
- `.message`: формируется текстовое сообщение с указанием типа угрозы (`.alert_type`), IP-адреса (`.detected_ip`) и пользователя (`.detected_user`).

Синтаксис on_correlate в режиме конструктора

При сохранении правила через конструктор блок обогащения автоматически записывается с использованием синтаксиса `!expression`, при этом логика работы блока остается неизменной. В этом случае действия представляют собой последовательность вложенных объектов, описывающих логику обогащения. Структура сохраняется в формате, где каждое действие (`$assign`, `$vrl`, `$message`) и выражение (`$const`, `$get`) записываются в виде отдельных элементов.

Основные элементы синтаксиса:

- `$assign`: присваивает значение определенному полю в итоговом корреляционном событии. Значение может быть задано напрямую (`$const`) или извлечено из исходного события (`$get`).
- `$vrl`: выполняет программу, написанную на языке VRL, которая может использоваться для выполнения сложных вычислений, преобразований и проверки данных.
- `$message`: формирует текстовое сообщение, которое будет добавлено к корреляционному событию. Сообщение может использовать значения полей из событий или фиксированные строки.
- `$const`: используется для задания фиксированного значения, передаваемого в другие элементы, такие как `$assign`.
- `$get`: извлекает значение из поля или события. Поле задается через параметр `path`, а событие указывается через параметр `event`.

Пример блока обогащения, созданного в конструкторе

```
on_correlate:
  - $assign:
      path: "severity"
      value:
        $const: "high"
  - $assign:
      path: "detected_ip"
      value:
        $get:
          event: "event1"
          path: "sourceIp"
  - $assign:
      path: "detected_user"
```

```
      value:
        $get:
          event: "event2"
          path: "duser"
  - $vrl: |
      if .event_count > 50 {
        .alert_type = "DDoS"
      } else if .event_count > 10 {
        .alert_type = "brute_force"
      } else {
        .alert_type = "suspicious_activity"
      }
  - $message: '"Обнаружена угроза типа " + .alert_type + " от IP-адреса " +
.detected_ip +
               ". Пользователь: " + (.detected_user ?? "неизвестен")'
```

В этом примере:

- `$assign` используется для:
  - Назначения фиксированного значения `"high"` полю `severity`.
  - Получения и присвоения значения поля `sourceIp` из события `event1` в поле `detected_ip`.
  - Получения и присвоения значения поля `duser` из события `event2` в поле `detected_user`.
- `$vrl` применяется для выполнения программы на языке VRL:
  - В зависимости от значения `event_count` задается значение поля `alert_type`.
- `$message` используется для формирования текстового сообщения:
  - Сообщение включает тип угрозы, IP-адрес источника и имя пользователя (если доступно).

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, обрабатываемых правилом.	массив	да
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Операторы

В декларативных правилах операторы используются для определения логики соединения и анализа данных между различными потоками событий. Они позволяют формулировать условия для

сопоставления и группировки событий, основываясь на определенных атрибутах или характеристиках этих событий.



На данный момент в корреляционных правилах поддерживается только оператор `eq`.

Оператор `eq`

Оператор `eq` используется для определения условия равенства между полями в различных потоках событий. Этот оператор является частью механизма `join`, который позволяет связывать события на основе общих характеристик или атрибутов.

Синтаксис

Оператор `eq` используется внутри блока `join` и принимает следующий синтаксис:

```
join:
  alias: <alias_name>
  on:
    - eq: { <alias_1>: <field_1>, <alias_2>: <field_2> }
```

Здесь `<alias_name>` — это псевдоним потока событий, к которому применяется оператор `join`, `<alias_1>` и `<alias_2>` — это псевдонимы потоков событий, между которыми осуществляется сравнение, а `<field_1>` и `<field_2>` — это поля в этих потоках событий, которые сравниваются.

Пример:

В следующем примере демонстрируется использование оператора `eq` для соединения двух потоков событий на основе равенства IP-адресов источника:

```
aliases:
  Traffic1:
    filter: !vrl |
      .event_type == "HTTP_REQUEST"
  Traffic2:
    filter: !vrl |
      .event_type == "FTP_REQUEST"

select:
  alias: Traffic1
  join:
    alias: Traffic2
    on:
      - eq: { Traffic1: .source_ip, Traffic2: .source_ip }
```

В этом примере, события из `Traffic1` и `Traffic2` соединяются, когда поле `.source_ip` совпадает между этими двумя потоками.

Составление правила



Управление корреляционными правилами в системе осуществляется в разделе **Экспертиза**.

Для составления правила корреляции в системе необходимо заполнить обязательные поля правила. Это включает поля метаданных и поля, отвечающие за логику работы правила.

1. Определите метаданные правила:
   - `id`: задайте уникальный идентификатор правила.
   - `name`: задайте название правила.
   - `version`: укажите версию правила, следуя стандартам Semantic Versioning.
   - `type`: определите тип правила как `correlation_rule`.
   - `window_mode`: оставьте поле пустым, чтобы система ожидала декларативное правило.
   - `author` (опциональное): укажите имя и контактные данные автора.
   - `description` (опциональное): предоставьте краткое описание назначения и функциональности правила.
   - `data_source` (опциональное): укажите список совместимых с правилом источников данных для классификации правила в системе.
   - `status` (опционально): опишите статус разработки текущей версии правила.
   - `date` (опционально): укажите дату создания или изменения правила.
   - `reference` (опционально): предоставьте список ссылок на дополнительный материал об атаках, которые пытается обнаружить правило.
   - `known_false_positives` (опционально): укажите список возможных ложноположительных срабатываний правила.
   - `severity`: укажите уровень угрозы корреляционного события, испускаемого правилом (`critical`, `high`, `medium` или `low`).
   - `tags` (опциональное): добавьте теги для классификации правила.
2. Разработайте логику фильтрации и корреляции событий:
   - `filter`: создайте условие для фильтрации входящих событий. События, которые не удовлетворяют условиям фильтра, исключаются из дальнейшего анализа и не учитываются в цепочке корреляционных операций.
   - `aliases`: создайте псевдонимы для потоков событий с соответствующими фильтрами.
   - `select` и `join`: настройте логику соединения потоков событий, начиная с указанного псевдонима (`alias`) и определяя условия соединения (`on`).
   - `ttl`: определите общее максимальное время (`ttl`) для всей цепочки корреляционных операций, чтобы ограничить продолжительность хранения промежуточных результатов в памяти.
3. Задайте логику группировки событий (опционально):
   - `group`: определите параметры для группировки результатов корреляции, включая псевдоним (`alias`), поля для группировки (`by`) и количество элементов в группе (`count`).

    - `throttle_time_sec`: определите интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group`.
4. Опишите пост-корреляционные действия (опционально):
    - `on_correlate`: определите логику обогащения корреляционного события, используя язык VRL. VRL-программа будет применяться к корреляционному событию после успешной корреляции.
5. Разработайте тесты для правила (опционально):
    - `tests`: создайте тесты, включающие различные типы событий и условия, для проверки эффективности и точности правила корреляции.

Поля метаданных

Метаданные являются основой для идентификации и описания правила корреляции в системе. Они должны быть заполнены следующим образом:

- `id`: установите уникальный идентификатор правила. Этот идентификатор должен быть уникальным в рамках системы.
- `name`: укажите название правила.
- `version`: задайте версию правила, следуя принципам Semantic Versioning (например, 1.0.0).
- `type`: установите тип правила `correlation_rule`.
- `author`: укажите имя и контактные данные создателя правила.
- `description`: предоставьте краткое описание назначения и функциональности правила.
- `data_source`: укажите список совместимых с правилом источников данных для классификации правила в системе.
- `status`: опишите статус разработки текущей версии правила.
- `date`: укажите дату создания или изменения правила.
- `reference`: предоставьте список ссылок на дополнительный материал об атаках, которые пытается обнаружить правило.
- `known_false_positives`: укажите список возможных ложноположительных срабатываний правила.
- `severity`: укажите уровень угрозы корреляционного события, испускаемого правилом (`critical`, `high`, `medium` или `low`).
- `tags`: укажите теги для классификации правила в системе.

Поле `rate_limit`

Поле `rate_limit` позволяет задать ограничения на генерацию корреляционных событий.

1. Укажите в поле `max_rate` максимально допустимое количество корреляционных событий, которое должно генерировать правило.
2. Укажите в поле `duration_secs` период времени в секундах, в течение которого допускается генерация не более `max_rate` корреляционных событий.
3. Укажите в поле `grace_period_secs` период времени в секундах, в течение которого система будет ожидать восстановления нормальной работы правила.

Если за `duration_secs` секунд будет сгенерировано более `max_rate` событий, то система будет ожидать восстановления нормальной работы правила в течение `grace_period_secs` секунд. Если нормальная работа правила не восстановится, система отключит правило корреляции.



Ограничение работает только при отключенном режиме распределенной корреляции.

Пример:

```
rate_limit:
    max_rate: 100
    duration_secs: 60
    grace_period_secs: 300
```

В этом примере устанавливается ограничение на генерацию не более 100 корреляционных событий за 60 секунд (1 минуту). При превышении заданного ограничения, если в течение последующих 300 секунд (5 минут) все еще будет генерироваться более 100 событий в минуту, система отключит правило корреляции.

Поле `max_correlation_windows`

Поле `max_correlation_windows` определяет максимально допустимое количество корреляционных окон. При превышении система останавливает работу правила корреляции. Для возобновления работы правила необходимо заново установить конфигурацию конвейера.



Ограничение работает только при отключенном режиме распределенной корреляции.

Пример:

```
max_correlation_windows: 100
```

В этом примере устанавливается ограничение на открытие не более 100 корреляционных окон.

Поле `filter`

Поле `filter` определяет подмножества событий, которые будут обработаны правилом. Это поле использует язык VRL для описания логики фильтрации, позволяя точно настроить, какие события будут участвовать в корреляции.

Предназначение поля `filter` — оценить каждое входящее событие и определить, соответствует ли оно заданным критериям. Если событие не соответствует условиям фильтра, оно исключается из дальнейшего анализа и не учитывается в корреляционном анализе. Это позволяет снизить объем обрабатываемых данных и увеличить точность и эффективность корреляционных правил.

В VRL для написания условий фильтра используются логические операторы (`&&`, `||`, `!`), сравнения (`==`, `!=`, `<`, `>`, `<=`, `>=`) и функции языка для работы с данными (например, `includes`, `starts_with`). Это позволяет создавать сложные и гибкие условия фильтрации.

Пример:

```
filter: !vrl |
    .source_ip != null &&          # Событие должно иметь ненулевой IP-адрес источника
    .request_path != null &&       # Путь запроса также должен быть указан
    .response_code == 404          # Код ответа должен быть 404
```

В этом примере фильтр выбирает события, у которых есть IP-адрес источника и путь запроса, а также код ответа равен 404. Это может быть полезно для выявления потенциальных попыток несанкционированного доступа или сканирования уязвимостей на веб-сервере.

Использование глобальных функций

Если условия фильтра нужно использовать в нескольких правилах, вы можете создать глобальную функцию с необходимой логикой:

1. Создайте для указанного примера глобальную функцию со следующим исполняемым кодом и аргументами:

   ```
   ...
   # Имя функции.
   name: check_request
   ...

   # Основной исполняемый код.
   source: !vrl |
     source_ip != null &&
     request_path != null &&
     response_code == 404

   # Аргументы функции.
   arguments:
     - keyword: source_ip
       kind: bytes
     - keyword: request_path
       kind: bytes
     - keyword: response_code
       kind: integer
   ...
   ```

2. Вызовите глобальную функцию в фильтре:

   ```
   filter: !vrl |
       check_request(.source_ip, .request_path, .response_code)
   ```

3. Укажите в поле `setup_tests` искусственные данные для тестирования глобальной функции.

Поле `aliases`

Поле `aliases` в правиле корреляции используется для разделения и фильтрации потока событий.

1. Определите псевдонимы. Назначьте каждому потоку событий уникальный псевдоним. Это имя будет использоваться для ссылки на этот поток в других частях правила.

   Например, для потока неудачных попыток входа можно использовать псевдоним `FailedLogins`.

2. Сформулируйте фильтры для каждого псевдонима на языке VRL, аналогично полю `filter`. Укажите, какие события будут относиться к данному потоку, основываясь на их атрибутах, таких как тип события, источник или содержание.

Пример фильтра: `.event_type == "login_failure" && .source_ip != null` — этот фильтр отбирает все события неудачных попыток входа, где указан IP-адрес источника.

Пример:

```
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "login_failure" && .source_ip != null
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST"
```

Поле `select`

Поле `select` в правилах корреляции определяет логику соединения потоков событий. Это поле обеспечивает структурирование и организацию процесса корреляции, позволяя связывать события из разных источников на основе заданных критериев. Левый псевдоним в `join` является основным источником событий, к которому добавляются события из правого псевдонима, удовлетворяющие условиям соединения. Если в правом псевдониме нет событий, соответствующих условиям соединения, то события из левого псевдонима все равно будут обработаны, но без добавления данных из правого псевдонима.

1. Выберите исходный псевдоним. Укажите в поле `alias` псевдоним, который будет использоваться как отправная точка для процесса корреляции. Этот псевдоним должен быть одним из определенных в разделе `aliases`.
2. Разработайте логику соединения (`join`). Используйте раздел `join`, чтобы определить, какие дополнительные потоки событий будут привлекаться для корреляции. В каждом `join` указывается псевдоним другого потока событий, который будет соединяться с исходным.
3. Определите условия соединения (`on`). В каждом блоке `join` укажите условия соединения в разделе `on`, которые определяют критерии соотношения событий из разных потоков.
4. Используйте `absent` (опционально) для определения логики корреляции, основанной на отсутствии события.
5. Укажите `wait_for` (опционально, но обязательно при `absent: true`) для установки времени ожидания для каждой операции соединения. Система будет накапливать события в течение указанного времени для каждого `join` и затем анализировать их на предмет соответствия

заданным критериям соединения. Если `absent: true`, наряду с поступлением событий система будет регистрировать и их отсутствие.

Пример:

```
select:
  alias: TrafficA
  join:
    alias: TrafficB
    on:
      - eq: { TrafficA: .source_ip, TrafficB: .source_ip }
    wait_for: 80
    join:
      alias: TrafficC
      absent: true
      on:
        - eq: { TrafficA: .source_ip, TrafficC: .source_ip }
      wait_for: 60
```

В этом примере, начиная с потока `TrafficA`, сначала происходит соединение с `TrafficB` по совпадению источников IP. Затем проверяется отсутствие событий из `TrafficC`, соответствующих тому же условию, в течение 60 секунд.



Можно создать цепочку `join`, соединяя более двух потоков событий. Это позволяет формировать сложные условия корреляции для выявления более сложных угроз или аномалий.

Поле `join`

Поле `join` в правиле корреляции отвечает за создание корреляционных отношений между потоками событий. Оно используется для объединения двух или более потоков событий на основе заданных критериев, включая временной интервал хранения промежуточных результатов корреляции (`ttl`).

- Структура `join`:
  - Каждый `join` включает поля `alias`, `on`, `absent` (опционально), и `wait_for` (опционально).
  - `alias` в `join` указывает псевдоним потока событий, который будет соединяться с начальным потоком, определенным в `select`.
  - `on` описывает условие соединения, обычно основываясь на совпадении значений определенных полей между событиями.
  - `absent` (опционально) определяет, должно ли отсутствие события из потока `alias` рассматриваться как условие для перехода к следующему `join`. Если указано `absent: true`, то для этого `join` поле `wait_for` является обязательным.
  - `wait_for` (опционально, если `absent: false`) устанавливает максимальное время хранения промежуточных результатов корреляции для конкретной операции соединения. Если для конкретного `join` не указано значение `wait_for`, применяется общее ограничение времени, заданное на уровне правила в поле `ttl`. `wait_for` является обязательным полем, если указано `absent: true`.

- Условия соединения `on`:
  - Условие `on` с операцией `eq` (равенство) используется для указания полей, по которым будет происходить сопоставление между потоками событий.
- Создание цепочек `join`:
  - В `join` можно формировать цепочки, где каждый последующий `join` добавляет новый поток событий к уже существующей группе.
  - Это позволяет строить многоуровневые корреляции, где каждый новый уровень добавляет уточнения или расширяет условия корреляции.

Пример:

Для соединения двух потоков событий `TrafficA` и `TrafficB`, когда `source_ip` в `TrafficA` равен `destination_ip` в `TrafficB`, и одновременно проверяется отсутствие события `TrafficC` в заданный временной интервал, используется следующая конструкция:

```
select:
  alias: TrafficA
  join:
    alias: TrafficB
    on:
      - eq: { TrafficA: .source_ip, TrafficB: .destination_ip }
    wait_for: 300  # Время жизни промежуточных результатов в секундах.
    join:
      alias: TrafficC
      absent: true
      on:
        - eq: { TrafficA: .source_ip, TrafficC: .destination_ip }
      wait_for: 60  # Время ожидания отсутствия события в секундах.
```

Поле `group`

Поле `group` в правиле корреляции отвечает за группировку событий, что позволяет проводить более сложный и точный корреляционный анализ.

1. Определите псевдоним для группировки. В `alias` указывается псевдоним потока событий, для которого может быть произведена группировка. Этот псевдоним должен соответствовать одному из определенных в разделе `aliases`.
2. Выберите поля для группировки. Поле `by` определяет, по каким критериям (полям) будут группироваться события. Можно указать одно или несколько полей, на основе которых будут сформированы группы событий.
3. Установите лимит количества событий в группе. Поле `count` определяет минимальное количество событий в каждой группе. Этот параметр позволяет контролировать, когда группа достигает достаточного размера для корреляции или инициирования действия.
4. При необходимости определите признак уникальности группировки событий. В `unique` указывается, должна ли группировка выполняться по уникальным значениям полей, выбранных в `by`:

- Если `unique: true`, то в группу будут собираться события с различными значениями полей.
- Если `unique: false`, то в группу могут собираться события с одинаковыми значениями полей.

Пример:

```
group:
  - alias: TrafficA
    by:
      - source_ip
    count: 10
  - alias: TrafficB
    by:
      - suser
    count: 2
    unique: true
```

В данном примере производится группировка событий из потоков `TrafficA` и `TrafficB`:

- Для событий из потока `TrafficA` группировка производится по одинаковым значениям поля `source_ip`.
- Для событий из потока `TrafficB` группировка производится по уникальным значениям поля `suser`.

Корреляционное событие будет создано, когда в группе соберется 10 событий с одинаковым `source_ip` из потока `TrafficA` и 2 события с различными `suser` из потока `TrafficB`.



Можно задать несколько критериев в `by`, чтобы группировать события по более чем одному полю, что позволяет создавать более сложные условия корреляции.

Поле `throttle_time_sec`

Поле `throttle_time_sec` указывает интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group`.

Пример:

```
throttle_time_sec: 60
```

В этом примере интервал между корреляционными событиями составляет 60 секунд для каждой группы.

Поле `ttl`

Поле `ttl` (time-to-live) определяет общее максимальное время в секундах, в течение которого промежуточные результаты всех корреляционных операций в правиле сохраняются в памяти, прежде чем будут удалены. Это значение применяется ко всем операциям корреляции в правиле. В отдельных `join` блоках может применяться поле `wait_for` для определения времени хранения промежуточных результатов корреляции конкретного `join` блока. При его отсутствии применяется

общее значение `ttl` на уровне правила.



Если в правиле не используются операции соединения (`join`) и группировки (`group`), поле `ttl` является необязательным.

Пример:

```
select:
  alias: TrafficA
  join:
    alias: TrafficB
    on:
      - eq: { TrafficA: .source_ip, TrafficB: .destination_ip }
    ttl: 3600 # Промежуточные результаты для всей цепочки событий этого `join` будут
храниться в течение 1 часа.
```

Поле `on_correlate`

Поле `on_correlate` в декларативных правилах корреляции используется для определения действий обогащения, которые выполняются после успешной корреляции событий. Эти действия могут включать преобразование данных, добавление новой информации к событиям или выполнение вычислений на основе данных событий. Для определения этих действий используется язык VRL. В действиях обогащения могут использоваться глобальные функции, извлечение информации из активных списков и взаимодействие с таблицами обогащения.



Глобальные функции не изменяют исходное событие и его метаданные.

Пример заполнения поля `on_correlate`:

```
on_correlate: !vrl |
  .message = "Обнаружена аномалия доступа с источника: " + .source_ip
  count = to_int!(.event_count)
  if count >= 1000 {
    .threat_level = "Критический"
  } else if count >= 500 {
    .threat_level = "Высокий"
  } else {
    .threat_level = "Средний"
  }
  .analysis_summary = "Источник " + .source_ip + " сгенерировал " +
tostring(.event_count) + " событий, что указывает на потенциальную угрозу."
```

В этом примере:

- Присваивается новое сообщение событию с детальным описанием обнаруженной аномалии.
- Устанавливается уровень угрозы на основе количества событий, связанных с источником.
- Формируется сводка анализа, включающая IP-адрес источника и общее количество событий.

Поле `trigger_alert`

Поле `trigger_alert` определяет, должны ли по результатам корреляции событий создаваться оповещения в системе. Поле `trigger_alert` может принимать значения `true` или `false`:

- Если `trigger_alert` установлено в `true`, оповещение будет создаваться при каждом успешном случае корреляции, соответствующем условиям правила.
- Если `trigger_alert` установлено в `false`, оповещения создаваться не будут, даже в случае корреляции.



Если поле `trigger_alert` не указано в правиле, его значение по умолчанию считается `true`.

Изменение значения `trigger_alert` в существующем правиле влияет на обработку будущих событий, но не затрагивает уже созданные оповещения.

Пример:

```
# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
 .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(.source_ip)
 .additional_info = "Высокая частота запросов и подозрительные паттерны в пути запроса
/api/highload"

# Управляет созданием оповещений по результатам корреляции.
trigger_alert: true

# Тесты для проверки работы правила.
tests:
...
```

Тестирование правила

Тесты в правиле корреляции позволяют проверить корректность работы правила и его способность реагировать на заданные типы событий.

Тестирование правила корреляции гарантирует, что правило корректно обрабатывает данные, точно идентифицирует события и правильно реагирует на них, обеспечивая надежную работу системы.

В случае, если правило корреляции использует глобальные функции или данные из активных списков и таблиц обогащения, необходимо инициализировать искусственные данные через поле `setup_tests`. Это позволит тестировать правило в условиях, максимально приближенных к реальной эксплуатации, без необходимости доступа к продуктивным данным.

Поле `setup_tests`



Если в элементе экспертизы используются внешние зависимости, то есть активные списки, таблицы обогащения или глобальные функции, обязательно заполните поле

> `setup_tests`.

Поле `setup_tests` в конфигурации предназначено для определения тестового окружения, включая предварительную настройку искусственных данных для активных списков, таблиц обогащения и глобальных функций, используемых в элементе экспертизы. Это обеспечивает проверку правила на корректность выполнения и взаимодействие с внешними данными до его применения в операционной среде.

Поле `setup_tests` используется для:

- Имитации взаимодействия с активными списками, таблицами обогащения и глобальными функциями.
- Проверки логики правила или глобальной функции с учетом ожидаемых данных обогащения.

Процедура настройки `setup_tests`:

1. В корне конфигурации укажите секцию `setup_tests`.
2. Внутри секции `setup_tests` через ключ `enrichment_tables`, `active_lists` или `global_functions` задайте искусственные таблицы обогащения, активные списки и глобальные функции, используемые правилом.
3. Для каждого искусственного элемента укажите его структуру и набор искусственных данных, которые будут использоваться при тестировании.



Искусственные данные должны соответствовать ожидаемым данным из реальных таблиц обогащения, активных списков или глобальных функций и позволять полноценно проверить работу элемента экспертизы.

Структура `setup_tests` для активного списка:

```
setup_tests:
  active_lists:
    <active_list_name>:
      <field>:
        - [<field_value_1>]
        - [<field_value_2>]
```

Здесь:

- `<active_list_name>` — имя активного списка.
- `<field>` — поле активного списка.
- `<field_value_N>` — значения поля активного списка.

Структура `setup_tests` для таблицы обогащения:

```
setup_tests:
  enrichment_tables:
    <enrichment_table_name>:
      fields:
```

```
        - <field_1>
        - <field_2>
      data:
        - [<field_1_value_1>, <field_2_value_1>]
        - [<field_1_value_2>, <field_2_value_2>]
```

Здесь:

- `<enrichment_table_name>` — имя таблицы обогащения.
- `<field_N>` — поля таблицы обогащения.
- `<field_N_value_M>` — значения полей таблицы обогащения.

Структура `setup_tests` для глобальной функции:

```
setup_tests:
  global_functions:
    <global_function_name>:
      arguments:
        - keyword: <parameter_1_name>
          kind: <parameter_1_type>
        - keyword: <parameter_2_name>
          kind: <parameter_2_type>
      values_map:
        - argument: <argument>
          event: <event>
          return: <result>
      default: <default_result>
      fallible: <bool_value>
```

Здесь:

- `<global_function_name>` — имя глобальной функции.
- `<parameter_N_name>` — имена параметров глобальной функции.
- `<parameter_N_type>` — типы параметров глобальной функции.
- `<argument>`, `<event>` — аргументы глобальной функции.
- `<result>` — результат глобальной функции.
- `<default_result>` — результат глобальной функции по умолчанию.
- `<bool_value>` — логическое значение.

При вызове искусственная глобальная функция возвращает значение, заданное в `values_map`, если найдена запись с совпадающими аргументами. Если подходящей записи нет, возвращается значение из поля `default`.

Поле `tests`

Поле `tests` включает набор тестов для проверки и верификации правила корреляции. Эти тесты помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Чтобы создать тестовый сценарий, в поле `tests` определите следующие параметры:

- В `name` укажите название теста.
- В `events` создайте массив событий, которые будут использоваться для тестирования. Эти события должны имитировать поток реальных исходных событий, которые поступают на коррелятор.
- В `assertion` установите условие или набор условий, которые должны быть выполнены в результате обработки тестовых событий правилом. Это может включать проверку конкретных полей в итоговых агрегированных событиях или подсчет количества таких событий.

Тестовые события

Тестовые события должны имитировать данные, которые в реальности поступают на коррелятор. Они должны соответствовать формату и структуре событий, обрабатываемых правилом корреляции и содержать поля, которые используются в правиле.



Если в правиле настроена корреляция по полю с типом `DateTime`, например `timestamp`, то значение этого поля в `tests` необходимо задавать в формате `t'<DateTime-значение>'`. Например: `t'15.05.2024 06:28:15'`. Это важно при корреляции на отсутствие событий.

В противном случае при запуске тестов правила система обработает поле в виде строки и корреляция по дате-времени будет работать некорректно.

Условия проверки

Условия должны точно отражать ожидаемый результат применения правила к тестовым событиям. Это может быть проверка наличия определенных полей в событии, соответствие их значений ожидаемым или сравнение количества событий с предполагаемым исходом. Для более сложных правил могут потребоваться более сложные утверждения, которые могут включать логические операции и сравнения.

Пример:

```
tests:
  - name: "Тестирование обнаружения DDoS атаки"
    events:
      - { source_ip: "192.168.1.1", event_type: "traffic_anomaly", traffic_volume:
"10000" }
    assertion: !vrl |
        assert_eq!(.alert_type, "DDoS", "Ожидается обнаружение DDoS атаки")
```

Обогащение через внешние зависимости

Обогащение корреляционных событий представляет собой процесс дополнения событий, сгенерированных в ходе корреляции, дополнительными данными. В декларативных правилах корреляции можно настроить логику обогащения в поле on_correlate.

В рамках языка VRL обогащение корреляционных событий включает в себя применение

специфических функций и выражений для динамического добавления или модификации данных в событиях. Помимо этого, это может включать извлечение информации из активных списков, которые могут предоставлять как статическую, так и динамическую информацию в зависимости от настроек схемы активного списка. А также взаимодействие с таблицами обогащения, которые предоставляют статическую информацию, например, сведения о геолокации, данные организаций, соответствующие спискам разрешенных и заблокированных ресурсов.

Активные списки

Активные списки представляют собой структуры данных, которые используются для хранения и управления как динамически изменяющимися наборами информации, так и статическими списками и таблицами, такими как IP-адреса, доменные имена, идентификаторы устройств или пользователей. В контексте декларативных правил корреляции, поле on_correlate может включать вызовы функций для взаимодействия с активными списками, используя язык VRL. Это позволяет добавлять, извлекать и удалять записи из активных списков на основе логики обработки корреляционных событий.

Функции для взаимодействия с активными списками:

Функция	Описание	Параметры
`active_record_excluded(active_list, key)`	Проверяет, есть ли запись с указанным ключом в активном списке.	• `active_list`: название схемы активного списка, в который добавляется запись. • `key`: ключ записи.
`add_active_record(list, key, value)`	Добавляет запись в активный список. Если запись с таким ключом уже существует, значение обновляется. В противном случае создается новая запись.	• `list`: название схемы активного списка, в который добавляется запись. • `key`: уникальный ключ для записи. • `value`: значение записи.
`get_active_record(list, key)`	Извлекает значение из активного списка по указанному ключу. Возвращает значение, ассоциированное с ключом, или `null`, если запись отсутствует.	• `list`: название схемы активного списка, из которого нужно извлечь запись. • `key`: уникальный ключ записи.
`remove_active_record(list, key)`	Удаляет запись из активного списка. Если запись с таким ключом отсутствует, ничего не происходит.	• `list`: название схемы активного списка, из которого необходимо удалить запись. • `key`: уникальный ключ записи, которую необходимо удалить.

Пример использования в поле `on_correlation` правила корреляции:

```
on_correlation: !vrl |
  if .threat_level == "High" {
    add_active_record!("suspicious_ips", { "ip": .source_ip, "is_suspicious": true })
  }
```

```
  if .event_type == "authentication_failure" {
    failure_count = get_active_record!("auth_failure_counts", { "user_id": .user_id
}).failure_count || 0
    add_active_record!("auth_failure_counts", { "user_id": .user_id, {"failure_count":
failure_count + 1 })
  }

  if .event_type == "authentication_success" {
    remove_active_record!("auth_failure_counts", { "user_id": .user_id })
  }
```

В этом примере:

- IP-адреса с высоким уровнем угрозы добавляются в активный список подозрительных IP-адресов.
- Считается количество неудачных попыток аутентификации для пользователей и обновляется соответствующая запись в активном списке.
- При успешной аутентификации запись пользователя удаляется из списка неудачных попыток аутентификации, что может свидетельствовать об окончании атаки или восстановлении доступа пользователя.

Таблицы обогащения

Таблицы обогащения представляют собой структурированные наборы данных, используемые для добавления контекстной информации к событиям в процессе их корреляции. Это может включать данные о геолокации, информацию об устройствах, базы данных угроз и другие внешние источники данных. В поле `on_correlate` декларативных правил корреляции можно использовать специализированные функции VRL для запроса и извлечения данных из таблиц обогащения, тем самым обогащая события дополнительными деталями для более глубокого анализа.

Функции для взаимодействия с таблицами обогащения:

- `find_enrichment_table_records(table, condition, [select, case_sensitive])`: позволяет выполнить поиск записей в таблице обогащения, которые соответствуют заданным условиям. Возвращает массив записей, удовлетворяющих условиям поиска.
- `get_enrichment_table_record(table, condition, [select, case_sensitive])`: используется для извлечения одной записи из таблицы обогащения, которая точно соответствует заданному условию. В случае, если запись не найдена или найдено более одной записи, функция возвращает ошибку.

Пример использования в поле `on_correlate`:

```
on_correlate: !vrl |
  let geo_info = find_enrichment_table_records("geoip", {"ip": .source_ip}, select:
["country_name", "city_name"])
  if !geo_info.is_empty() {
    .source_country = geo_info[0].country_name
    .source_city = geo_info[0].city_name
  }
```

```
  let threat_info = get_enrichment_table_record("threat_intelligence", {"ip":
.source_ip}, select: ["category", "score"])
  if !is_null(threat_info) {
    .threat_category = threat_info.category
    .threat_score = threat_info.score
  }
```

В этом примере:

- Выполняется поиск геолокационной информации по IP-адресу источника события в таблице `geoip`, и результаты добавляются к событию как `.source_country` и `.source_city`.
- Затем производится запрос к таблице `threat_intelligence` для извлечения информации о категории угрозы и оценке уровня угрозы, ассоциированных с IP-адресом источника. Эти данные добавляются к событию как `.threat_category` и `.threat_score`.

Тестирование правила с внешними зависимостями

Тестирование правил корреляции, которые обращаются к таблицам обогащения и/или активным спискам, требует предварительной настройки тестового окружения с помощью поля `setup_tests`, которое позволяет определить искусственные данные для внешнего элемента.

Поле `setup_tests` позволяет инициализировать необходимые данные для таблиц обогащения и активных списков перед запуском тестов правила корреляции. Это обеспечивает возможность проверить не только синтаксическую корректность правила, но и его логику обработки данных при взаимодействии с внешними элементами.

Процесс тестирования

1. Определение искусственных данных: для начала тестирования необходимо определить искусственные данные для всех активных списков и таблиц обогащения, с которым взаимодействует тестируемое правило. Эти данные должны быть репрезентативными и охватывать различные сценарии использования списка в правиле.
2. Конфигурация `setup_tests`: в корне конфигурации правила корреляции необходимо добавить секцию `setup_tests`, где указываются все используемые таблицы обогащения и активные списки с их искусственными данными. Имя каждого внешнего элемента в секции должно точно совпадать с именем, используемым в теле правила.
3. Заполнение искусственных данных: внутри внешнего элемента в секции `setup_tests` указывается список полей и соответствующих им искусственных данных. Данные задаются в формате, соответствующем ожидаемым запросам к активному списку и/или таблице обогащения в рамках теста.

Примеры:

Структура искусственных данных для активного списка:

```
setup_tests:
  active_lists:
    suspicious_ips:
      schema:
```

```
      source_ip: string
      is_suspicious: bool
    key:
      - ip
    data:
      - source_ip: "192.0.2.1"
        is_suspicious: true
      - source_ip: "192.0.2.2"
        is_suspicious: false
  auth_failure_counts:
    schema:
       user_id: string
       failure_count: integer
    key:
      - user_id
    data:
      - user_id: "user1"
        failure_count: 1
      - user_id: "user2"
        failure_count: 2
```

Структура искусственных данных для таблицы обогащения:

```
enrichment_tables:
  geoip:
    fields:
      - ip
      - country
      - city
    data:
      - ["8.8.8.8", "US", "Mountain View"]
      - ["9.9.9.9", "GB", "London"]
```

Создание и выполнение тестов

После настройки искусственных данных следующим шагом является создание набора тестов, которые будут проверять взаимодействие правила корреляции с активными списками. Каждый тест должен включать в себя:

1. Описание теста (`name`): краткое описание цели теста.
2. События (`events`): список событий, которые будут подаваться на вход правила в процессе тестирования.
3. Утверждения (`assertions`): логика проверки результата работы правила на основе предоставленных событий и искусственных данных активных списков.

Пример теста:

```
tests:
  - name: "Test for Suspicious IP Detection"
```

```
    events:
      - { source_ip: "192.168.1.1", event_type: "login_attempt" }
    assertion: !vrl |
        is_suspicious = get_active_record!("suspicious_ips", {"ip":
.source_ip}.is_suspicious);
        assert_eq!(is_suspicious, true, "IP should be marked as suspicious")
  - name: "Authentication Failure Count Test"
    events:
      - { user_id: "user2", event_type: "login_failure" }
    assertion: !vrl |
        failure_count = get_active_record!("auth_failure_counts", {"user_id":
.user_id});
        assert_eq!(failure_count, 2, "User2 should have 2 authentication failures")
```

В разделе событий (`events`) указываются события, которые подаются на вход правила корреляции для проверки его срабатывания. В первом тесте это попытка входа с IP-адреса, который должен быть помечен как подозрительный. Во втором тесте — неудачная попытка входа от пользователя, для которого уже зафиксированы подобные инциденты.

Утверждения (`assertions`) содержат логику проверки, выполняемую после обработки событий правилом. В первом тесте проверяется, что IP-адрес из события корректно идентифицирован как подозрительный на основе данных активного списка `suspicious_ips`. Во втором — что количество неудачных попыток входа для пользователя соответствует ожидаемому значению, основанному на данных активного списка `auth_failure_counts`.

Если утверждение (`assertion`) выполнено успешно, это означает, что правило корреляции корректно обрабатывает входные данные и правильно взаимодействует с активными списками. В противном случае, если результат не соответствует ожидаемому, это указывает на проблемы в логике правила или в его взаимодействии с активными списками.



- Убедитесь, что искусственные данные адекватно отражают разнообразие реальных сценариев, с которыми правило будет сталкиваться в продуктивной среде.
- Тщательно проверяйте соответствие имен активных списков и таблиц обогащения в `setup_tests` именам, используемым в правиле корреляции.
- Используйте результаты тестирования для уточнения и оптимизации логики правила корреляции и взаимодействия с внешним элементом.

Пример:

▼ *Пример правила корреляции с активным списком, схема RObject*

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_enrich

# Название правила.
name: DDoSDetectionRuleEnrich

# Тип элемента экспертизы.
```

```
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика с учетом репутации IP-адресов
из активного списка DDoSAttackIPs.

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")
  NormalTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path), "/home")

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
```

```
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в
секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  attack_info = get_active_record!("DDoSAttackIPs", {"ip":
%HighRateTraffic[0].source_ip});
  if attack_info != null {
    .attack_type = attack_info.attack_type
    .last_detected = attack_info.last_detected
    .attack_volume = attack_info.attack_volume
    .reputation_score = attack_info.reputation_score
    .involved_attacks = attack_info.involved_attacks
    .notes = attack_info.notes
  }
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip) +
             ", тип атаки: " + to_string!(.attack_type) +
             ", объем атаки: " + to_string!(.attack_volume) +
             ", репутация IP: " + to_string!(.reputation_score) +
             ", количество атак: " + to_string!(.involved_attacks) +
             ", заметки: " + to_string!(.notes)

  if .reputation_score != null && to_int!(.reputation_score) < 50 {
    .ddos_alert = "Высокий уровень запросов и низкая репутация IP, потенциальная
DDoS атака"
  }
# Управляет созданием оповещений по результатам корреляции.
trigger_alert: true

# Искусственные данные для активного списка в тестовом окружении.
setup_tests:
  active_lists:
    DDoSAttackIPs:
      schema:
```

```
        ip: string
        last_detected: string
        attack_type: string
        reputation_score: integer
        attack_volume: string
        involved_attacks: integer
        notes: string
      key:
        - ip
      data:
      - ip: "192.0.2.1"
        last_detected: "2023-01-01T12:00:00Z"
        attack_type: "SYN Flood"
        reputation_score: 10
        attack_volume: "1.5 Gbps"
        involved_attacks: 5
        notes: "Repeated attacks on multiple targets"
      - ip: "192.0.2.3"
        last_detected: "2023-01-03T18:45:00Z"
        attack_type: "Amplification"
        reputation_score: 15
        attack_volume: "2 Gbps"
        involved_attacks: 3
        notes: "DNS amplification attack"
      - ip: "192.0.2.4"
        last_detected: "2023-01-04T20:00:00Z"
        attack_type: "HTTP Flood"
        reputation_score: 5
        attack_volume: "750 Mbps"
        involved_attacks: 4
        notes: "Targeted at web services"

# Тесты для проверки работы правила.
tests:
  - name: HighRateTraffic from Malware IP
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "140s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.111", "window": "10s", "request_path": "/api/highload" }

    assertion: !vrl |
      assert_eq!(.[0]ddos_alert, "Высокий уровень запросов и низкая репутация IP,
потенциальная DDoS атака", "Alert generation failed for malware IP")
  - name: HighRateTraffic from Clean IP
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
```

```
 "source_ip": "192.0.2.2", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
 "source_ip": "192.0.2.2", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
 "source_ip": "192.0.2.2", "window": "140s", "request_path": "start" }

    assertion: !vrl |
      assert!(.[0]ddos_alert == null || .[0]ddos_alert == "", "Alert should not be
 generated for clean IP")
```

Пример правила корреляции

Правило корреляции, схема RObject

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_basic

# Название правила.
name: DDoSDetectionRule

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
```

```
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip)
  .additional_info = "Высокая частота запросов или подозрительные паттерны в пути
запроса /api/highload"

# Управляет созданием оповещений по результатам корреляции.
trigger_alert: true

# Тесты для проверки работы правила.
tests:
  - name: DDoS Attack Detected
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
```

```
"source_ip": "192.0.2.1", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "140s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.111", "window": "10s", "request_path": "/api/highload" }
    assertion: !vrl |
      assert_eq!(.[0]message, "Обнаружена потенциальная DDoS атака с источника:
192.0.2.1")
```

Интерпретация

Правило корреляции "DDoSDetectionRule" предназначено для выявления DDoS-атак на основе анализа потока поступающих событий.

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `correlation_rule` указывает на то, что это правило корреляции.
- `version`: версия правила.
- `description`: описывает цель правила — "Обнаружение атаки DDoS на основе трафика".
- `author`: создатель правила.
- `data_source`: указывает источник правила:
  - `platform`: платформа — Windows.
  - `source`: источник событий — Sysmon.
  - `events`: события `EventID_13` и `EventID_1`.
- `tags`: теги `ddos` и `security` классифицируют правило по его функциональности и области применения.

Логика фильтрации (filter):

- Условия фильтрации `.source_ip != null` и `.request_path != null` выбирают события, у которых есть IP-адрес источника и путь запроса.

Логика фильтрации (aliases):

- Псевдоним `HighRateTraffic`: фильтр для событий с высокой частотой запросов, идентифицируемых по `event_type == "HTTP_REQUEST"` и `parse_duration!(.window) >= 100`. Отбирает события с частотой запросов, превышающей или равной 100 запросов в определенный временной интервал.
- Псевдоним `SuspiciousPatterns`: фильтр для событий с подозрительными путями запроса, определяемый как `event_type == "HTTP_REQUEST"` и `ends_with(to_string!(.request_path), "/api/highload")`. Отбирает запросы, пути которых оканчиваются на "/api/highload".

Соединение потоков событий (select и join):

- Выбор и соединение (`select` и `join`) потоков событий `HighRateTraffic` и `SuspiciousPatterns` осуществляется на основе совпадения `source_ip`. Это соединение позволяет коррелировать высокочастотные запросы с подозрительными паттернами, исходящими от одного источника.
- В поле `wait_for` определяется, что система должна ожидать в течение 30 секунд перед тем, как принять решение о соединении событий из различных псевдонимов. Это значит, что система будет накапливать события в течение указанного времени с момента получения первого события и затем анализировать их на предмет соответствия заданным критериям соединения.
- В операции соединения с `SuspiciousPatterns` используется поле `absent: true`, что указывает на то, что отсутствие события из `SuspiciousPatterns` в течение 30 секунд будет добавлено в цепочку корреляционных событий.

Группировка событий (group):

- `group`: группировка событий применяется к псевдониму `HighRateTraffic` по полю `source_ip`, что позволяет агрегировать и анализировать события, исходящие от одного источника IP. Количество событий для формирования группы в поле `count` установлено в 3, что означает, что группа будет сформирована при накоплении 3 событий от одного источника IP.
- `throttle_time_sec`: задает интервал времени в секундах между генерацией корреляционных событий в рамках каждой группы событий. Например, 60 секунд.

Срок хранения промежуточных данных корреляции (ttl):

- `ttl` на уровне правила задает общее максимальное время жизни промежуточных корреляционных данных. Например, 60 секунд. Это время определяет, как долго промежуточные результаты для всей цепочки событий будут храниться в памяти до их автоматического удаления, если корреляция не завершается.

Пост-корреляционная логика (on_correlate):

- `on_correlate`: обогащает корреляционное событие полями `.message` и `.additional_info`.
- `trigger_alert`: управляет созданием оповещений по результатам корреляции. В данном случае, с `trigger_alert: true`, система будет генерировать оповещения при успешной корреляции событий.

По накоплении 3 событий правило генерирует корреляционное событие со следующими атрибутами:

```
{
  "type": "correlation_rule",
  "severity": "high",
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена потенциальная DDoS атака с источника: 192.168.1.1",
  "additional_info": "Высокая частота запросов и подозрительные паттерны в пути
запроса /api/highload"
}
```

Тестирование правила (tests):

- Набор тестовых сценариев "DDoS Attack Detected" включает набор событий, часть из которых имитируют характеристики DDoS-атаки, а остальные являются контрольными примерами обычного трафика. Цель тестирования — подтвердить, что правило корректно идентифицирует только события, связанные с DDoS-атаками и игнорирует неподходящие события. Тесты включают события с HTTP-запросами, различающимися по `source_ip` и `request_path`, что позволяет проверить способность правила различать атакующий трафик от легитимного.
- События, моделирующие DDoS-атаку, имеют одинаковый `source_ip` и конечный путь запроса "/api/highload", что соответствует фильтрам `HighRateTraffic` и `SuspiciousPatterns`. Это сценарий, в котором ожидается активация правила и генерация корреляционного события.
- Контрольные события с разными `source_ip` и неподозрительными путями запроса ("/home", "/contact", "/about") предназначены для подтверждения, что правило не реагирует на обычные запросы, не связанные с DDoS-активностью.

`assertion` в тестовом блоке предназначен для проверки того, что только события, соответствующие характеристикам DDoS-атаки, приводят к формированию корреляционного события с ожидаемым сообщением об обнаружении атаки.

Обработка событий правилом корреляции

Проанализируем логику работы правила "DDoSDetectionRule", фокусируясь на процессе обработки событий.

Предположим, что за короткий промежуток времени на коррелятор, на котором установлено правило "DDoSDetectionRule", поступает поток JSON-событий от различных источников IP, включая множество запросов с IP-адреса 192.168.1.1. Эти события соответствуют критериям, заданным в фильтрах `HighRateTraffic` и `SuspiciousPatterns`.

Исходное событие, JSON

```
{
  "event_type": "HTTP_REQUEST",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:34:56Z",
  "request_path": "/api/highload",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/93.0.4472.124 Safari/537.36"
}
```

По мере поступления этих событий, они фильтруются правилом "DDoSDetectionRule". Фильтр `HighRateTraffic` отбирает события, соответствующие запросам с высокой частотой (более 100 запросов за заданный временной интервал), а `SuspiciousPatterns` фокусируется на запросах с определенным путем запроса `/api/highload`.

Когда события от одного источника IP, например, 192.168.1.1, соответствуют обоим фильтрам и проходят через механизм соединения `join`, правило "DDoSDetectionRule" приступает к группировке. Правило использует поле `group` для агрегирования событий от одного источника IP. Если количество событий от одного источника соответствует заданному порогу в 3 события, правило генерирует корреляционное событие.

Корреляционное событие, генерируемое правилом "DDoSDetectionRule"

```
{
  "type": "correlation_rule",
  "severity": "high",
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена потенциальная DDoS атака с источника: 192.168.1.1",
  "additional_info": "Высокая частота запросов и подозрительные паттерны в пути
запроса /api/highload"
}
```

Данное корреляционное событие указывает на то, что система обнаружила аномальную активность, соответствующую критериям DDoS-атаки, исходящей с одного IP-адреса. Правило "DDoSDetectionRule" коррелирует события, основываясь на двух ключевых факторах: частоте запросов и специфических путях запросов, что является индикатором потенциальной атаки.

Расширение конфигурации существующего правила

Рассмотрим сценарий, в котором необходимо добавить дополнительные условия фильтрации и корреляции для правила "DDoSDetectionRule", чтобы повысить его эффективность и надежность в обнаружении более сложных DDoS-атак.

▼ *Исходное правило корреляции "DDoSDetectionRule", схема RObject*

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_basic

# Название правила.
name: DDoSDetectionRule

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
```

```
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в
секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
```

```
to_string!(%HighRateTraffic[0].source_ip)
  .additional_info = "Высокая частота запросов или подозрительные паттерны в пути
запроса /api/highload"

# Управляет созданием оповещений по результатам корреляции.
trigger_alert: true

# Тесты для проверки работы правила.
tests:
  - name: DDoS Attack Detected
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "140s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.111", "window": "10s", "request_path": "/api/highload" }
    assertion: !vrl |
      assert_eq!(.[0]message, "Обнаружена потенциальная DDoS атака с источника:
192.0.2.1")
```

Для этого добавим дополнительный псевдоним с фильтрацией для отбора событий по значению поля `user_agent` исходного события. Также добавим обращение к активному списку для динамической оценки уровня угрозы, основанной на текущей информации о подозрительных IP-адресах.

Пример улучшенного правила "EnhancedDDoSDetectionRule", схема RObject

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_enhanced

# Название правила.
name: EnhancedDDoSDetectionRule

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Расширенное обнаружение DDoS-атак на основе трафика

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>
```

```
# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security
  - enhanced_detection

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")
  # Дополнительный псевдоним.
  UnusualUserAgents:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && !match(to_string!(.user_agent), r'Mozilla\/5\.0
\(Windows NT 10\.0; Win64; x64\) AppleWebKit\/537\.36 \(KHTML, like Gecko\)
Chrome\/91\.0\.4472\.124 Safari\/537\.36')

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30
    join:
      alias: UnusualUserAgents
      on:
        - eq: { HighRateTraffic: .user_agent, UnusualUserAgents: .user_agent }
      wait_for: 40
```

```
# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
      - user_agent # Дополнительный критерий фильтрации.
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip)
  .additional_info = "Высокая частота запросов или подозрительные паттерны в пути
запроса /api/highload и необычные user agents"
  # Обращение к активному списку "suspicious_ips"
  if get_active_record!("suspicious_ips", {"ip": %HighRateTraffic[0].source_ip}) !=
null {
    .threat_level = "high"
  } else {
    .threat_level = "medium"
  }

# Управляет созданием оповещений по результатам корреляции.
trigger_alert: true

# Искусственные данные активного списка.
setup_tests:
  active_lists:
    suspicious_ips:
      schema:
        ip: string
      key:
        - ip
      data:
        - ip: "192.0.2.1"
        - ip: "198.51.100.10"

# Тесты для проверки работы правила.
tests:
  - name: Enhanced DDoS Attack Detection
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "130s", "request_path": "start" }
```

```
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "140s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.111", "window": "10s", "request_path": "/api/highload" }
    assertion: !vrl |
      assert_eq!(.[0]message, "Обнаружена потенциальная DDoS атака с источника:
192.0.2.1")
      assert_eq!(.[0]threat_level, "high")
```

Таким образом, в улучшенное правило "EnhancedDDoSDetectionRule" внесены следующие изменения:

1. Дополнительный псевдоним и условие фильтрации:
   - Добавлен псевдоним `UnusualUserAgents`, в рамках которого происходит фильтрация событий по `user_agent` для идентификации необычных или подозрительных запросов.
   - Фильтр `.event_type == "HTTP_REQUEST" && regex_match("^Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (HTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36", .user_agent)` отбирает события со значением "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (HTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36" в поле `user_agent`.
   - Добавлен `UnusualUserAgents` в `select` с последующей корреляцией по `user_agent`.
2. Настройка `ttl`:
   - Значение `ttl` установлено в 80 секунд, увеличивая период хранения промежуточных корреляционных данных.
3. Дополнительный критерий группировки:
   - Добавлена группировка по `user_agent`, для более точного определения распределенных атак.
4. Интеграция с активным списком:
   - Использована функция `get_active_record("suspicious_ips", {"ip": %HighRateTraffic[0].source_ip})` для обогащения результатов проверки данными из активного списка "suspicious_ips". Это реализовано через условие:

     ```
       if get_active_record("suspicious_ips", {"ip": %HighRateTraffic[0].source_ip})
     != null {
         .threat_level = "high"
       } else {
         .threat_level = "medium"
       }
     ```

     Таким образом, при генерации корреляционного события, помимо основного сообщения и дополнительной информации, теперь также устанавливается уровень угрозы в зависимости от статуса IP-адреса в активных списках. Уровень угрозы устанавливается в "high", если IP-адрес источника обнаружен в списке подозрительных, и в "medium" в остальных случаях.
5. Тестовая проверка:
   - Добавлен блок `setup_tests` для предварительной настройки тестового окружения, включающий активный список `suspicious_ips` с указанными IP-адресами "192.0.2.1" и "198.51.100.10". Это позволяет эмулировать обращение к активному списку во время

тестирования правила и проверять логику обогащения событий на основе наличия IP-адреса в списке подозрительных.

- Добавлены тестовые события со значением "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (HTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36" в поле `user_agent`, чтобы проверить правило на способность корректно идентифицировать и обрабатывать события с конкретным значением в `user_agent`.
- Добавлена проверка динамической оценки уровня угрозы на основе данных из активного списка. Проверяется, каким образом правило реагирует на обнаружение подозрительных IP-адресов и присваивает соответствующий уровень угрозы (`high` или `medium`).

Интерпретация

Основной поток (select):

- `alias: HighRateTraffic` определяет начальный поток событий для корреляции, который в данном случае представляет поток `HighRateTraffic`. Поток `HighRateTraffic` охватывает события, соответствующие определенным критериям, таким как высокая частота HTTP-запросов от одного источника IP.

Первый уровень соединения (join):

- `alias: SuspiciousPatterns` указывает на то, что поток `HighRateTraffic` будет соединяться с потоком `SuspiciousPatterns`. Поток `SuspiciousPatterns` фильтрует события на основе определенных паттернов в пути запроса (например, оканчивающихся на "/api/highload").
- `on: - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }` говорит о том, что соединение между потоками `HighRateTraffic` и `SuspiciousPatterns` будет осуществляться на основе совпадения значения поля `.source_ip`. То есть корреляция произойдет, если источник IP в обоих потоках событий совпадает.
- В операции соединения `HighRateTraffic` и `SuspiciousPatterns` используется поле `absent: true`, что указывает на то, что отсутствие события из `SuspiciousPatterns` в течение 30 секунд будет добавлено в цепочку корреляционных событий.

Второй уровень соединения (join):

- `alias: UnusualUserAgents` добавляет третий поток событий `UnusualUserAgents` в цепочку корреляции. Этот поток фокусируется на событиях с "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (HTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36" в поле `user_agent`.
- `on: - eq: { HighRateTraffic: .user_agent, UnusualUserAgents: .user_agent }` устанавливает, что соединение между `HighRateTraffic` и `UnusualUserAgents` осуществляется на основе совпадения по полю `.user_agent`. Это означает, что для корреляции в рамках данного правила событие должно соответствовать критериям как высокой частоты запросов, так и "подозрительного" `user_agent`.

Таким образом, создается многоуровневая логика корреляции, где события должны соответствовать критериям всех трех потоков — `HighRateTraffic`, `SuspiciousPatterns` и `UnusualUserAgents`. Это позволяет правилу "EnhancedDDoSDetectionRule" выявлять потенциальные DDoS-атаки, основываясь на более сложных и детализированных критериях, чем в исходной версии правила.

Правила нормализации

В данном разделе представлен обзор структуры, принципов работы, процесса создания и расширения конфигурации правила нормализации. Управление правилами нормализации в системе осуществляется в разделе **Экспертиза**.

О правилах нормализации

Правила нормализации в системе обрабатывают и преобразовывают поступающие события согласно условиям и коду нормализации, составленным на языке VRL. В данном разделе представлен обзор процесса создания правила нормализации, его структуры и методов расширения конфигурации существующего правила нормализации.

Работа с правилом нормализации

Доступные операции над правилом нормализации:

- Создание правила нормализации
- Изменение правила нормализации
- Удаление правила нормализации

Создание правила нормализации

Чтобы добавить правило нормализации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило нормализации**. Система отобразит окно создания правила нормализации.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля правила нормализации, чтобы определить логику его работы.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст правило нормализации и отобразит соответствующее уведомление. Новое правило появится в таблице раздела **Экспертиза**.



Чтобы правило нормализации стало доступно на конвейерах событий, его необходимо включить.

Вы можете создать черновик правила нормализации без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик правила и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение правила нормализации

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление правила нормализации

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура правила нормализации

Правила нормализации описываются на языке RObject и содержат следующие поля:

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила.	строка	Да
`name`	Название правила, которое обозначает его цель или функцию.	строка	Да
`version`	Версия правила, представленная в формате Semantic Versioning, указывает на текущую версию правила.	строка	Да
`type`	Тип правила. Всегда имеет значение `normalization_rule` для правила этого типа.	строка	Да
`author`	Имя и контактные данные автора правила нормализации.	строка	Нет
`description`	Описание правила, где можно указать дополнительную информацию о его назначении и функциональности.	строка	Нет
`data_source`	Список источников данных, с которыми совместимо правило нормализации. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных правилу нормализации для классификации и быстрого поиска.	массив строк	Нет
`filter`	Блок правил на языке VRL, определяющий условия (фильтр), при которых правило будет применяться к событиям. Фильтр позволяет выбирать определенные события для нормализации на основе заданных критериев.	строка	Нет
`normalizer`	Блок правил на языке VRL, определяющий правила нормализации события.	строка	Нет

Поле	Описание	Тип данных	Обязательное поле
`tests`	Блок тестирования правила нормализации, содержащий массив тестовых событий и условия для проверки корректности работы правила.	объект	Нет
`setup_tests`	Настройка тестового окружения для правил, включая предварительную конфигурацию всех используемых в правиле внешних зависимостей, таких как таблицы обогащения. Это поле позволяет определить и инициализировать все необходимые ресурсы для выполнения тестов правила, включая искусственные данные для активных списков.	объект	нет (если нет внешних зависимостей)



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

В блоках VRL используются правила экранирования строк VRL.

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
```

```
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, обрабатываемых правилом.	массив	да
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Структура `setup_tests`



Если в элементе экспертизы используются внешние зависимости, то есть активные списки, таблицы обогащения или глобальные функции, обязательно заполните поле `setup_tests`.

Поле `setup_tests` в конфигурации предназначено для определения тестового окружения, включая предварительную настройку искусственных данных для активных списков, таблиц обогащения и глобальных функций, используемых в элементе экспертизы. Это обеспечивает проверку правила на корректность выполнения и взаимодействие с внешними данными до его применения в операционной среде.

Поле `setup_tests` используется для:

- Имитации взаимодействия с активными списками, таблицами обогащения и глобальными функциями.
- Проверки логики правила или глобальной функции с учетом ожидаемых данных обогащения.

Процедура настройки `setup_tests`:

1. В корне конфигурации укажите секцию `setup_tests`.
2. Внутри секции `setup_tests` через ключ `enrichment_tables`, `active_lists` или `global_functions` задайте искусственные таблицы обогащения, активные списки и глобальные функции, используемые правилом.
3. Для каждого искусственного элемента укажите его структуру и набор искусственных данных, которые будут использоваться при тестировании.



Искусственные данные должны соответствовать ожидаемым данным из реальных таблиц обогащения, активных списков или глобальных функций и позволять полноценно проверить работу элемента экспертизы.

Структура `setup_tests` для активного списка:

```
setup_tests:
  active_lists:
    <active_list_name>:
      <field>:
        - [<field_value_1>]
        - [<field_value_2>]
```

Здесь:

- `<active_list_name>` — имя активного списка.
- `<field>` — поле активного списка.
- `<field_value_N>` — значения поля активного списка.

Структура `setup_tests` для таблицы обогащения:

```
setup_tests:
  enrichment_tables:
    <enrichment_table_name>:
      fields:
        - <field_1>
        - <field_2>
      data:
        - [<field_1_value_1>, <field_2_value_1>]
        - [<field_1_value_2>, <field_2_value_2>]
```

Здесь:

- `<enrichment_table_name>` — имя таблицы обогащения.

- `<field_N>` — поля таблицы обогащения.
- `<field_N_value_M>` — значения полей таблицы обогащения.

Структура `setup_tests` *для глобальной функции:*

```
setup_tests:
  global_functions:
    <global_function_name>:
      arguments:
        - keyword: <parameter_1_name>
          kind: <parameter_1_type>
        - keyword: <parameter_2_name>
          kind: <parameter_2_type>
      values_map:
        - argument: <argument>
          event: <event>
          return: <result>
      default: <default_result>
      fallible: <bool_value>
```

Здесь:

- `<global_function_name>` — имя глобальной функции.
- `<parameter_N_name>` — имена параметров глобальной функции.
- `<parameter_N_type>` — типы параметров глобальной функции.
- `<argument>`, `<event>` — аргументы глобальной функции.
- `<result>` — результат глобальной функции.
- `<default_result>` — результат глобальной функции по умолчанию.
- `<bool_value>` — логическое значение.

При вызове искусственная глобальная функция возвращает значение, заданное в `values_map`, если найдена запись с совпадающими аргументами. Если подходящей записи нет, возвращается значение из поля `default`.

Составление правила

Для составления правила нормализации в системе необходимо выполнить следующие шаги:

1. Определите метаданные правила:
   - `id`: задайте уникальный идентификатор правила.
   - `name`: задайте название правила.
   - `version`: укажите версию правила, следуя стандартам Semantic Versioning.
   - `type`: укажите тип правила как `normalization_rule`.
   - `author` (опционально): укажите имя и контактные данные автора.
   - `description` (опционально): предоставьте краткое описание назначения и функциональности

правила.

- `data_source` (опционально): укажите список совместимых с правилом источников данных для классификации правила в системе.
- `tags` (опционально): добавьте теги для классификации и быстрого поиска правила.

2. Разработайте логику фильтрации событий:
   - `filter`: создайте условие для фильтрации входящих событий. События, которые не удовлетворяют условиям фильтра, исключаются из дальнейшего процесса нормализации.
3. Определите правила нормализации:
   - `normalizer`: создайте блок правил на языке VRL, который будет применяться для нормализации выбранных событий.
4. Разработайте тесты для проверки правила (опционально):
   - `tests`: создайте тесты, включающие различные типы событий и условия, для проверки корректности работы правила нормализации.

Пример правила нормализации

Правило нормализации, схема RObject

```
# Уникальный идентификатор правила.
id: example/basic_normalization_rule

# Название правила.
name: BasicNormalizationRule

# Имя автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Базовое правило нормализации

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - logs/default.log
    - logs/somelog.log

# Теги, связанные с правилом.
tags:
  - normalization
  - security

# Тип элемента экспертизы.
```

```
type: normalization_rule

# Блок правил на языке VRL с условиями фильтрации событий.
filter: !vrl |
  .event.host == "198.51.100.191"

# Блок с правилами на языке VRL для нормализации полей события.
normalizer: !vrl |
    e = parse_json(.event) ?? {}
    . |= {
      "userId": e."user-identifier",
      "timestamp": parse_timestamp(e.datetime, "%d/%b/%Y:%H:%M:%S") ?? now(),
      "host": e.host,
      "method": e.method,
      "path": e.request,
      "protocol": e.protocol
    }

    if e.status == "404" || e.status == "401" {
      .responseStatus = "Bad request"
    }

    if e.protocol == "HTTP/2.0" {
        .protocol = "HTTP/2"
    }
```

Интерпретация

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `normalization_rule` указывает на то, что это правило нормализации.
- `version`: версия правила.
- `description`: описывает цель правила — "Базовое правило нормализации".
- `author`: создатель правила.
- `data_source`: указывает источник правила:
  - `platform`: платформа — Windows.
  - `source`: источник событий — Sysmon.
  - `events`: журналы событий `logs/default.log` и `logs/somelog.log`.
- `tags`: теги `normalization` и `security` классифицируют правило по его функциональности и области применения.

Фильтрация событий (`filter`):

- `.event.host == "198.51.100.191"`:

Сравнение поля `host` со значением `198.51.100.191` в сыром событии. Если значение поля `host` в сыром событии совпадает с `198.51.100.191`, то событие будет нормализовано согласно правилам в поле `normalizer`. Если значение поля `host` не совпадает с указанным IP-адресом, правила нормализации к событию применены не будут.

Нормализация событий (`normalizer`):

- `e = parse_json(.event) ?? {}`:

  Попытка преобразовать содержимое поля `.event` из формата JSON в структуру данных VRL.

  Если преобразование не удалось или результатом является `null`, переменной `e` будет присвоен пустой объект `{}`, что достигается с помощью оператора `??`.

- `. |= { ⋯ }`:

  Оператор обновления или добавления новых полей к текущему объекту на основе значений из объекта `e`.

  Структура, которая присваивается текущему объекту:

  - `userId`: присваивает значение поля `user-identifier` из объекта `e` к полю `userId` текущего объекта.
  - `timestamp`: преобразует строковое значение поля `datetime` из объекта `e` в метку времени с использованием указанного формата. Если преобразование не удастся или значение поля `datetime` равно `null`, будет использовано текущее время, возвращаемое функцией `now()`.
  - `host`: присваивает значение поля `host` из объекта `e` к полю `host` текущего объекта.
  - `method`: присваивает значение поля `method` из объекта `e` к полю `method` текущего объекта.
  - `path`: присваивает значение поля `request` из объекта `e` к полю `path` текущего объекта.
  - `protocol`: присваивает значение поля `protocol` из объекта `e` к полю `protocol` текущего объекта.

- `if e.status == "404" || e.status == "401"`:

  Условное выражение проверяет, равно ли значение поля `status` из объекта `e` строке `404` или `401`.

- `.responseStatus = "Bad request"`:

  Если условие выше выполняется, то полю `responseStatus` текущего объекта присваивается значение `Bad request`.

- `if e.protocol == "HTTP/2.0"`:

  Условное выражение проверяет, равно ли значение поля `protocol` из объекта `e` строке `HTTP/2.0`.

- `.protocol = "HTTP/2"`:

  Если условие выше выполняется, то поле `protocol` текущего объекта заменяется на значение `HTTP/2`.

Обработка событий правилом нормализации "BasicNormalizationRule"

Рассмотрим работу правила на примере разбора базового (сырого) события.

Допустим, на нормализатор поступают события следующего вида:

Исходное событие, JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"IvanIvanov\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\",\
"request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"status\":
\"404\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}"
}
```

После применения к событию правила "BasicNormalizationRule", оно принимает следующий вид:

Нормализованное событие, JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"IvanIvanov\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\",\
"request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"status\":
\"404\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}",
  "host": "198.51.100.191",
  "method": "GET",
  "path": "/observability/metrics/production",
  "protocol": "HTTP/2",
  "referer": "https://example.url.com",
  "responseBytes": 22427,
  "responseStatus": "Bad request",
  "serverIP": "198.51.100.191",
  "timestamp": "2023-07-30T23:55:49Z",
  "userAgent": null,
  "userId": "IvanIvanov"
}
```

Событие удовлетворяет условию из поля `filter`, следовательно, оно подлежит нормализации и обрабатывается согласно правилам из поля `normalizer`. В структуре нормализованного события в поле `event` записывается исходное событие. Прочие поля добавляются к объекту и определяются в соответствии с правилами нормализации, хранящимся в поле `normalizer`.

Расширение конфигурации существующего правила нормализации

Рассмотрим сценарий, в котором требуется расширить конфигурацию существующего правила "BasicNormalizationRule" таким образом, чтобы помимо базового преобразования полей времени, источника и статуса события, оно также устанавливало уровень угрозы в отдельном поле в зависимости от значения в поле `status`. Также добавим условия для разбора остальных полей события, которые не обрабатываются в исходном правиле нормализации.

Допустим, исходное правило нормализации "BasicNormalizationRule" выглядит следующим образом:

▼ *Исходное правило нормализации BasicNormalizationRule, схема RObject*

```
# Уникальный идентификатор правила.
id: example/basic_normalization_rule

# Название правила.
name: BasicNormalizationRule

# Имя автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Базовое правило нормализации

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - logs/default.log
    - logs/somelog.log

# Теги, связанные с правилом.
tags:
  - normalization
  - security

# Тип элемента экспертизы.
type: normalization_rule

# Блок правил на языке VRL с условиями фильтрации событий.
filter: !vrl |
  .event.host == "198.51.100.191"

# Блок с правилами на языке VRL для нормализации полей события.
normalizer: !vrl |
    e = parse_json(.event) ?? {}
    . |= {
      "userId": e."user-identifier",
      "timestamp": parse_timestamp(e.datetime, "%d/%b/%Y:%H:%M:%S") ?? now(),
      "host": e.host,
      "method": e.method,
      "path": e.request,
      "protocol": e.protocol
    }

    if e.status == "404" || e.status == "401" {
      .responseStatus = "Bad request"
```

```
    }

    if e.protocol == "HTTP/2.0" {
        .protocol = "HTTP/2"
    }
```

Тогда расширенная конфигурация правила на основе правила будет выглядеть следующим образом:

Конфигурация расширенного правила, схема RObject

```
# Уникальный идентификатор правила.
id: example/enhanced_normalization_rule

# Название правила.
name: EnhancedNormalizationRule

# Имя автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Улучшенное правило нормализации JSON-события с определением уровня угрозы
в поле `severity`.

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
     - logs/default.log
     - logs/somelog.log

# Теги, связанные с правилом.
tags:
  - security
  - normalization

# Тип элемента экспертизы.
type: normalization_rule

# Блок правил на языке VRL с условиями фильтрации событий.
filter: !vrl |
  .event.host == "198.51.100.191"

# Блок с правилами на языке VRL для нормализации полей события.
normalizer: !vrl |
  e = parse_json(.event) ?? {}
  . |= {
    "userId": e."user-identifier",
```

```
    "timestamp": parse_timestamp(e.datetime, "%d/%b/%Y:%H:%M:%S") ?? now(),
    "host": e.host,
    "method": e.method,
    "path": e.request,
    "protocol": e.protocol,
    "responseStatus": e.status,
    "responseBytes": e.bytes,
    "referer": e.referer,
    "userAgent": e."user-agent",
    "serverIP": "198.51.100.191",
    "severity": e.severity,
  }

  if e.status == "404" || e.status == "401" {
    .responseStatus = "Bad request"
  }

  if e.status == "404" || e.status == "401" {
    .severity = "High"
  }
```

Улучшенное правило "EnhancedNormalizationRule" представляет собой расширенную версию исходного правила. Оно разбирает недостающие поля сырого события и добавляет поле `severity`, которое определяет уровень угрозы в зависимости от статуса события. Если статус события соответствует "404" или "401", в поле `severity` устанавливается значение "High". В противном случае полю `severity` будет присвоено "null".

Таким образом, после применения правила "EnhancedNormalizationRule", нормализованное событие принимает следующий вид:

Событие, обработанное правилом "EnhancedNormalizationRule", JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"IvanIvanov\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\",\
"request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"status\":
\"404\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}",
  "host": "198.51.100.191",
  "method": "GET",
  "path": "/observability/metrics/production",
  "protocol": "HTTP/2.0",
  "referer": "https://example.url.com",
  "responseBytes": 22427,
  "responseStatus": "Bad request",
  "serverIP": "198.51.100.191",
  "severity": "High", # Добавлено поле severity, уровень угрозы установлен в значение
High.
  "timestamp": "2023-07-31T06:55:49Z",
  "userAgent": null,
  "userId": "IvanIvanov"
```

```
}
```

Использование глобальных функций

Если одинаковые преобразования данных нужно выполнять в нескольких правилах, вы можете создать глобальную функцию с необходимой логикой:

1. Создайте для указанного примера глобальную функцию со следующим исполняемым кодом и аргументами:

```
...
# Имя функции.
name: set_severity
...

# Основной исполняемый код.
source: !vrl |
  if status == "404" || status == "401" {
    "High"
  } else {
    null
  }

# Аргументы функции.
arguments:
  - keyword: status
    kind: bytes
...
```

2. Вызовите глобальную функцию в правиле нормализации:

```
...
.severity = set_severity(e.status)
```

3. Укажите в поле `setup_tests` искусственные данные для тестирования глобальной функции.

Правила агрегации

В данном разделе представлен обзор структуры, принципов работы, процесса создания и расширения конфигурации правил агрегации в системе. Управление правилами агрегации в системе осуществляется в разделе **Экспертиза**.

О правилах агрегации

Правило агрегации — это структурированный объект данных в формате RObject (.ro), который описывает логику объединения и анализа групп событий безопасности.

Структура правила агрегации включает определение условий фильтрации событий, параметров

группировки, условий начала и завершения агрегации, а также стратегий агрегации для каждого поля. Фильтрация событий осуществляется на основе заранее заданных критериев, определяющих, какие события будут включены в процесс агрегации. Условия начала и завершения агрегации определяют логику работы агрегационного окна, определяя, когда и как будет открываться и закрываться окно.

Важной частью правила является определение группировки событий, что позволяет собирать события в логически согласованные группы для последующей агрегации. Группировка может осуществляться по различным полям события, например, содержащим данные об IP-адресах, идентификаторах устройств или пользователях. Если указано несколько полей для группировки, то события будут группироваться по уникальным комбинациям значений этих полей.

Стратегии агрегации для каждого поля определяют, как будут объединяться значения соответствующих полей событий в процессе агрегации. Доступны различные стратегии, такие как суммирование, выбор максимального и минимального значений, различные формы конкатенации строк, такие как обычное соединение, соединение с новой строки или без разделителей, а также методы для работы с массивами, включая добавление всех значений, формирование массива уникальных значений и выбор самого длинного или короткого массива. Кроме того, существуют стратегии для сохранения значений из первого или последнего события в агрегации.

Работа с правилом агрегации

Доступные операции над правилом агрегации:

- Создание правила агрегации
- Изменение правила агрегации
- Удаление правила агрегации

Создание правила агрегации

Чтобы добавить правило агрегации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило агрегации**. Система отобразит окно создания правила агрегации.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля правила агрегации, чтобы определить логику его работы.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст правило агрегации и отобразит соответствующее уведомление. Новое правило появится в таблице раздела **Экспертиза**.



Чтобы правило агрегации стало доступно на конвейерах событий, его

> необходимо включить.

Вы можете создать черновик правила агрегации без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик правила и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение правила агрегации

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление правила агрегации

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура правила агрегации

Правило агрегации в системе представляет собой структурированный объект данных в формате RObject (.ro), который включает в себя ряд обязательных и опциональных полей для определения процесса агрегации.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила агрегации.	строка	да
`name`	Название правила агрегации.	строка	да
`type`	Тип элемента. Всегда имеет значение `aggregation_rule` для элементов экспертизы этого типа.	строка	да
`version`	Версия правила агрегации, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора правила агрегации.	строка	нет
`description`	Описание правила агрегации, содержащее дополнительную информацию о его назначении и функциональности.	строка	нет
`data_source`	Список источников данных, с которыми совместимо правило агрегации. Используется для классификации и быстрого поиска правил в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных правилу агрегации для классификации и быстрого поиска.	массив строк	нет
`filter`	Фильтр на языке VRL, определяющий условия, при которых события будут агрегироваться.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`group_by`	Поля, по которым будут группироваться агрегированные события. Если ничего не указано — все поступающие события будут объединяться в одну группу.	массив строк	нет
`starts_when`	Условие на языке VRL, при выполнении которого открывается новое агрегационное окно. Нельзя использовать одновременно с параметром `ends_when`.	строка	нет
`max_events`	Максимальное количество событий для агрегации.	число	нет
`expire_after_ms`	Время в миллисекундах, после которого агрегационное окно закрывается, если не поступают новые события.	число	нет
`flush_period_ms`	Интервал в миллисекундах для проверки и закрытия агрегационных окон.	число	нет
`merge_strategies`	Стратегии, определяющие, как будут объединяться значения полей из агрегированных событий.	объект	нет
`ends_when`	Условие на языке VRL, при выполнении которого агрегационное окно закрывается. Нельзя использовать одновременно с параметром `starts_when`.	строка	нет
`tests`	Блок тестирования правила агрегации, содержащий массив тестовых событий и условия для проверки срабатывания правила.	объект	нет



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

В блоках VRL используются правила экранирования строк VRL.

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, обрабатываемых правилом.	массив	да
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Опции `merge_strategies`

Значение, указанное в `merge_strategies`, определяет метод, с помощью которого значения из полей разных событий будут объединяться в ходе агрегации.

Опция	Описание
`array`	Создает массив, включающий все встречающиеся значения поля из агрегированных событий. Используется, когда необходимо сохранить все значения без исключения.
`concat`	Объединяет строковые значения в одну строку с разделением пробелами. Подходит для ситуаций, где необходимо сформировать единое текстовое представление из разных строковых значений.
`concat_newline`	Аналогично `concat`, но использует перенос строки в качестве разделителя. Это облегчает различение отдельных значений в одной длинной строке.
`concat_raw`	Соединяет строковые значения без какого-либо разделителя между ними. Эта стратегия полезна, когда необходимо сформировать непрерывную строку данных.
`discard`	Игнорирует все значения, кроме первого встретившегося. Полезно, когда важно сохранить только первоначальное значение поля из всей группы событий.
`flat_unique`	Формирует массив уникальных значений, исключая повторения. Используется для идентификации уникальных характеристик в группе событий.
`longest_array`	Выбирает массив максимальной длины из всех массивов, встречающихся в агрегированных событиях. Применяется для определения наиболее обширного набора данных.
`max`	Находит и сохраняет максимальное числовое значение среди всех агрегированных событий. Это может быть полезно для определения пиковых значений.
`min`	Аналогично `max`, но для определения минимального числового значения.
`retain`	Сохраняет значение из последнего события в агрегированной группе. Это полезно для отслеживания самых последних изменений.
`shortest_array`	Выбирает массив минимальной длины, что может быть полезно для определения наименее обширного набора данных в группе.
`sum`	Вычисляет сумму всех числовых значений поля. Эта стратегия широко используется для агрегации количественных данных, таких как счетчики или объемы.

Составление правила

Для составления правила агрегации в системе необходимо определить ключевые элементы структуры правила. Эти элементы включают поля метаданных и поля, управляющие логикой работы

правила.

1. Определите метаданные правила:
    - `id`: задайте уникальный идентификатор правила.
    - `name`: задайте название правила.
    - `version`: укажите версию правила, следуя стандартам Semantic Versioning.
    - `type`: определите тип правила как `aggregation_rule`.
    - `author` (опциональное): укажите имя и контактные данные автора.
    - `description`(опциональное): предоставьте краткое описание назначения и функциональности правила.
    - `data_source` (опциональное): укажите список совместимых с правилом источников данных для классификации и фильтрации.
    - `tags` (опциональное): добавьте теги для классификации и поиска правила.
2. Разработайте логику работы правила:
    - `filter` (обязательное): создайте условие для фильтрации событий.
    - `group_by` (обязательное): определите поля для группировки событий.
    - `starts_when` и `ends_when` (опциональные): настройте условия открытия и закрытия окна агрегации. В рамках правила допускается одновременное использование только одного из параметров.
    - `expire_after_ms` и `flush_period_ms` (опциональные): установите временные параметры для управления агрегационными окнами.
    - `max_events` (опциональное): укажите максимальное количество событий для агрегации.
    - `merge_strategies` (опциональное): выберите стратегии агрегации для полей событий.
3. Разработайте тесты для правила (опциональное):
    - `tests`: создайте тесты, включая сценарии и условия, для проверки работы правила.

Поля метаданных

Метаданные являются основой для идентификации и описания правила агрегации в системе. Они должны быть заполнены следующим образом:

- `id`: установите уникальный идентификатор правила. Этот идентификатор должен быть уникальным в рамках системы.
- `name`: укажите название правила.
- `version`: задайте версию правила, следуя принципам Semantic Versioning (например, 1.0.0).
- `type`: установите тип правила `aggregation_rule`.
- `author` (опциональное): укажите имя и контактные данные создателя правила.
- `description` (опциональное): предоставьте краткое описание назначения и функциональности правила.
- `data_source` (опциональное): укажите список совместимых с правилом источников данных.

- `tags` (опциональное): добавьте теги для классификации правила в системе.

Поле `filter`

Поле `filter` служит для определения критериев отбора событий, которые будут включены в процесс агрегации. Фильтрация основана на языке VRL, предоставляя гибкие возможности для анализа и обработки данных событий.

Пример:

```
filter: !vrl |
  .event_type == "login_failure" && .source_ip != "192.168.1.1"
```

В этом примере выбираются события с типом "login_failure", исключая события от конкретного IP-адреса.

Поле `group_by`

Поле `group_by` задает основу для группировки событий в рамках агрегации. Можно указать одно или несколько полей, по которым события будут группироваться, что позволяет более точно анализировать сходные или связанные данные. Если ничего не указано — все поступающие события будут объединяться в одну группу.

Пример:

```
group_by:
  - source_ip
  - user_agent
```

Здесь события группируются по IP-адресу источника и типу пользовательского агента.

Поле `starts_when`

`starts_when` определяет условие, при котором откроется новое окно агрегации. Это условие задает начальную точку для создания новой группы событий.



Параметр `starts_when` нельзя использовать одновременно с параметром `ends_when` в рамках одного правила агрегации.

Пример:

```
starts_when: !vrl |
  .event_type == "network_connection" && .port == 443
```

Агрегация начнется, когда обнаружится событие типа "network_connection" на порту 443.

Поле `ends_when`

`ends_when` определяет условие для закрытия окна агрегации. Это условие может быть основано на различных критериях, соответствующих потребностям безопасности и анализа.



Параметр `ends_when` нельзя использовать одновременно с параметром `starts_when` в рамках одного правила агрегации.

Пример:

```
ends_when: !vrl |
  .event_type == "network_disconnection"
```

Здесь агрегация завершается, когда появляется событие разъединения сети.

Поле `expire_after_ms`

`expire_after_ms` устанавливает максимальный период ожидания для агрегационного окна в миллисекундах. Это поле помогает контролировать продолжительность агрегации и гарантирует, что данные обрабатываются в разумные временные рамки.

Пример:

```
expire_after_ms: 30000
```

Агрегационное окно будет закрыто, если новые события не появятся в течение 30 секунд.

Поле `flush_period_ms`

`flush_period_ms` задает периодичность проверки и закрытия агрегационных окон. Этот параметр важен для обеспечения своевременной обработки данных и управления памятью.

Пример:

```
flush_period_ms: 10000
```

Система будет проверять и закрывать агрегационные окна каждые 10 секунд.

Поле `max_events`

`max_events` определяет максимальное количество событий, которое может быть включено в одно агрегационное окно. Этот параметр помогает контролировать объем данных в каждой агрегации.

Пример:

```
max_events: 20
```

Агрегационное окно будет закрыто после обработки 20 событий.

Поле `merge_strategies`

`merge_strategies` определяет стратегии обработки и объединения данных из различных событий в процессе агрегации. Эти стратегии позволяют гибко управлять данными, создавая информативные и ценные агрегированные записи.

Если для указанного поля не определена стратегия, будет использоваться поведение по умолчанию.

Поведение по умолчанию:

- Первое значение строкового поля сохраняется, а последующие значения отбрасываются.
- Для полей временных меток сохраняется первое значение, и добавляется новое поле `[имя-поля]_end` с временной меткой последнего полученного значения.
- Числовые значения суммируются.

Пример:

```
merge_strategies:
  bytes_sent: sum
  messages: concat_newline
```

Здесь значения поля `bytes_sent` суммируются, а строки в поле `messages` объединяются с разделителем в виде новой строки.

Поле `tests`

Поле `tests` включает набор тестов для проверки и верификации правила агрегации. Эти тесты помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Определите параметры теста:

- В `name` укажите название теста.
- В `events` создайте массив событий, которые будут использоваться для тестирования. Эти события должны имитировать поток реальных исходных событий, которые поступают на агрегатор.
- В `assertion` устанавливается условие или набор условий, которые должны быть выполнены в результате обработки тестовых событий правилом. Это может включать проверку конкретных полей в итоговых агрегированных событиях или подсчет количества таких событий.

Тестовые события

Тестовые события должны имитировать данные, которые в реальности поступают на агрегатор. Они должны соответствовать формату и структуре событий, обрабатываемых правилом агрегации и содержать поля, которые используются в правиле.

Условия проверки

Условия должны точно отражать ожидаемый результат применения правила к тестовым событиям. Это может быть проверка наличия определенных полей в событии, соответствие их значений ожидаемым или сравнение количества событий с предполагаемым исходом. Для более сложных правил могут потребоваться более сложные утверждения, которые могут включать логические операции и сравнения.

Пример:

```
tests:
  - name: AggregationTest
    events:
      - { "event_type": "login_failure", "source_ip": "192.168.1.1" }
      - { "event_type": "login_failure", "source_ip": "192.168.1.2" }
    assertion: !vrl |
      assert_eq!(.aggregated_event_count, 2)
```

Здесь проверяется, что правило агрегирует два события типа "login_failure".

Структура агрегированного события

В результате выполнения агрегации создается агрегированное событие. Помимо полей, которые явно определяются правилом агрегации, структура агрегированного события содержит ряд служебных полей, заполняемых по умолчанию.

Ключ поля	Тип данных	Описание
`id`	UUID	ID агрегированного события. Генерируется автоматически при создании события.
`sourceIp`	IPv6	IP-адрес источника базового события. По умолчанию записывается значение для первого базового события в агрегации.
`tenantId`	LCString	ID тенанта, к которому относится агрегированное событие.
`art`	DateTime	Дата и время получения базового события. По умолчанию записывается значение для первого базового события в агрегации.
`raw`	String	Текст базового события. По умолчанию записывается значение для первого базового события в агрегации.
`collectorId`	LCString	ID коллектора, на котором создается агрегированное событие.
`timestamp`	DateTime	Дата и время создания агрегированного события.
`aggregationRuleName`	LCString	Название правила агрегации, обработавшего базовое событие.
`aggregationRuleId`	LCString	ID правила агрегации, обработавшего базовое событие.

Ключ поля	Тип данных	Описание
`type`	Enum	Тип события. По умолчанию устанавливается значение "2", соответствующее типу "агрегированное событие".
`cnt`	UInt32	Количество базовых событий, на основе которых создается агрегированное событие.
`baseEventIds`	Массив UUID	Список ID базовых событий, на основе которых создается агрегированное событие.
`baseEventTimestamps`	Массив DateTime	Список дат и времен получения базовых событий, на основе которых создается агрегированное событие.
`groupedBy`	Массив LCString	Список полей модели событий, по которому группируются базовые события при агрегации.



Поля `id`, `tenantId`, `collectorId`, `timestamp`, `type`, `cnt`, `baseEventIds`, `baseEventTimestamps` заполняются системой автоматически. Их правила заполнения не могут быть изменены.

Поля `groupedBy`, `aggregationRuleId` и `aggregationRuleName` заполняются на основе настроек, заданных в правиле агрегации.

При необходимости вы можете изменить правила заполнения остальных полей. Для этого задайте соответствующие правила в блоке `merge_strategies` правила агрегации.

Прочие служебные и регулярные поля модели события заполняются в соответствии с настройками, заданными в блоке `merge_strategies` правила агрегации.

Пример правила агрегации

Правило агрегации, схема RObject

```
# Уникальный идентификатор правила.
id: example/network_activity_rule

# Название правила.
name: NetworkActivityAggregationRule

# Тип элемента экспертизы.
type: aggregation_rule

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Агрегация событий сетевой активности для анализа трафика

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>
```

```
# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги, связанные с правилом.
tags:
  - network
  - traffic_analysis

# Отбрасывает события, которые не соответствуют критериям.
filter: !vrl |
  # Фильтрует события, не относящиеся к категории "Network" или имеющие низкий уровень
серьезности.
  .category == "Network" &&
  to_int!(.severity) > 3

# Группировка событий по источнику и назначению.
group_by:
  - src_ip
  - dest_ip

# Условие открытия агрегационного окна.
start_when: !vrl |
  # Начинаем агрегацию при обнаружении события начала соединения.
  .event_type == "connection_start"

# Максимальное количество событий.
max_events: 5

# Время ожидания последнего события (миллисекунды).
expire_after_ms: 60000

# Интервал проверки окон агрегации (миллисекунды).
flush_period_ms: 2000

# Стратегии агрегации значений для каждого указанного поля.
merge_strategies:
  bytes_sent: sum
  bytes_received: sum
  severity: max

# Тесты для проверки работы правила.
tests:
 - name: Network Activity Test
   events:
   - {"category":"Network", "severity":5, "event_type":"connection_start",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":1500,
```

```
"bytes_received":2000}
   - {"category":"Network", "severity":4, "event_type":"data_transfer",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":3000,
"bytes_received":4000}
   - {"category":"Network", "severity":5, "event_type":"connection_end",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":500, "bytes_received":600}
   assertion: !vrl |
     # Проверяем, что агрегированные значения соответствуют ожиданиям.
     assert_eq!(.[0], {
       "category": "Network",
       "event_type": "connection_start",
       "src_ip": "192.0.2.10",
       "dest_ip": "192.0.2.20",
       "bytes_sent": 5000,
       "bytes_received": 6600,
       "severity": 5
     })
```

Интерпретация

Правило агрегации "NetworkActivityAggregationRule" служит для агрегации сетевых событий, соответствующих передаче данных в рамках одного сетевого соединения.

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `aggregation_rule` указывает на то, что это правило агрегации.
- `version`: версия правила.
- `description`: описывает цель правила — "Агрегация событий сетевой активности для анализа трафика".
- `author`: создатель правила.
- `data_source`: указывает источник правила:
  - `platform`: платформа — Windows.
  - `source`: источник событий — Sysmon.
  - `events`: события `EventID_1` и `EventID_2`.
- `tags`: теги `network` и `traffic_analysis` классифицируют правило по его функциональности и области применения.

Фильтрация событий (filter):

- `filter`: фильтрует события по категории "Network" и уровню серьезности более 3. Это сужает область применения правила только к событиям, которые удовлетворяют условиям фильтрации.

Группировка событий (group_by):

- `group_by`: группирует события по полям `src_ip` и `dest_ip`. Это позволяет анализировать

взаимодействия между конкретными источниками и назначениями в сети.

Условие начала агрегации (starts_when):

- `starts_when`: определяет начало агрегации при появлении событий типа "connection_start", что является признаком начала нового сетевого соединения.

Параметры агрегационного окна (max_events, expire_after_ms, flush_period_ms):

- `max_events`: ограничивает количество событий в агрегации до 5. После достижения этого количества, агрегационное окно закрывается и генерируется агрегированное событие.
- `expire_after_ms`: устанавливает максимальный период ожидания последнего события в агрегационном окне в 60000 миллисекунд (1 минута).
- `flush_period_ms`: задает частоту проверки и закрытия агрегационных окон в 2000 миллисекунд (2 секунды).

Стратегии агрегации (merge_strategies):

- `merge_strategies`: определяет, как будут объединяться поля событий. Для "bytes_sent" и "bytes_received" применяется стратегия суммирования (`sum`).

Тестирование правила (tests):

- `name`: `Network Activity Test` — название теста.
- `events`: описывает ряд тестовых событий, имитирующих реальный сетевой трафик, для проверки эффективности правила.
- `assertion`: проверяет, что правило корректно агрегирует события и вычисляет суммы переданных и полученных байтов. Это подтверждает, что правило работает как предполагалось и агрегирует данные согласно заданным критериям.

Обработка событий правилом агрегации "NetworkActivityAggregationRule"

Проанализируем работу правила агрегации "NetworkActivityAggregationRule", фокусируясь на процессе обработки событий.

Представим, что на агрегатор поступает поток сетевых событий, которые соответствуют критериям, заданным в фильтре `filter` правила. Эти события содержат данные о типе сетевого соединения, IP-адресах источника и назначения, а также объеме переданных и полученных данных.

```
{
  "category": "Network",
  "severity": 5,
  "event_type": "connection_start",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.20",
  "bytes_sent": 1500,
  "bytes_received": 2000
}
```

По мере поступления этих событий, они фильтруются правилом "NetworkActivityAggregationRule". Правило агрегирует события, группируя их по IP-адресам источника и назначения (`group_by`). Таким образом, события, исходящие от одной пары IP-адресов, объединяются в одно агрегированное событие.

Агрегация инициируется, когда обнаруживается событие типа `connection_start` (`starts_when`). Правило продолжает агрегировать события до тех пор, пока не будет достигнуто максимальное количество событий (`max_events`) или не истечет время ожидания (`expire_after_ms`).

Агрегированное событие включает в себя сумму значений `bytes_sent` и `bytes_received` для всех событий в группе (`merge_strategies`).

```
{
  "event_type": "aggregated_network_activity",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.20",
  "total_bytes_sent": 5000,
  "total_bytes_received": 6600
}
```

Данное агрегированное событие отражает общую активность между двумя конкретными IP-адресами, позволяя идентифицировать аномалии или необычную активность в сети.

Расширение конфигурации существующего правила

Рассмотрим сценарий, в котором необходимо расширить функциональность правила агрегации "NetworkActivityAggregationRule", чтобы улучшить анализ сетевого трафика и повысить точность агрегации событий.

▼ *Исходное правило агрегации "NetworkActivityAggregationRule", схема RObject*

```
# Уникальный идентификатор правила.
id: example/network_activity_rule

# Название правила.
name: NetworkActivityAggregationRule

# Тип элемента экспертизы.
type: aggregation_rule

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Агрегация событий сетевой активности для анализа трафика

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
```

```
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги, связанные с правилом.
tags:
  - network
  - traffic_analysis

# Отбрасывает события, которые не соответствуют критериям.
filter: !vrl |
  # Фильтрует события, не относящиеся к категории "Network" или имеющие низкий
уровень серьезности.
  .category == "Network" &&
  to_int!(.severity) > 3

# Группировка событий по источнику и назначению.
group_by:
  - src_ip
  - dest_ip

# Условие открытия агрегационного окна.
start_when: !vrl |
  # Начинаем агрегацию при обнаружении события начала соединения.
  .event_type == "connection_start"

# Максимальное количество событий.
max_events: 5

# Время ожидания последнего события (миллисекунды).
expire_after_ms: 60000

# Интервал проверки окон агрегации (миллисекунды).
flush_period_ms: 2000

# Стратегии агрегации значений для каждого указанного поля.
merge_strategies:
  bytes_sent: sum
  bytes_received: sum
  severity: max

# Тесты для проверки работы правила.
tests:
 - name: Network Activity Test
   events:
   - {"category":"Network", "severity":5, "event_type":"connection_start",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":1500,
"bytes_received":2000}
```

```
    - {"category":"Network", "severity":4, "event_type":"data_transfer",
  "src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":3000,
  "bytes_received":4000}
    - {"category":"Network", "severity":5, "event_type":"connection_end",
  "src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":500,
  "bytes_received":600}
    assertion: !vrl |
      # Проверяем, что агрегированные значения соответствуют ожиданиям.
      assert_eq!(.[0], {
        "category": "Network",
        "event_type": "connection_start",
        "src_ip": "192.0.2.10",
        "dest_ip": "192.0.2.20",
        "bytes_sent": 5000,
        "bytes_received": 6600,
        "severity": 5
      })
```

Для этого добавим новые условия группировки и стратегии агрегации, а также дополнительные параметры для более тонкой настройки агрегации событий.

Пример расширенного правила "EnhancedNetworkActivityAggregationRule", схема RObject

```
# Уникальный идентификатор правила.
id: example/network_activity_rule_enhanced

# Название правила.
name: EnhancedNetworkActivityAggregationRule

# Тип элемента экспертизы.
type: aggregation_rule

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Расширенная агрегация событий сетевой активности для детального анализа
трафика

# Имя и контактные данные автора правила.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги, связанные с правилом.
```

```
tags:
  - network
  - advanced_traffic_analysis

#  Условия для фильтрации событий.
filter: !vrl |
  # Фильтрация событий по категории "Network" и высокому уровню серьезности, а также
специфическим характеристикам запросов.
  .category == "Network" && .severity > 3 && match(.user_agent, "^Mozilla")

# Расширенные параметры группировки событий.
group_by:
  - src_ip
  - dest_ip
  - user_agent
  - http_method

# Условие открытия агрегационного окна.
start_when: !vrl |
  # Агрегация начинается при обнаружении начала соединения или определенных типов
HTTP-запросов.
  .event_type == "connection_start" || includes(["POST", "GET"], .http_method)

# Увеличенное максимальное количество событий.
max_events: 10

# Измененное время ожидания последнего события (миллисекунды).
expire_after_ms: 120000

# Интервал проверки окон агрегации (миллисекунды).
flush_period_ms: 3000

# Расширенные стратегии агрегации значений.
merge_strategies:
  bytes_sent: sum
  bytes_received: sum
  request_paths: longest_array

# Тесты для проверки работы расширенного правила.
tests:
 - name: Enhanced Network Activity Test
   events:
   - {"category":"Network", "severity":5, "event_type":"connection_start",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "user_agent": "Mozilla/5.0",
"http_method": "POST", "bytes_sent":2000, "bytes_received":3000}
   - {"category":"Network", "severity":4, "event_type":"data_transfer",
"src_ip":"192.0.2.10", "dest_ip":192.0.2.20", "user_agent": "Mozilla/5.0",
"http_method": "GET", "bytes_sent":4000, "bytes_received":5000}
   assertion: !vrl |
     # Проверяем, что агрегированные значения соответствуют ожиданиям.
     assert_eq!(.[0], {
```

```
        "category": "Network",
        "event_type": "connection_start",
        "src_ip": "192.0.2.10",
        "dest_ip": "192.0.2.20",
        "user_agent": "Mozilla/5.0",
        "http_method": "POST",
        "bytes_sent": 6000,
        "bytes_received": 8000,
        "request_paths": "/api/load"
      })
```

Изменения, внесенные в улучшенное правило "EnhancedNetworkActivityAggregationRule":

1. Дополнительные параметры группировки и стратегии агрегации:
    - Добавлена группировка по дополнительным полям, таким как `user_agent` или `http_method`, для более детального анализа сетевого трафика.
    - Введены новые стратегии агрегации, например, `longest_array` для поля `request_paths`, чтобы отслеживать наиболее часто используемые пути запросов.
2. Изменение параметров `max_events` и `expire_after_ms`:
    - Увеличение значения `max_events` для обработки большего количества событий в одном агрегационном окне.
    - Изменение `expire_after_ms` для оптимизации времени ожидания агрегации событий, учитывая характер трафика.
3. Тестовая проверка:
    - Добавлены тестовые сценарии с учетом новых параметров группировки и стратегий агрегации для проверки правила на способность адекватно агрегировать и анализировать более сложные сетевые события.

Интерпретация правила "EnhancedNetworkActivityAggregationRule"

Рассмотрим механику работы усовершенствованного правила агрегации "EnhancedNetworkActivityAggregationRule", фокусируясь на логике обработки и агрегации событий.

Правило "EnhancedNetworkActivityAggregationRule" предназначено для более глубокого анализа сетевых событий, используя расширенные критерии фильтрации и группировки.

Основные параметры правила:

- `filter`: фильтрация событий основывается на категории "Network", уровне серьезности событий и определенных характеристиках, таких как соответствие `user_agent` шаблону регулярного выражения "Mozilla".
- `group_by`: агрегация событий выполняется по полям `src_ip`, `dest_ip`, `user_agent` и `http_method`, что позволяет проводить многофакторный анализ сетевой активности.
- `starts_when`: условие начала агрегации активируется при обнаружении события начала соединения или определенных типов HTTP-запросов.
- `merge_strategies`: применяются стратегии агрегации `sum` для полей `bytes_sent` и `bytes_received` и

`longest_array` для поля `request_paths`.

Логика агрегации:

- При срабатывании условий `filter` и `starts_when`, события группируются по определенным IP-адресам источника и назначения, типам запросов и агентам пользователя.
- Агрегированные события включают общее количество переданных и полученных байтов (`bytes_sent` и `bytes_received`) и наиболее часто встречающиеся пути запросов (`request_paths`).

Тестирование правила:

Тестовые сценарии "Enhanced Network Activity Test" включают события, соответствующие критериям фильтрации и группировки. Они направлены на проверку способности правила корректно агрегировать данные, учитывая новые параметры группировки и агрегации.

Предположим, что события, поступающие на агрегатор, на котором установлено правило агрегации, имеют следующий вид:

Тестовое событие, JSON

```
{
  "category": "Network",
  "severity": 5,
  "event_type": "connection_start",
  "src_ip": "192.0.2.10",
  "dest_ip": "192.0.2.20",
  "user_agent": "Mozilla/5.0",
  "http_method": "POST",
  "bytes_sent": 2000,
  "bytes_received": 3000
}
```

Тогда, ожидаемый результат агрегации примет следующий вид:

Агрегированное событие, JSON

```
{
  # Служебные поля.
  "id": "e1336694-b3d6-4e02-9f20-92f642bb64fa", # ID агрегированного события.
  "sourceIp": "192.0.2.10", # IP-адрес источника события.
  "tenantId": "99311ef1-32a1-4b2d-9d38-7613be5e4c2b", # ID тенанта.
  "art": "15.03.2024 15:25:05", # Дата и время получения первого события.
  "raw":
"{\"category\":\"Network\",\"severity\":5,\"event_type\":\"connection_start\",\"src_ip
\":\"192.0.2.10\",\"dest_ip\":\"192.0.2.20\",\"user_agent\":\"Mozilla/5.0\",\"http_met
hod\":\"POST\",\"bytes_sent\":2000,\"bytes_received\":3000}", # Текст первого события.
  "collectorId": "891d9a49-8417-4ce9-abf4-184b0c093e46", # ID коллектора.
  "timestamp": "15.03.2024 15:45:10", # Дата и время создания агрегированного события.
  "aggregationRuleName": "example/network_activity_rule_enhanced", # Название правила
агрегации.
  "aggregationRuleId": "EnhancedNetworkActivityAggregationRule", # ID правила
```

```
агрегации.
  "type": "2", # Тип события - "агрегированное событие".
  "cnt": "5", # Количество событий в агрегации.
  "baseEventIds": "41627060-c610-494d-834c-0f6c3223f0eb, b5db4311-b978-45b5-8790-
0b02834bf479, dd3d4ad4-9de9-478e-be65-24eb3e70abd8, 34a9661f-f460-4285-b11b-
8d2388811819, 933a0aaa-85eb-477a-a1a4-285478a42f5a", # Список ID событий в агрегации.
  "baseEventTimestamps": "15.03.2024 15:25:05, 15.03.2024 15:28:35, 15.03.2024
15:33:59, 15.03.2024 15:37:02, 15.03.2024 15:41:17", # Список дат и времен получения
событий в агрегации.
  "groupedBy": "src_ip, dest_ip, user_agent, http_method", # Поля группировки в
агрегации.
  "correlationRuleName": "", # Название корреляционного правила.
  "correlationRuleId": "", # ID корреляционного правила.
  "correlationSeverity": "", # Уровень угрозы корреляционного события.

  # Регулярные поля.
  "category": "Network",
  "event_type": "connection_start",
  "src_ip": "192.0.2.10",
  "dest_ip": "192.0.2.20",
  "user_agent": "Mozilla/5.0",
  "http_method": "POST",
  "total_bytes_sent": 6000,
  "total_bytes_received": 8000,
  "request_paths": "/api/load"
}
```



Подробное описание служебных полей представлено в разделе Универсальная модель события 2.0.

Таким образом создается многоуровневая логика агрегации для правила "EnhancedNetworkActivityAggregationRule", где события должны соответствовать заданным критериям фильтрации и группировки. Эти критерии включают не только IP-адреса источника и назначения, но также типы запросов и агенты пользователя, что позволяет проводить более детализированный анализ трафика.

Правила сегментации

В данном разделе представлен обзор структуры, принципов работы и процесса создания правил сегментации в системе. Управление правилами сегментации в системе осуществляется в разделе **Экспертиза**.

О правилах сегментации

Правило сегментации в системе представляет собой структурированный объект данных, определяющий логику для классификации и обработки корреляционных событий, на основе которых правило сегментации создает оповещения. Каждое правило сегментации, оформленное в формате RObject (.ro), содержит набор параметров для группировки событий по определенным критериям, таким как тип события, источник или другие атрибуты.

Основная функция правил сегментации заключается в управлении потоком корреляционных событий, обеспечивая их агрегацию, классификацию и преобразование в оповещения согласно заданным условиям.

Параметры сегментации позволяют задать условия, по которым события будут собираться в оповещении до достижения определенного количества или до истечения заданного временного интервала. Например:

- Можно задать условия для определения количества событий от конкретного источника, которые должны будут собираться в оповещении в течение установленного временного интервала. По истечении этого временного интервала, окно сегментации будет закрыто и последующие корреляционные события будут отправляться в новое оповещение.
- Можно задать условия для накопления в оповещении определенного количества событий от конкретного источника. По достижении этого количества, окно сегментации будет закрыто и последующие корреляционные события будут отправляться в новое оповещение.

В системе правила сегментации используются в рамках сервиса оповещений конвейера обработки событий.

Применение правил сегментации

Правила сегментации задают логику разделения и классификации корреляционных событий для формирования оповещений на основе анализа, выполненного правилами корреляции. Правила сегментации применяются к потоку событий на конвейере обработки в рамках элемента сервис оповещений, который использует эти правила для дополнительной обработки и группировки корреляционных событий, позволяя задавать многоуровневые условия для создания оповещений.

Особенностью сервиса оповещений является его способность ассоциировать одно правило сегментации с несколькими правилами корреляции, что позволяет агрегировать разнородные корреляционные события в рамках единой логики сегментации. Такая модель обеспечивает унифицированную обработку событий, исходящих от различных источников. Вместе с тем, правило корреляции, уже связанное с определенным правилом сегментации, не может быть ассоциировано с другим правилом сегментации, что исключает возможность перекрестных или конфликтующих связей между правилами.

Система поддерживает одновременное использование множества правил сегментации в рамках сервиса оповещений, что позволяет создавать многоуровневую логику сегментации и обработки событий.

Работа с правилом сегментации

Доступные операции над правилом сегментации:

- Создание правила сегментации
- Изменение правила сегментации
- Удаление правила сегментации

Создание правила сегментации

Чтобы добавить правило сегментации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Правило сегментации**. Система отобразит окно создания правила сегментации.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля правила сегментации, чтобы определить логику его работы.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст правило сегментации и отобразит соответствующее уведомление. Новое правило появится в таблице раздела **Экспертиза**.



Чтобы правило сегментации стало доступно на конвейерах событий, его необходимо включить.

Вы можете создать черновик правила сегментации без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик правила и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение правила сегментации

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление правила сегментации

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура правила сегментации

Правило сегментации в системе представляет собой структурированный объект данных в формате RObject (.ro), который включает в себя ряд обязательных и опциональных полей для определения процесса сегментации.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила сегментации.	строка	да
`name`	Название правила сегментации.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`type`	Тип элемента. Всегда имеет значение `segmentation_rule` для элементов экспертизы этого типа.	строка	да
`version`	Версия правила сегментации, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора правила сегментации.	строка	нет
`description`	Описание правила сегментации, содержащее дополнительную информацию о его назначении и функциональности.	строка	нет
`data_source`	Список источников данных, с которыми совместимо правило сегментации. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных правилу сегментации для классификации и быстрого поиска.	массив строк	нет
`group_by`	Поля для группировки событий в рамках процесса сегментации. Каждая группа агрегируется независимо. Если ничего не указано — все поступающие события будут объединяться в одну группу.	массив строк	нет
`threshold`	Числовое значение, задающее максимальное количество событий в группе. Достижение этого количества инициирует создание нового оповещения. Если указано вместе с полем `timeout`, то оповещение создается по тому условию, которое сработает раньше.	число	нет
`timeout`	Числовое значение, задающее максимальный временной интервал в секундах для группы. По истечении этого времени создается новое оповещение, даже если количество событий не достигло `threshold`. Если указано вместе с полем `threshold`, то оповещение создается по тому условию, которое сработает раньше.	число	нет

Поле	Описание	Тип данных	Обязательное поле
`alert_name`	VRL-выражение для определения логики формирования имени оповещения (например из частей имен событий). Если поле не определено, оповещению будет автоматически присвоено название правила корреляции, которое первым привело к его срабатыванию. Выполняется только при создании оповещения (при обновлении существующего оповещения не выполняется).	строка	нет
`tests`	Блок тестирования правила сегментации, содержащий массив тестовых сценариев и условия для проверки логики сегментации.	объект	нет



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

В блоках VRL используются правила экранирования строк VRL.

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
```

```
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, обрабатываемых правилом.	массив	да
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Составление правила

Для составления правила сегментации в системе необходимо определить ключевые элементы структуры правила. Эти элементы включают поля метаданных и поля, управляющие логикой работы правила.

1. Определите метаданные правила:
    - `id`: задайте уникальный идентификатор правила.
    - `name`: задайте название правила.
    - `version`: укажите версию правила, следуя стандартам Semantic Versioning.
    - `type`: определите тип правила как `segmentation_rule`.
    - `author` (опциональное): укажите имя и контактные данные автора.

- `description` (опциональное): предоставьте краткое описание назначения и функциональности правила.
- `data_source` (опциональное): укажите список совместимых с правилом источников данных для классификации правила в системе.
- `tags` (опциональное): добавьте теги для классификации и поиска правила.

2. Разработайте логику работы правила:



Для корректной работы, в конфигурации правила сегментации необходимо задать минимум одно из полей: `group_by`, `threshold`, или `timeout`.

- `group_by` (опциональное): определите поля для группировки событий.
- `threshold`(опциональное): установите максимальное количество событий в группе оповещения. По достижении порогового значения, система создаст новое оповещение.
- `timeout`(опциональное): установите максимальное время в секундах для группы событий, после чего создается новое оповещение.
- `alert_name`(опциональное): используйте VRL-выражение, чтобы определить логику формирования имени оповещения, которое будет присвоено при создании нового оповещения на основе сегментации. Если поле не определено, оповещению будет автоматически присвоено название правила корреляции, которое первым привело к его срабатыванию. Выполняется только при создании оповещения (при обновлении существующего оповещения не выполняется).

3. Разработайте тесты для правила (опциональное):
   - `tests`: создайте тесты, включая сценарии и условия, для проверки работы правила. Это включает определение потока тестовых событий и использование VRL-выражений для проверки результатов сегментации.

Поля метаданных

Метаданные являются основой для идентификации и описания правила сегментации в системе. Они должны быть заполнены следующим образом:

- `id`: установите уникальный идентификатор правила. Этот идентификатор должен быть уникальным в рамках системы.
- `name`: укажите название правила.
- `version`: задайте версию правила, следуя принципам Semantic Versioning (например, 1.0.0).
- `type`: установите тип правила `segmentation_rule`.
- `author` (опциональное): укажите имя и контактные данные создателя правила.
- `description` (опциональное): предоставьте краткое описание назначения и функциональности правила.
- `data_source` (опциональное): укажите список совместимых с правилом источников данных для классификации правила в системе.
- `tags` (опциональное): добавьте теги для классификации правила в системе.

Поле `group_by`



Для корректной работы, в конфигурации правила сегментации необходимо задать минимум одно из полей: `group_by`, `threshold`, или `timeout`.

Поле `group_by` определяет критерии для группировки корреляционных событий в процессе сегментации. Это основной компонент в логике сегментации, позволяющий классифицировать события на основе общих атрибутов, таких как IP-адрес источника, тип события или пользовательский идентификатор.

Для указания полей, по которым должна происходить группировка, используется массив строк. Каждая строка представляет собой имя поля в событии, по которому будет производиться группировка. Поле `group_by` может включать несколько таких имен.

Если поле не определено, все поступающие события будут объединяться в одну группу.

Пример:

```
group_by:
  - source_ip
  - event_type
```

В этом примере события группируются по IP-адресу источника и типу события. Это позволяет собирать и анализировать события, исходящие от одного источника и относящиеся к одному и тому же типу действия.

Поле `threshold`



Для корректной работы, в конфигурации правила сегментации необходимо задать минимум одно из полей: `group_by`, `threshold`, или `timeout`.

Поле `threshold` определяет пороговое количество событий в оповещении, при достижении которого система создаст новое оповещение. В случае одновременного использования с полем `timeout`, условием для генерации оповещения становится первое выполненное из двух — достижение порога событий или истечение времени ожидания.



Это значение рекомендуется определять на основе анализа типичного поведения системы и потенциальных угроз, чтобы выявлять действительные инциденты безопасности, минимизируя ложные срабатывания.

Если используется вместе с group_by, поле `threshold` указывает максимальное количество событий, которое может быть сгруппировано вместе на основе критериев, заданных в поле `group_by`, прежде чем будет сгенерировано новое оповещение.

Пример:

```
threshold: 100
```

В данном примере, система создает новое оповещение, если в группе накапливается 100 событий.

Поле `timeout`



Для корректной работы, в конфигурации правила сегментации необходимо задать минимум одно из полей: `group_by`, `threshold`, или `timeout`.

Поле `timeout` задает временной интервал в секундах, в течение которого события могут быть сгруппированы в один сегмент перед созданием оповещения. Этот интервал позволяет правилу сегментации реагировать не только на количество событий, но и на их временное распределение. Если поле указано вместе с полем `threshold`, то оповещение создается по тому условию, которое сработает раньше.

Поле `timeout` принимает целое число, представляющее максимальную продолжительность временного интервала, в течение которого группа событий, сформированная согласно критериям, заданным в поле group_by, должна накопиться для инициирования оповещения.

Пример:

```
timeout: 3600
```

В этом примере, если события, относящиеся к одной группе по критериям поля `group_by`, поступают в течение часа (3600 секунд), система создает оповещение.

Поле `alert_name`

Поле `alert_name` позволяет задать правило на языке VRL для формирования имени оповещения на основе атрибутов анализируемых событий. Выполняется только при создании оповещения, при обновлении существующего оповещения не выполняется.



Если поле `alert_name` не определено, оповещению будет автоматически присвоено название правила корреляции, которое привело к его срабатыванию.

Применение динамического именования оповещений позволяет идентифицировать источник и характер угрозы по названию оповещения.



При составлении выражения для `alert_name` рекомендуется использовать такие атрибуты событий, которые наиболее точно отражают суть инцидента. Это могут быть IP-адреса, типы событий, идентификаторы пользователей или другие ключевые данные.

Поле `alert_name` принимает строку, формируемую с использованием синтаксиса VRL, что позволяет включить в имя оповещения значения из полей событий, участвующих в сегментации.

Пример:

```
alert_name: !vrl |
    "ALERT_" + upcase!(.event_type) + "_" + to_string!(.source_ip)
```

В данном примере имя оповещения будет составлено из префикса `ALERT_`, типа события в верхнем регистре, и IP-адреса источника события.



В VRL-выражении в поле `alert_name` допустимо обращаться только к тем полям события, которые существуют в его структуре. Так, в примере выше подразумевается, что в структуре события присутствуют поля `event_type` и `source_ip`.

В противном случае поле `alert_name` будет проигнорировано и оповещению будет присвоено название породившего его правила корреляции.

Тестирование правила

Тесты в правиле сегментации позволяют проверить его способность корректно группировать и классифицировать события по заданным параметрам.

Тестирование правила сегментации обеспечивает верификацию логики группировки событий и создания оповещений на их основе, гарантируя точность и эффективность в выявлении и реагировании на потенциальные угрозы.

Поле `tests`

Поле `tests` включает набор тестов для проверки и верификации правила сегментации. Эти тесты помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Чтобы создать тестовый сценарий, в поле `tests` определите следующие параметры:

- В `name` укажите название теста.
- В `events` создайте массив событий, которые будут использоваться для тестирования. Эти события должны имитировать поток реальных исходных событий, которые поступают в сервис сегментации.
- В `assertion` установите условие или набор условий, которые должны быть выполнены в результате обработки тестовых событий правилом. Это может включать проверку конкретных полей в итоговых сегментированных событиях или подсчет количества таких событий.

Тестовые события

Тестовые события должны имитировать данные, которые в реальности поступают в сервис сегментации. Они должны соответствовать формату и структуре событий, обрабатываемых правилом сегментации, и содержать поля, которые используются в правиле.

Условия проверки

Условия должны точно отражать ожидаемый результат применения правила к тестовым событиям. Это может быть проверка наличия определенных полей в событии, соответствие их значений ожидаемым, или сравнение количества событий с предполагаемым исходом. Для более сложных правил могут потребоваться более сложные утверждения, которые могут включать логические операции и сравнения.

Пример:

```
tests:
  - name: "Тестирование сегментации для DDoS атак"
    events:
      - { source_ip: "192.168.1.1", event_type: "DDoS Detection", datetime: "2023-09-
28T12:45:00Z" }
      - { source_ip: "192.168.1.1", event_type: "DDoS Detection", datetime: "2023-09-
28T12:46:00Z" }
      - { source_ip: "192.168.1.2", event_type: "DDoS Detection", datetime: "2023-09-
28T12:47:00Z" }
    assertion: !vrl |
      assert_eq!(length(.), 2)
      assert_eq!(.[0].group_by.source_ip, "192.168.1.1")
      assert_eq!(.[0].alert_name, "DDoS_ALERT_192.168.1.1")
      assert_eq!(.[1].group_by.source_ip, "192.168.1.2")
      assert_eq!(.[1].alert_name, "DDoS_ALERT_192.168.1.2")
```

В данном тестовом сценарии проверяется работа правила сегментации "DDoSAlertsSegmentation", направленного на выявление и классификацию корреляционных событий по IP-адресам источника. Тест включает имитацию трех событий, два из которых имеют одинаковый IP-адрес. Ожидается, что правило сегментации сгруппирует эти события и сформирует два оповещения: одно для событий с IP-адресом 192.168.1.1 и второе для события с IP-адресом 192.168.1.2.

Пример правила сегментации

Правило сегментации, схема RObject

```
# Уникальный идентификатор правила сегментации.
id: segmentation/ddos_alerts

# Название правила сегментации.
name: DDoSAlertsSegmentation.

# Тип элемента экспертизы.
type: segmentation_rule

# Версия правила сегментации.
version: 1.0.0

# Описание назначения правила сегментации.
description: Сегментация и классификация оповещений для атак типа DDoS.

# Имя и контактные данные автора правила сегментации.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
```

```
    - logs/default.log
    - logs/somelog.log

# Теги, связанные с правилом сегментации.
tags:
  - ddos
  - alert
  - segmentation

# Группирует корреляционные события по указанным полям. Если не указано, все события
помещаются в единую группу.
group_by:
  - source_ip

# Задает максимальное количество корреляционных событий в группе оповещения.
# При достижении максимального значения создается новое оповещение.
threshold: 1

# Задает максимальное время жизни корреляционных событий в группе.
# При достижении максимального значения создается новое оповещение.
timeout: 60

# Код VRL для формирования имени оповещения (например из частей имен событий).
# Выполняется только при создании оповещения (при обновлении существующего оповещения
не выполняется).
# Если поле не определено, оповещению будет автоматически присвоено название правила
корреляции, которое первым привело к его срабатыванию.
alert_name: !vrl |
  "DDoS_ALERT_" + to_string!(.source_ip)

# Тесты для проверки работы правила сегментации.
tests:
  - name: High traffic from single source IP
    events:
      - { type: "correlation_rule", severity: "high", event_type: "DDoS Detection",
source_ip: "198.51.100.1", datetime: "2023-09-28T12:45:00Z" }
      - { type: "correlation_rule", severity: "high", event_type: "DDoS Detection",
source_ip: "198.51.100.2", datetime: "2023-09-28T12:47:00Z" }
      - { type: "correlation_rule", severity: "high", event_type: "DDoS Detection",
source_ip: "198.51.100.1", datetime: "2023-09-28T12:49:00Z" }
    assertion: !vrl |
      assert_eq!(length(.), 3)
      assert_eq!(.[0].group_by.source_ip, "198.51.100.1")
      assert_eq!(.[0].name, "DDoS_ALERT_198.51.100.1")
      assert_eq!(.[1].group_by.source_ip, "198.51.100.2")
      assert_eq!(.[1].name, "DDoS_ALERT_198.51.100.2")
```

Интерпретация

Правило сегментации "DDoSAlertsSegmentation" создано для обработки и классификации

корреляционных событий, генерируемых правилом "DDoSDetectionRule", целью которого является выявление DDoS-атак на основе анализа потока событий.

Метаданные правила сегментации:

- `id`: уникальный идентификатор правила сегментации в системе.
- `name`: название правила сегментации, отражающее его назначение, — сегментация оповещений, связанных с DDoS-атаками.
- `type`: тип элемента экспертизы указывает, что это правило относится к категории правил сегментации.
- `version`: версия правила сегментации, позволяющая отслеживать его итерации и обновления.
- `description`: краткое описание назначения правила сегментации — классификация и сегментация оповещений для атак DDoS.
- `author`: автор правила сегментации и его контактные данные.
- `tags`: метки, связанные с правилом, такие как ddos, alert, segmentation, помогающие в классификации и поиске правила.

Параметры сегментации:

- `group_by`: указывает на поля событий, которые используются для группировки корреляционных событий. В данном случае, это `source_ip`, что позволяет сегментировать события по уникальным источникам IP.
- `threshold`: количество событий от одного источника, при достижении которого будет сгенерировано новое оповещение.
- `timeout`: временной интервал в секундах, в течение которого агрегируются события для сегментации.
- `alert_name`: код VRL для формирования имени оповещения на основе атрибутов события. В данном случае, IP-адреса источника. Если поле не определено, оповещению будет автоматически присвоено название правила корреляции, которое привело к его срабатыванию.

Тестирование правила сегментации "DDoSAlertsSegmentation":

Поле `tests` в правиле сегментации "DDoSAlertsSegmentation" предназначено для верификации и валидации логики правила путем имитации реальных сценариев обработки корреляционных событий. Это позволяет убедиться, что логика сегментации работает корректно и соответствует заранее определенным требованиям к обработке и классификации событий.

В данном случае, тестовый блок содержит следующие ключевые компоненты:

- `name`: имя теста, в данном случае "High traffic from single source IP", что указывает на цель теста — проверить способность правила сегментации агрегировать и классифицировать события, характерные для DDoS-атак.
- `events`: перечисление событий, которые имитируют корреляционные события, сгенерированные правилом обнаружения DDoS. Эти события содержат различные `source_ip` и метаданные, характерные для событий обнаружения DDoS, такие, как тип события "DDoS Detection", уровень серьезности "high", и временная метка.

- `assertion`: содержит скрипт на языке VRL, предназначенный для дополнительной валидации атрибутов каждого сгенерированного оповещения. В частности, проверяется, что имя оповещения соответствует ожидаемому формату, основанному на IP-адресе источника события.

Таким образом, поле `tests` обеспечивает комплексную проверку правила сегментации "DDoSAlertsSegmentation", демонстрируя его способность идентифицировать и классифицировать события, ассоциированные с DDoS-атаками, и соответственно генерировать оповещения для дальнейшего анализа и реагирования.

Схемы активных списков

В данном разделе представлен обзор структуры, принципов работы, процесса создания схемы активного списка в системе. Управление активными списками осуществляется в разделе **Активные списки**.

О схемах активных списков

Схема активного списка — это структурированный объект данных в формате RObject (.ro), который задает конфигурацию и параметры для создания и управления активным списком. Схема определяет структуру данных, правила хранения и управления записями, а также ограничения по количеству и времени жизни записей.

Схема активного списка включает следующие компоненты: идентификатор списка, разметку данных, правила хранения и уникальные ключи для идентификации записей. Поле `capacity` задает максимальное количество записей, которое может быть сохранено в списке, а поле `ttl` (time-to-live) определяет срок хранения каждой записи в секундах. Структура данных описывается с помощью поля `schema`, которое задает типы данных, такие как строки, числа, логические значения и метки времени.

Каждая запись в активном списке идентифицируется по одному или нескольким ключам, указанным в поле `key`, что позволяет гарантировать уникальность записей. Поле `content` содержит фактические данные списка, которые соответствуют структуре, определенной в `schema`.

Работа с активным списком

Доступные операции над активным списком:

- Создание схемы активного списка
- Изменение схемы активного списка
- Удаление схемы активного списка

Создание схемы активного списка

Чтобы добавить схему активного списка:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Схема активного списка**. Система отобразит окно создания схемы активного списка.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля схемы активного списка, чтобы определить логику ее работы.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст схему активного списка и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.



Чтобы схема активного списка стала доступна при создании активного списка, ее необходимо включить.

Вы можете создать черновик схемы активного списка без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение схемы активного списка

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление схемы активного списка

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура схемы активного списка

Схема активного списка в системе содержит набор обязательных и опциональных полей.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор схемы активного списка в системе.	строка	да
`name`	Название схемы списка. Допустимы строчные и прописные буквы латинского алфавита, цифры, подчеркивания и дефис. Пробелы не допускаются.	строка	да
`version`	Версия схемы активного списка, представленная в формате Semantic Versioning.	строка	да
`description`	Описание активного списка, где можно указать дополнительную информацию о его назначении и функциональности.	строка	нет
`type`	Тип списка. Всегда имеет значение `active_list` для элемента этого типа.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`author`	Имя и контактные данные автора схемы активного списка.	строка	нет
`data_source`	Список источников данных, с которыми совместима схема активного списка. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных схеме активного списка для классификации и быстрого поиска.	массив строк	нет
`capacity`	Максимальное количество записей в активном списке. По достижении лимита, записи удаляются и перезаписываются. Если значение установлено в "0", записи хранятся бессрочно и не перезаписываются.	целое число	да
`ttl`	Время жизни записи в активном списке в секундах. По истечении этого времени запись удаляется. Если значение установлено в "0", запись считается постоянной и хранится бессрочно.	целое число	да
`schema`	Объект, описывающий структуру данных активного списка, включая типы данных для ключей. Поддерживает типы: `string`, `vrl_string`, `bool`, `float`, `integer`, `timestamp`.	объект	да
`key`	Поле, которое определяет набор полей, которые используются в качестве уникальных идентификаторов в активном списке.	массив строк	да
`content`	Поле содержит массив с фактическими данными активного списка, которые организованы согласно структуре в `schema`.	массив строк	нет



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Пример схемы активного списка

```
# Уникальный идентификатор схемы активного списка, необходим для ее идентификации.
id: unique_identifier/active_list_name

# Название схемы активного списка.
name: ActiveListName

# Версия схемы активного списка.
version: 1.0.0

# Описание схемы активного списка, его назначение и функциональность.
description: Описание схемы активного списка

# Тип элемента экспертизы. Всегда должно быть 'active_list' для элементов такого типа.
```

```
type: active_list

# Информация об авторе схемы активного списка.
author: Имя Автора <email@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_3
    - EventID_22

# Теги для улучшения поиска схемы в системе.
tags:
  - tag1
  - tag2

# Максимальное количество значений в схеме списка. При переполнении удаляется самое
старое значение.
capacity: 10000

# Ключевые поля схемы активного списка должны быть уникальными.
key:
  - field1
  - field2

# Схема элементов в активном списке. Поддерживает типы: 'string', `vrl_string`,
'bool', 'float', 'integer', 'timestamp'.
schema:
  field1: string
  field2: vrl_string
  field3: bool
  field4: float
  field5: integer
  field6: timestamp

# Время жизни значения внутри активного списка, в секундах. По истечении этого времени
значение будет автоматически удалено из списка.
ttl: 0

# Пример содержимого, соответствующего схеме.
content:
  - field1: "value1"
  - field2: true
  - field3: 1.23
  - field4: 123
  - field5: "2023-01-01T12:00:00Z"
```

Функции управления активными списками



При обращении к активному списку в VRL-песочнице будет возвращено значение `null`.

Для работы с записями активного списка в полях элементов экспертизы, которые поддерживают VRL, применяются специальные функции. Доступны функции добавления, получения и удаления записи из активного списка.

Функция	Описание	Параметры
`active_record_excluded(active_list, key)`	Проверяет, есть ли запись с указанным ключом в активном списке.	• `active_list`: название схемы активного списка, в который добавляется запись. • `key`: ключ записи.
`add_active_record(list, key, value)`	Добавляет запись в активный список. Если запись с таким ключом уже существует, значение обновляется. В противном случае создается новая запись.	• `list`: название схемы активного списка, в который добавляется запись. • `key`: уникальный ключ для записи. • `value`: значение записи.
`get_active_record(list, key)`	Извлекает значение из активного списка по указанному ключу. Возвращает значение, ассоциированное с ключом, или `null`, если запись отсутствует.	• `list`: название схемы активного списка, из которого нужно извлечь запись. • `key`: уникальный ключ записи.
`remove_active_record(list, key)`	Удаляет запись из активного списка. Если запись с таким ключом отсутствует, ничего не происходит.	• `list`: название схемы активного списка, из которого необходимо удалить запись. • `key`: уникальный ключ записи, которую необходимо удалить.

Пример использования в поле `on_correlation` правила корреляции:

```
on_correlation: !vrl |
  if .threat_level == "High" {
    add_active_record!("suspicious_ips", { "ip": .source_ip, "is_suspicious": true })
  }

  if .event_type == "authentication_failure" {
    failure_count = get_active_record!("auth_failure_counts", { "user_id": .user_id
}).failure_count || 0
    add_active_record!("auth_failure_counts", { "user_id": .user_id, {"failure_count":
failure_count + 1 })
  }

  if .event_type == "authentication_success" {
    remove_active_record!("auth_failure_counts", { "user_id": .user_id })
```

```
}
```

В этом примере:

- IP-адреса с высоким уровнем угрозы добавляются в активный список подозрительных IP-адресов.
- Считается количество неудачных попыток аутентификации для пользователей и обновляется соответствующая запись в активном списке.
- При успешной аутентификации запись пользователя удаляется из списка неудачных попыток аутентификации, что может свидетельствовать об окончании атаки или восстановлении доступа пользователя.

Составление схемы активного списка

Управление схемами активных списков в системе осуществляется в разделе **Экспертиза**.

Чтобы составить схему активного списка, необходимо заполнить ее обязательные поля.

Поля метаданных

Заполните поля метаданных схемы активного списка:

- `id`: задайте уникальный идентификатор схемы активного списка.
- `name`: задайте название схемы активного списка. Допустимы строчные и прописные буквы латинского алфавита, цифры, точки, подчеркивания и дефис. Пробелы не допускаются.
- `version`: укажите версию схемы активного списка в формате Semantic Versioning.
- `type`: укажите `active_list` в поле, чтобы определить тип элемента экспертизы.
- `author` (опциональное): укажите имя и контактные данные автора.
- `description` (опциональное): предоставьте краткое описание назначения и функциональности схемы активного списка.
- `data_source` (опциональное): укажите список совместимых со схемой источников данных для классификации и фильтрации.
- `tags` (опциональное): добавьте теги для классификации и поиска списка в системе.

Пример структуры метаданных:

```
id: "example_whitelist_id"
name: "White list"
version: "1.0.0"
author: "Ivan Ivanov <ivanov@example.ru>"
description: "This static active list contains admin IPs."
data_source:
  - platform: Windows
  - source: Sysmon
tags:
  - ddos
```

```
  - threat intelligence
  - blacklist
```

Поле `capacity`

Поле `capacity` (обязательное) определяет максимальное количество записей, которое может содержать активный список. Это ограничение позволяет контролировать объем хранимых данных и поддерживать производительность системы на оптимальном уровне. По достижении лимита, записи удаляются из списка и перезаписываются. Если значение `capacity` установлено в "0", записи хранятся бессрочно и не перезаписываются.

Пример задания максимального количества записей:

```
capacity: 10000
```

В данном случае, активный список будет ограничен 10000 записями. При достижении этого предела, новые записи будут замещать старые.



Максимальное допустимое количество записей в активном списке составляет 2 147 483 646 записей.



Значение по умолчанию для `capacity` не установлено, однако, рекомендуется устанавливать это значение с учетом требований к производительности и объему хранимых данных.

Поле `ttl`

Поле `ttl` (time-to-live) (обязательное) в контексте схемы активного списка определяет максимальное время жизни каждой записи в активном списке, выраженное в секундах. После истечения указанного времени запись автоматически удаляется из списка. Если значение установлено в "0", запись считается постоянной и хранится бессрочно.

Пример:

```
ttl: 86400  # Записи в активном списке будут храниться в течение 24 часов.
```



Максимальное допустимое время жизни записей в активном списке составляет 157 680 000 секунд (5 лет).



Установка оптимального значения `ttl` зависит от специфики данных и задач, которые должен выполнять активный список. Для списков с динамически изменяющейся информацией рекомендуется устанавливать `ttl` с учетом ожидаемого времени актуальности данных.

Поле `schema`

Поле `schema` (обязательное) определяет структуру данных активного списка. Это описание включает типы данных для каждого из полей активного списка, которые используются для хранения информации. Структура `schema` позволяет определить формат и тип данных каждого элемента списка, обеспечивая тем самым обработку и интерпретацию сохраненной информации.

Поля в активном списке поддерживают следующие типы данных:

- `string` — текстовая строка.
- `vrl_string` — строка VRL-программы.
- `bool` — логическое значение.
- `float` — число с плавающей точкой.
- `integer` — целое число.
- `timestamp` — метка времени.

Структура поля `schema`:

```
schema:
  <field_1_name>: <field_1_data_type>
  <field_2_name>: <field_2_data_type>
  <field_3_name>: <field_3_data_type>
```

Здесь:

- `<field_N_name>` — название поля в активном списке;
- `<field_N_data_type>` — тип данных поля в активном списке.

Пример поля `schema`

```
schema:
  ip: string        # IP-адрес, представленный в виде строки.
  active: bool      # Признак активности IP-адреса: true или false.
  score: float      # Оценка репутации IP-адреса, выраженная числом с плавающей
точкой.
  attempts: integer # Количество попыток подключения с данного IP-адреса.
  last_seen: timestamp      # Метка времени последнего обнаружения активности с
данного IP-адреса.
  first_detected: timestamp # Метка времени первого обнаружения активности.
```

Поле `keys`

Поле `keys` (обязательное) определяет набор полей, которые используются в качестве уникальных идентификаторов для каждой записи в активном списке. Эти ключи позволяют идентифицировать записи в списке, обеспечивая возможность поиска, обновления или удаления данных. Уникальность ключа гарантирует, что каждая запись в активном списке будет отличаться от других, предотвращая дублирование информации.

Пример:

```
key:
  - ip
```

В данном примере поле `ip` используется в качестве уникального ключа для записей в активном списке. Это означает, что для каждого уникального IP-адреса в списке может быть только одна запись. При попытке добавить новую запись с уже существующим в списке IP-адресом, старая запись будет обновлена или заменена, в зависимости от реализации механизма управления активным списком.



Выбор ключей для активного списка должен основываться на требованиях к уникальности и возможностях идентификации записей. Важно убедиться, что выбранные поля могут обеспечить уникальность записей в контексте предметной области активного списка.

Поле `content`

Поле `content` содержит фактические данные активного списка, представляя собой массив записей, соответствующих определенной схеме данных `schema`. Эта схема определяет структуру и типы данных для каждого элемента в записях, гарантируя таким образом их корректную обработку и хранение.

Пример схемы `schema`, по которой формируется `content`:

```
schema:
  id: integer
  ip: string
  last_detected: timestamp
  attack_type: string
```

Согласно схеме, поле `content` может содержать следующие данные:

```
content:
  - id: 1
    ip: "192.168.0.1"
    last_detected: "2023-09-01T12:00:00Z"
    attack_type: "DDoS"
  - id: 2
    ip: "192.168.0.2"
    last_detected: "2023-09-02T13:00:00Z"
    attack_type: "Phishing"
```

Каждая запись в `content` соответствует структуре и типу данных, определенных в `schema`. В данном примере каждая запись содержит уникальный идентификатор (`id`), IP-адрес (`ip`), дату последнего обнаружения (`last_detected`) и тип атаки (`attack_type`).

Пример схемы активного списка

Пример схемы активного списка "DDoSAttackIPs", схема RObject

```
# Уникальный идентификатор списка.
id: threat_intel/ddos_attack_ips

# Название активного списка, которое обозначает его цель или функцию.
name: DDoSAttackIPs

# Версия активного списка, представленная в формате SemVer (Semantic Versioning).
version: 1.0.0

# Описание активного списка, где можно указать дополнительную информацию о его
назначении и функциональности.
description: Список IP-адресов, обнаруженных как участники в DDoS-атаках, для
мониторинга и предотвращения потенциальных угроз.

# Тип элемента экспертизы.
type: active_list

# Имя и контактные данные автора активного списка.
author: Ivan Ivanov <ivanov@example.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_3
    - EventID_22

# Теги, связанные с активным списком.
tags:
  - ddos
  - threat intelligence
  - blacklist

# Максимальное количество записей в активном списке. Опционально, по умолчанию 100000.
capacity: 10000

# Время жизни записи активного списка в секундах. Опционально, если не указан, то
хранится бессрочно.
ttl: 0

# Схема описывает поля активного списка.
schema:
  ip: string                # IP-адрес.
  lastDetected: timestamp   # Метка времени последнего обнаружения.
  attackType: string        # Тип атаки.
  reputationScore: integer  # Оценка репутации IP-адреса.
```

```
  attackVolume: string      # Объем атаки.
  involvedAttacks: integer  # Количество атак.
  notes: string             # Примечания.

# Перечень полей, которые являются ключом.
key:
  - ip

# Содержимое активного списка.
content:
  - ip: "192.0.2.1"
    lastDetected: "2023-01-01T12:00:00Z"
    attackType: "SYN Flood"
    reputationScore: 10
    attackVolume: "1.5 Gbps"
    involvedAttacks: 5
    notes: "Repeated attacks on multiple targets"
  - ip: "192.0.2.2"
    lastDetected: "2023-01-02T15:30:00Z"
    attackType: "UDP Flood"
    reputationScore: 20
    attackVolume: "500 Mbps"
    involvedAttacks: 2
    notes: "High packet rate observed"
  - ip: "192.0.2.3"
    lastDetected: "2023-01-03T18:45:00Z"
    attackType: "Amplification"
    reputationScore: 15
    attackVolume: "2 Gbps"
    involvedAttacks: 3
    notes: "DNS amplification attack"
  - ip: "192.0.2.4"
    lastDetected: "2023-01-04T20:00:00Z"
    attackType: "HTTP Flood"
    reputationScore: 5
    attackVolume: "750 Mbps"
    involvedAttacks: 4
    notes: "Targeted at web services"
```

Интерпретация

Схема активного списка "DDoSAttackIPs" служит для учета IP-адресов, обнаруженных как участники в DDoS-атаках.

Метаданные:

- `id`: `threat_intel/ddos_attack_ips` — уникальный идентификатор схемы активного списка.
- `name`: `DDoSAttackIPs` — название схемы активного списка.
- `version`: `1.0.0` — версия схемы, использующаяся для управления изменениями.

- `description`: Схема описывает структуру данных для мониторинга и предотвращения потенциальных угроз от IP-адресов, участвующих в DDoS-атаках.
- `type`: `active_list` — тип элемента экспертизы, указывающий на его предназначение для хранения и обработки информации о потенциальных угрозах.
- `author`: `Ivan Ivanov <ivanov@example.ru>` — информация об авторе схемы, включая контактные данные.
- `tags`: массив тегов (`ddos`, `threat intelligence`, `blacklist`), классифицирующих схему и упрощающих ее поиск и использование в различных контекстах.

Параметры схемы:

- `capacity`: `10000` — максимальное количество записей, которое может содержать активный список, созданный на основе данной схемы.
- `ttl`: `0` — время жизни каждой записи в секундах; значение `0` означает, что записи хранятся бессрочно.
- `key`: массив, указывающий на поля, которые являются ключами для уникальной идентификации записей в списке; в данном случае это поле `ip`.

Схема данных:

Описывает поля активного списка, включая типы данных и их назначение:

- `ip`: строка, содержащая IP-адрес.
- `lastDetected`: дата, когда IP-адрес был последний раз обнаружен как участник DDoS-атаки.
- `attackType`: строка, описывающая тип DDoS-атаки.
- `reputationScore`: целое число, оценка репутации IP-адреса.
- `attackVolume`: строка, описывающая объем атаки.
- `involvedAttacks`: целое число, количество атак, в которых участвовал данный IP.
- `notes`: строка, содержащая примечания или дополнительную информацию об IP-адресе.

Имена полей могут включать буквы латинского алфавита и цифры (`a–z`, `A–Z`, `0–9`).

Содержимое активного списка:

Примеры записей, которые могут быть добавлены в активный список на основе данной схемы, включают информацию о дате обнаружения, типе атаки, репутационной оценке IP-адреса, объеме атаки, количестве атак, в которых участвовал адрес, и заметках о конкретных атаках.

Схема активного списка "DDoSAttackIPs" используется в активных списках для динамического обновления и хранения информации об IP-адресах, связанных с DDoS-атаками.

Схемы дашбордов

В данном разделе представлен обзор структуры, принципов работы и процесса создания схемы дашборда в системе. Схемы дашбордов используются для настройки конфигурации и параметров отображения данных в дашборде. Управление схемами дашбордов в системе осуществляется в

разделе **Экспертиза**.

О схемах дашбордов

Схема дашборда — это структурированный объект данных в формате RObject (.ro), который задает конфигурацию и параметры для создания дашборда. Схема определяет структуру дашборда, правила отображения и обновления данных, а также настройки расположения виджетов.

Схема дашборда включает следующие компоненты: разметку виджетов, параметры каждого виджета и набор переменных для получения данных. Размещение задает позицию, размеры и ограничения для каждого виджета на сетке дашборда. Виджеты отображают данные в различных форматах, таких как графики, таблицы или сводные значения.

Каждый виджет включает описание источника данных, параметры запроса и настройки визуализации. В качестве источников данных могут использоваться метрики, хранилища событий, активные списки или оповещения. Например, графики отображают динамику метрик, таблицы — детализированные данные по событиям, а сводные виджеты — итоговые значения.

Схема позволяет настраивать параметры обновления данных, включая частоту и период отображения. Это обеспечивает возможность отображения как текущих, так и ранее собранных данных. Переменные внутри схемы используются для передачи идентификаторов источников данных и фильтров в запросы.

Работа со схемой дашборда

Доступные операции над схемой дашборда:

- Создание схемы дашборда
- Изменение схемы дашборда
- Удаление схемы дашборда

Создание схемы дашборда

Чтобы создать схему дашборда:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Схема дашборда**. Система отобразит окно создания схемы.

> При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля схемы дашборда, чтобы определить функциональность и внешний вид дашбордов.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст схему дашборда и отобразит

соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.



Чтобы схема дашборда стала доступной для создания дашбордов, ее необходимо включить.

Вы можете создать черновик схемы дашборда без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.



Вы можете также создать схему на основе готового дашборда.

Изменение схемы дашборда

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление схемы дашборда

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура схемы дашборда

Схема дашборда содержит обязательные и опциональные поля, определяющие его поведение и отображение.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор схемы дашборда. Значение — пять групп из шестнадцатеричных цифр в формате 8-4-4-4-12, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.	UUID	да
`name`	Название схемы дашборда. Допустимы латинские буквы, цифры, точки, подчеркивания и дефисы.	строка	да
`description`	Краткое описание схемы дашборда.	строка	нет
`type`	Тип схемы. Для дашбордов всегда указывается `dashboard_schema`.	строка	да
`author`	Имя и контактные данные автора схемы дашборда.	строка	нет
`data_source`	Список источников данных, с которыми совместима схема дашборда. Используется для классификации и быстрого поиска элементов экспертизы в системе по совместимым источникам.	массив строк массив объектов	нет

Поле	Описание	Тип данных	Обязательное поле
`tags`	Список тегов, присвоенных схеме дашборда для классификации и быстрого поиска.	массив строк	нет
`version`	Версия схемы дашборда в формате Semantic Versioning.	строка	да
`scope`	Доступность дашборда. Указывает, является ли дашборд персональным (`user`) или общим (`tenant`).	строка	да
`variables`	Набор переменных, используемых в дашборде для получения данных. Поле `key` — уникальный идентификатор переменной. Поле `type` определяет тип источника данных, с которым связана переменная. Допустимые значения для `type`: `event_storage` — хранилище событий, `active_list` — активный список, `metric` — метрики, `alert` — оповещения.	массив объектов	да
`layout`	Разметка виджетов на дашборде, включающая их положение и размеры.	массив объектов	да
`widgets`	Описание виджетов, их настройки и параметры автоматического обновления.	массив объектов	да



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `variables`

Поле	Описание	Тип данных	Обязательное поле
`key`	Уникальный идентификатор переменной, используемой в дашборде.	строка	да
`type`	Тип источника данных, с которым связана переменная. Допустимые значения: • `event_storage` — хранилище событий; • `active_list` — активный список; • `metric` — метрики; • `alert` — оповещения.	строка	да

Структура `layout`

Поле `layout` описывает расположение виджетов на дашборде, включая их позицию на сетке и размеры.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор виджета.	строка	да
`x`, `y`	Позиция виджета на сетке (`x` — по горизонтали, `y` — по вертикали).	число	да
`w`, `h`	Размеры виджета в ячейках сетки (ширина `w`, высота `h`).	число	да
`maxH`, `minH`	Максимальная и минимальная высота виджета.	число	да
`minW`	Минимальная ширина виджета.	число	да

Структура `widgets`

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор виджета. Используется для его ссылки в других элементах схемы дашборда.	строка	да
`name`	Название виджета, которое будет отображаться на дашборде.	строка	да
`description`	Краткое описание виджета, поясняющее его назначение.	строка	нет
`autoUpdate`	Настройки автоматического обновления данных виджета. Включает единицу времени (`unit`) и значение интервала обновления (`value`).	объект	да
`last`	Указывает временной период данных, которые будут отображаться на виджете. Например, за последний час или месяц.	объект	нет
`variables`	Переменные, которые используются для запроса данных виджета. Включает источник данных (`source`) и дополнительные параметры запроса.	объект	да

Структура `autoUpdate`

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, через которую будет происходить автоматическое обновление данных виджета. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`value`	Интервал обновления в единицах, указанных в поле `unit`. Например, при значении `unit: MINUTE` и `value: 5`, обновление будет происходить каждые 5 минут.	число	да

Структура `last`

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, за которую будут отображаться данные. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`.	строка	да
`value`	Числовое значение интервала для выбранной единицы времени. Например, при значении `unit: DAY` и `value: 7` будут отображены данные за последние 7 дней.	число	да

Структура `variables`

Поле	Описание	Тип данных	Обязательное поле
`source`	Определяет источник данных для виджета. Допустимые значения: `metric`, `search`, `alert`.	строка	да
`metricId`	Идентификатор метрики, данные которой должны быть отображены. Применяется, если `source` установлен на `metric`.	строка	да (если `source` = `metric`)
`chartType`	Тип графика, который используется для визуализации данных. Допустимые значения: `line_chart`, `bar_chart`, `pie_chart`, `table`, `total`.	строка	да (если используется график)
`rql`	Запрос на языке RQL, который используется для получения данных. Применяется, если `source` установлен на `search`.	строка	да (если `source` = `search`)
`nameMapping`	Массив, который задает сопоставление полей данных с отображаемыми элементами (например, `x`, `y` для графиков или `timestamp`, `cnt` для таблиц).	массив объектов	да (для графиков и таблиц)
`kind`	Тип отображаемых оповещений. Допустимые значения: `active`, `new`, `total`, `last`, `distributionByField`, `distributionByTime`. Применяется, если `source` установлен на `alert`.	объект	да (если `source` = `alert`)

Составление схемы дашборда

Для создания схемы дашборда необходимо определить ряд обязательных и опциональных полей, которые формируют структуру и поведение дашборда.

Поля метаданных

Заполните поля метаданных схемы дашборда:

- `id`: уникальный идентификатор схемы дашборда. Значение — пять групп из шестнадцатеричных цифр в формате 8-4-4-4-12, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.
- `name`: название схемы дашборда. Допустимы латинские буквы, цифры, символы подчеркивания, точки и дефисы.
- `type`: всегда указывается `dashboard_schema` для элементов экспертизы данного типа.
- `description` (опционально): краткое описание назначения дашборда.
- `author` (опционально): имя и контактные данные автора схемы дашборда.
- `version`: версия схемы дашборда согласно стандарту Semantic Versioning.
- `scope`: доступность дашборда. Указывает, является ли он персональным (`user`) или общим (`tenant`).
- `data_source` (опционально): список источников данных, с которыми совместима схема дашборда. Используется для классификации и быстрого поиска элементов экспертизы в системе.
- `tags` (опционально): список тегов, присвоенных схеме дашборда для классификации и быстрого поиска.

Пример метаданных:

```
id: "61f0c404-5cb3-11e7-907b-000000000000"
name: "System Monitoring Dashboard"
type: dashboard_schema
description: "Дашборд для мониторинга системных метрик и событий"
author: Ivan Ivanov <ivanov@example.ru>
version: "0.0.1"
scope: tenant
data_source:
  - platform: Windows
  - source: Sysmon
tags:
  - monitoring
  - system
```

В этом примере:

- Схема дашборда с уникальным идентификатором `61f0c404-5cb3-11e7-907b-000000000000` предназначена для мониторинга системных метрик и событий.
- Название схемы `System Monitoring Dashboard` описывает назначение дашборда.

- Поле `type` указывает на тип элемента экспертизы (`dashboard_schema`).
- Версия схемы дашборда задана как `0.0.1` в соответствии с правилами семантического версионирования.
- Автором схемы является `Ivan Ivanov`, чьи контактные данные указаны в поле `author`.
- В поле `data_source` перечислены источники данных, совместимые с дашбордом — `Windows` и `Sysmon`.
- Теги `monitoring` и `system` присвоены для облегчения поиска и классификации дашборда в системе.

Поле `variables`

Поле `variables` используется для передачи переменных в дашборд. Переменные могут включать следующие типы источников данных: хранилища событий, метрики и оповещения.

- `key`: уникальный идентификатор переменной.
- `type`: тип источника данных. Допустимые значения:
  - `event_storage` — хранилище событий;
  - `active_list` — активный список;
  - `metric` — метрики;
  - `alert` — оповещения.

Пример:

```
variables:
  - key: VARIABLE_EVENT_STORAGE_ID_1
    type: event_storage
```

В этом примере:

- Переменная с ключом `VARIABLE_EVENT_STORAGE_ID_1` используется для обозначения хранилища событий в дашборде. Тип переменной установлен как `event_storage`, что указывает на использование хранилища событий в запросах данных.

Поле `layout`

Поле `layout` определяет размещение виджетов на дашборде. Каждый виджет имеет уникальный идентификатор и параметры, определяющие его положение и размеры:

- `id`: уникальный идентификатор виджета.
- `x`, `y`: координаты виджета на сетке (`x`-- позиция по горизонтали, `y` — позиция по вертикали).
- `w`, `h`: размеры виджета в ячейках (ширина `w` и высота `h`).
- `maxH`, `minH`, `minW`: ограничения по минимальной и максимальной высоте и ширине.

Пример разметки виджетов:

```
layout:
  - id: WIDGET_KEY_1
    x: 0
    y: 1
    w: 6
    h: 3
    maxH: 5
    minH: 3
    minW: 6
  - id: WIDGET_KEY_2
    x: 0
    y: 4
    w: 6
    h: 3
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` размещен в позиции `x: 0`, `y: 1`, что означает, что он будет располагаться в первом столбце и второй строке сетки дашборда.
- Его ширина составляет 6 ячеек (`w: 6`), а высота — 3 ячейки (`h: 3`).
- Для этого виджета также установлены ограничения: максимальная высота (`maxH`) — 5 ячеек, минимальная высота (`minH`) — 3 ячейки и минимальная ширина (`minW`) — 6 ячеек.
- Виджет с идентификатором `WIDGET_KEY_2` размещен в позиции `x: 0`, `y: 4`, что означает, что он будет находиться в первом столбце и на пятой строке сетки дашборда.
- Его размеры такие же, как у первого виджета — ширина 6 ячеек и высота 3 ячейки. Ограничения по высоте и ширине в данном примере для второго виджета не заданы.

Поле `widgets`

Поле `widgets` описывает каждый виджет, его данные и тип отображаемой информации. Основные параметры виджетов:

- `id`: уникальный идентификатор виджета.
- `name`: название виджета.
- `description`: краткое описание функциональности виджета.
- `autoUpdate`: настройки автоматического обновления данных (единица времени и интервал).
- `last`: период отображаемых данных (например, последний месяц или день).
- `variables`: параметры запроса данных для виджета.

Пример описания виджета:

```
widgets:
  - id: WIDGET_KEY_1
    name: "Host DB RAM"
    description: "Отображает текущее использование ОЗУ базы данных"
```

```
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: DAY
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
      queryArguments: []
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` назван "Host DB RAM" и предназначен для отображения текущего использования оперативной памяти базы данных.
- Поле `description` описывает функциональность виджета как "Отображает текущее использование ОЗУ базы данных".
- Виджет настроен на автоматическое обновление данных каждые 1 минуту, что задается параметрами в поле `autoUpdate` (`unit: MINUTE` и `value: 1`).
- Поле `last` указывает, что виджет будет отображать данные за последний день (`unit: DAY` и `value: 1`).
- В поле `variables` указано, что данные берутся из источника типа `metric`, с идентификатором метрики "c9c9e971-7d45-48eb-9ed5-000000000000". Дополнительные параметры запроса определены как `queryArguments: []`.

Структура виджетов

Виджеты на дашборде могут быть нескольких типов в зависимости от отображаемых данных и формата представления. Каждый тип виджета имеет уникальные параметры.

Виджеты с метриками

Метрики отображают количественные данные, такие как использование ресурсов системы, количество событий и другие параметры, собираемые из источников.

Поля виджета с метриками:

- `source`: определяет источник данных для виджета (допустимые значения: `metric`, `search`, `alert`).
- `metricId`: идентификатор метрики, данные которой отображаются.
- `step`: интервал времени, за который агрегируются данные.

Пример виджета с метрикой:

```
widgets:
  - id: WIDGET_KEY_1
    name: "CPU Usage"
    description: "Отображает использование CPU системы"
```

```
    autoUpdate:
      unit: MINUTE
      value: 5
    last:
      unit: HOUR
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
      step: ONE_MIN
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` называется "CPU Usage" и предназначен для отображения текущей загрузки процессора системы.
- Поле `description` содержит описание "Отображает использование CPU системы".
- Виджет обновляется каждые 5 минут, что задано в поле `autoUpdate` (`unit: MINUTE`, `value: 5`).
- Поле `last` указывает, что данные за последний час будут агрегироваться и отображаться на виджете (`unit: HOUR`, `value: 1`).
- Источник данных для виджета определен в поле `variables`. Здесь указывается, что данные берутся из метрики (`source: metric`), с идентификатором метрики `metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"`, а интервал агрегации данных установлен на одну минуту (`step: ONE_MIN`).

Виджеты с графиками

Графики визуализируют изменение данных во времени и поддерживают несколько типов отображения: линейные графики, столбчатые диаграммы, круговые диаграммы и др.

Поля виджета с графиком:

- `chartType`: тип графика (`line_chart`, `bar_chart`, `pie_chart`, `table`, `total`).
- `rql`: запрос на языке RQL для получения данных.
- `nameMapping`: сопоставление полей запроса с осями графика (например, `x` — время, `y` — количество).
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример виджета с линейным графиком:

```
widgets:
  - id: WIDGET_KEY_2
    name: "Event Storage Counter"
    description: "Отображает количество событий в хранилище"
    autoUpdate:
      unit: MINUTE
      value: 30
```

```
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: line_chart
      rql: "SELECT count(id) as cnt, toStartOfHour(timestamp) as ts GROUP BY ts"
      nameMapping:
        - key: x
          value: ts
        - key: y
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_2` называется "Event Storage Counter" и предназначен для отображения количества событий в хранилище.
- Поле `description` содержит описание "Отображает количество событий в хранилище".
- Виджет автоматически обновляется каждые 30 минут, что задано в поле `autoUpdate` (`unit: MINUTE`, `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний месяц (`unit: MONTH`, `value: 1`).
- В поле `variables` виджет использует источник данных типа `search` и тип графика `line_chart`, что означает, что данные будут отображаться в виде линейного графика.
- Запрос на языке RQL (`rql`) возвращает количество событий (`cnt`), сгруппированных по часам (`toStartOfHour(timestamp)`).
- Поле `nameMapping` используется для сопоставления полей запроса с осями графика: поле `ts` отображается на оси `x`, а количество событий `cnt` — на оси `y`.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Табличные виджеты

Табличные виджеты представляют данные в виде таблицы, показывая несколько столбцов с данными.

Поля табличного виджета:

- `chartType`: всегда имеет значение `TABLE`.
- `rql`: запрос на языке RQL для получения данных, которые будут отображены в таблице.
- `nameMapping`: сопоставление полей со столбцами таблицы (например, `timestamp`, `count`).
- `limit`: лимит строк для виджета.

- `searchEntity`: идентификатор переменной и тип источника данных.

Пример табличного виджета:

```
widgets:
  - id: WIDGET_KEY_3
    name: "Event Table"
    description: "Отображает события с временными метками"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: DAY
      value: 1
    variables:
      source: search
      chartType: table
      rql: "SELECT timestamp, count(id) as cnt FROM event_table GROUP BY timestamp"
      nameMapping:
        - key: timestamp
          value: timestamp
        - key: count
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_3` называется "Event Table" и отображает события с указанием временных меток.
- Поле `description` содержит описание "Отображает события с временными метками".
- Виджет автоматически обновляется каждые 30 минут (`autoUpdate` с параметрами `unit: MINUTE` и `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний день (`unit: DAY`, `value: 1`).
- В поле `variables` указано, что виджет использует тип отображения данных `table` (`chartType: TABLE`), а данные запрашиваются из источника типа `search`.
- Запрос на языке RQL (`rql`) получает временные метки (`timestamp`) и количество событий (`cnt`), сгруппированных по времени.
- Поле `nameMapping` сопоставляет поля запроса со столбцами таблицы: поле `timestamp` отображается в столбце с временными метками, а поле `cnt` — в столбце с количеством событий.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Виджеты с оповещениями

Виджеты с оповещениями отображают оповещения и события, классифицированные по уровням угрозы, времени создания и другим параметрам.

Поля виджета с оповещениями:

- `source`: всегда указывается `alert`.
- `kind`: тип отображаемых оповещений (`active`, `new`, `last`, `distributionByField`, `distributionByTime`, `total`).
- `filter`: опциональные фильтры для отображаемых оповещений (например, по уровню угрозы).

Пример виджета с активными оповещениями:

```
widgets:
  - id: WIDGET_KEY_4
    name: "Active Alerts"
    description: "Отображает активные оповещения в системе"
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: HOUR
      value: 1
    variables:
      source: alert
      kind:
        kind: active
        active: {}
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_4` называется "Active Alerts" и предназначен для отображения активных оповещений в системе.
- Поле `description` содержит описание "Отображает активные оповещения в системе").
- Виджет обновляется каждую минуту (`autoUpdate` с параметрами `unit: MINUTE`, `value: 1`).
- Поле `last` указывает, что виджет будет отображать оповещения за последний час (`unit: HOUR`, `value: 1`).
- Источник данных для виджета — это оповещения (`source: alert`).
- Поле `kind` указывает, что отображаются только активные оповещения (`kind: active`), без дополнительной фильтрации (параметры в `active: {}` не указаны).

Виджеты со сводными данными

Сводные виджеты отображают итоговые значения по выбранным параметрам, такие как общее количество событий или использования ресурсов.

Поля сводного виджета:

- `chartType`: всегда имеет значение `total`.
- `rql`: запрос на языке RQL для получения сводных данных.
- `nameMapping`: сопоставление результата с отображаемым полем.
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример сводного виджета:

```
widgets:
  - id: WIDGET_KEY_5
    name: "Total Event Count"
    description: "Отображает общее количество событий"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: total
      rql: "SELECT count(id) as cnt FROM event_storage"
      nameMapping:
        - key: value
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_5` назван "Total Event Count" и отображает общее количество событий.
- Поле `description` содержит описание "Отображает общее количество событий".
- Виджет автоматически обновляется каждые 30 минут (`autoUpdate` с параметрами `unit: MINUTE` и `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний месяц (`unit: MONTH`, `value: 1`).
- Источник данных — хранилище событий (`source: search`), тип виджета — сводный (`chartType: total`).
- Запрос на языке RQL (`rql`) возвращает общее количество событий в хранилище (`SELECT count(id) as cnt FROM event_storage`).
- Поле `nameMapping` сопоставляет поле `cnt` с отображаемым полем виджета, обозначенным как `value`.

- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Пример схемы дашборда

Пример схемы дашборда "System Monitoring Dashboard", схема RObject

```
# Универсальный уникальный идентификатор схемы дашборда.
id: 61f0c404-5cb3-11e7-907b-000000000000

# Название дашборда.
name: System Monitoring Dashboard

# Тип элемента экспертизы.
type: dashboard_schema

# Версия схемы.
version: 1.0.0

# Принадлежность дашборда (личный или общий).
scope: user

# Краткое описание назначения дашборда.
description: Дашборд для мониторинга системных метрик и событий в реальном времени

# Имя и контактные данные автора схемы.
author: Ivan Ivanov <ivanov@example.ru>

# Источники данных, с которыми совместима схема дашборда.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги для классификации и быстрого поиска схемы.
tags:
  - monitoring
  - system

# Переменные для источников данных виджетов.
variables:
  - key: VARIABLE_EVENT_STORAGE_ID_1
    type: EVENT_STORAGE

# Размещение виджетов на дашборде.
layout:
  - id: WIDGET_KEY_1
    x: 0
```

```
    y: 1
    w: 6
    h: 3
    maxH: 5
    minH: 3
    minW: 6
  - id: WIDGET_KEY_2
    x: 0
    y: 4
    w: 6
    h: 3
  - id: WIDGET_KEY_3
    x: 0
    y: 7
    w: 6
    h: 3

# Параметры виджетов на дашборде.
widgets:
  - id: WIDGET_KEY_1
    name: Host DB RAM
    description: Виджет для отображения текущего использования оперативной памяти
базой данных
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: DAY
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
      queryArguments: []

  - id: WIDGET_KEY_2
    name: Event Storage Counter (LineChart)
    description: Линейный график для отображения количества событий в хранилище за
месяц
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: line_chart
      rql: "SELECT count(id) as cnt, toStartOfHour(timestamp) as ts GROUP BY ts"
      nameMapping:
        - key: x
          value: ts
```

```
        - key: y
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage


- id: WIDGET_KEY_3
    name: Event Storage Table
    description: Таблица для отображения событий из хранилища с указанием времени и
количества событий
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: DAY
      value: 1
    variables:
      source: search
      chartType: table
      rql: "SELECT timestamp, count(id) as cnt FROM event_table GROUP BY timestamp"
      nameMapping:
        - key: timestamp
          value: timestamp
        - key: count
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

Интерпретация

Метаданные схемы дашборда:

- `id`: универсальный уникальный идентификатор схемы дашборда (UUID) — "61f0c404-5cb3-11e7-907b-000000000000".
- `name`: название схемы дашборда — "System Monitoring Dashboard".
- `type`: тип элемента — всегда указывается как `dashboard_schema`.
- `version`: версия схемы — "1.0.0".
- `scope`: принадлежность дашборда — "user", что указывает на то, что дашборд доступен только конкретному пользователю.
- `description`: краткое описание назначения дашборда — "Дашборд для мониторинга системных метрик и событий в реальном времени".
- `author`: автор схемы дашборда — "Ivan Ivanov <ivanov@example.ru>".
- `data_source`: источники данных, с которыми совместима схема — "Windows", "Sysmon".

- `tags`: теги, присвоенные схеме для классификации — "monitoring", "system".

Переменные:

- `variables`: переменные используются для передачи данных в виджеты. В данном примере переменная `VARIABLE_EVENT_STORAGE_ID_1` определена как идентификатор хранилища событий.

Разметка виджетов:

- `layout`: разметка задает размещение виджетов на дашборде. Например, виджет с идентификатором `WIDGET_KEY_1` размещен в позиции `x: 0`, `y: 1` с шириной 6 и высотой 3.

Виджеты:

1. **Host DB RAM**: отображает использование оперативной памяти базы данных. Виджет обновляется каждые 1 минуту, и отображаются данные за последний день.
   - `source`: метрика (используется источник данных типа `metric`).
   - `metricId`: идентификатор метрики для получения данных о RAM — "c9c9e971-7d45-48eb-9ed5-000000000000".
2. **Event Storage Counter (LineChart)**: линейный график для отображения количества событий в хранилище за последний месяц. Обновление происходит каждые 30 минут.
   - `source`: данные из хранилища событий (используется источник `search`).
   - `chartType`: тип графика — линейный (`line_chart`).
   - `rql`: запрос на языке RQL для получения количества событий и их группировки по часам.
   - `nameMapping`: сопоставление полей запроса с осями графика (ось `x` — время `ts`, ось `y` — количество событий `cnt`).
   - `limit`: лимит строк (10 000).
3. **Event Storage Table**: таблица для отображения событий с указанием времени и количества событий. Обновление происходит каждые 30 минут.
   - `source`: данные из хранилища событий (используется источник `search`).
   - `chartType`: тип отображения — таблица (`table`).
   - `rql`: запрос на языке RQL для группировки событий по времени и подсчета их количества.
   - `nameMapping`: сопоставление полей для столбцов таблицы (столбец `timestamp` — время события, столбец `cnt` — количество событий).
   - `limit`: лимит строк (10 000).

Схемы шаблонов отчетов

В данном разделе представлен обзор структуры, принципов работы и процесса создания схемы шаблона отчета в системе. Схемы шаблонов отчетов используются для настройки конфигурации и параметров формирования шаблонов отчетов. Управление схемами шаблонов отчетов в системе осуществляется в разделе **Экспертиза**.

О схемах шаблонов отчетов

Схема шаблона отчета — это структурированный объект данных в формате RObject (.ro), который задает конфигурацию и параметры для создания шаблона отчета. Схема определяет структуру отчета, правила его формирования и отображения данных.

Схема шаблона отчета включает следующие компоненты: параметры хранения данных, расписание формирования отчета, разметку и параметры виджетов, а также переменные для получения данных. Параметры хранения задают период времени, в течение которого данные остаются доступными для включения в отчеты. Разметка определяет расположение виджетов, которые визуализируют данные в различных форматах, таких как графики, таблицы и сводные значения.

Каждый виджет содержит источник данных, параметры обновления и запросы для получения информации. Данные могут поступать из различных источников, таких как метрики, хранилища событий, активные списки или оповещения. Переменные используются для передачи идентификаторов источников данных и других параметров в запросы, что позволяет управлять выборкой данных для отчета.

Работа со схемой шаблона отчета

Доступные операции над схемой шаблона отчета:

- Создание схемы шаблона отчета
- Изменение схемы шаблона отчета
- Удаление схемы шаблона отчета

Создание схемы шаблона отчета

Чтобы добавить схему шаблона отчета:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Схема шаблона отчета**. Система отобразит окно создания схемы.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля схемы, чтобы определить функциональность и внешний вид отчетов.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст схему шаблона отчета и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.



Чтобы схема стала доступной для создания шаблонов отчета, ее необходимо включить.

Вы можете создать черновик схемы шаблона без публикации с помощью кнопки **Сохранить**

черновик. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.



Вы можете также создать схему на основе готового шаблона отчета.

Изменение схемы шаблона отчета

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление схемы шаблона отчета

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура схемы шаблона отчета

Схема шаблона отчета содержит обязательные и опциональные поля, которые определяют его параметры и функциональность.

Поле	Описание	Тип данных	Обязательное поле
`id`	Универсальный уникальный идентификатор схемы шаблона отчета. Значение — пять групп из шестнадцатеричных цифр в формате 8-4-4-4-12, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.	UUID	да
`name`	Название шаблона отчета. Допустимы латинские буквы, цифры, точки, подчеркивания и дефисы.	строка	да
`description`	Краткое описание шаблона отчета.	строка	нет
`type`	Тип схемы. Для шаблонов отчетов всегда указывается `report_template_schema`.	строка	да
`author`	Имя и контактные данные автора схемы шаблона отчета.	строка	нет
`data_source`	Список источников данных, с которыми совместима схема шаблона отчета. Используется для классификации и быстрого поиска элементов экспертизы в системе по совместимым источникам.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных схеме шаблона отчета для классификации и быстрого поиска.	массив строк	нет
`version`	Версия схемы шаблона отчета в формате Semantic Versioning.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`storagePeriod`	Параметры времени хранения данных, включаемых в отчет.	объект	да
`schedule`	Параметры расписания формирования отчетов согласно формату cron.	объект	да
`variables`	Набор переменных, используемых в отчете для получения данных. Поле `key` — уникальный идентификатор переменной. Поле `type` определяет тип источника данных, с которым связана переменная. Допустимые значения для `type`: `active_list` — активный список, `event_storage` — хранилище событий, `metric` — метрики, `alert`-- оповещения.	массив объектов	да
`layout`	Размещение виджетов в отчете, определяющая их расположение и размеры.	массив объектов	да
`widgets`	Описание виджетов, их настройки и параметры автоматического обновления.	массив объектов	да
`autoUpdate`	Настройки автоматического обновления данных виджета. Включает единицу времени (`unit`), допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`, и значение интервала обновления (`value`) в виде положительного целого числа.	объект	да



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `storagePeriod`

Поле `storagePeriod` определяет временной интервал, в течение которого данные остаются доступными для формирования отчетов.

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, например, `MINUTE`, `HOUR`, `DAY`, `MONTH`.	строка	да
`value`	Числовое значение для указанной единицы времени. Например, при значении `unit: MINUTE` и `value: 60`, данные будут храниться 60 минут.	число	да

Структура `schedule`

Поле `schedule` задает параметры расписания согласно формату cron для автоматического формирования отчетов.

Поле	Описание	Тип данных	Обязательное поле
`hours`	Часы, в которые будет формироваться отчет. Указывается в формате `0-23`.	строка	да
`minutes`	Минуты, в которые будет формироваться отчет. Указывается в формате `0-59`.	строка	да
`seconds`	Секунды для формирования отчета. Указывается в формате `0-59`.	строка	да
`dayOfMonth`	День месяца для формирования отчета. Указывается в формате `1-31` или `*` для ежедневного формирования.	строка	да
`dayOfWeek`	День недели для формирования отчета. Указывается в формате `0-6` (где 0 — это воскресенье) или `*` для ежедневного формирования.	строка	да
`month`	Месяц для формирования отчета. Указывается в формате `1-12` или `*` для формирования отчета каждый месяц.	строка	да

Структура `variables`

Поле	Описание	Тип данных	Обязательное поле
`key`	Уникальный идентификатор переменной, используемой в шаблоне отчета.	строка	да
`type`	Тип источника данных, с которым связана переменная. Допустимые значения: • `event_storage` — хранилище событий; • `active_list` — активный список; • `metric` — метрики; • `alert` — оповещения.	строка	да

Структура `layout`

Поле `layout` описывает размещение виджетов в отчете, включая их позицию на сетке и размеры.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор виджета.	строка	да
`x`, `y`	Позиция виджета на сетке (`x` — по горизонтали, `y` — по вертикали).	число	да

Поле	Описание	Тип данных	Обязательное поле
`w`, `h`	Размеры виджета в ячейках сетки (ширина `w`, высота `h`).	число	да
`maxH`, `minH`	Максимальная и минимальная высота виджета.	число	да
`minW`	Минимальная ширина виджета.	число	да

Структура `widgets`

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор виджета. Используется для его ссылки в других элементах схемы шаблона отчета.	строка	да
`name`	Название виджета, которое будет отображаться в отчете.	строка	да
`description`	Краткое описание виджета, поясняющее его назначение.	строка	нет
`autoUpdate`	Настройки автоматического обновления данных виджета. Включает единицу времени (`unit`) и значение интервала обновления (`value`).	объект	да
`last`	Указывает временной период данных, которые будут отображаться на виджете. Например, за последний час или месяц.	объект	нет
`variables`	Переменные, которые используются для запроса данных виджета. Включает источник данных (`source`) и дополнительные параметры запроса.	объект	да

Структура `autoUpdate`

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, через которую будет происходить автоматическое обновление данных виджета. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`.	строка	да
`value`	Интервал обновления в единицах, указанных в поле `unit`. Например, при значении `unit: MINUTE` и `value: 5`, обновление будет происходить каждые 5 минут.	число	да

Структура `last`

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, за которую будут отображаться данные. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`.	строка	да
`value`	Числовое значение интервала для выбранной единицы времени. Например, при значении `unit: DAY` и `value: 7` будут отображены данные за последние 7 дней.	число	да

Структура `variables`

Поле	Описание	Тип данных	Обязательное поле
`source`	Определяет источник данных для виджета. Допустимые значения: `metric`, `search`, `alert`.	строка	да
`metricId`	Идентификатор метрики, данные которой должны быть отображены. Применяется, если `source` установлен на `metric`.	строка	да (если `source` = `metric`)
`chartType`	Тип графика, который используется для визуализации данных. Допустимые значения: `LINE_CHART`, `BAR_CHART`, `PIE_CHART`, `TABLE`, `TOTAL`.	строка	да (если используется график)
`rql`	Запрос на языке RQL, который используется для получения данных. Применяется, если `source` установлен на `search`.	строка	да (если `source` = `search`)
`nameMapping`	Массив, который задает сопоставление полей данных с отображаемыми элементами (например, `x`, `y` для графиков или `timestamp`, `cnt` для таблиц).	массив объектов	да (для графиков и таблиц)
`kind`	Тип отображаемых оповещений. Допустимые значения: `active`, `new`, `total`, `last`, `distributionByField`, `distributionByTime`. Применяется, если `source` установлен на `alert`.	объект	да (если `source` = `alert`)

Составление схемы шаблона отчета

Для создания схемы шаблона отчета необходимо определить ряд обязательных и опциональных полей, которые формируют структуру и поведение шаблона отчета.

Поля метаданных

Заполните поля метаданных схемы шаблона отчета:

- `id`: уникальный идентификатор схемы шаблона отчета. Значение — пять групп из

шестнадцатеричных цифр в формате 8-4-4-4-12, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`. Должен быть уникальным в рамках коллектора.

- `name`: название схемы шаблона отчета. Допустимы латинские буквы, цифры, символы подчеркивания, точки и дефисы.
- `description` (опционально): краткое описание назначения шаблона отчета.
- `type`: всегда указывается `report_template_schema` для элементов экспертизы данного типа.
- `version`: версия схемы шаблона отчета согласно стандарту Semantic Versioning.
- `author`: имя и контактные данные автора схемы шаблона отчета.
- `data_source`: список источников данных, с которыми совместима схема шаблона отчета. Это поле используется для классификации и быстрого поиска элементов в системе.
- `tags`: список тегов, присвоенных схеме шаблона отчета, для классификации и упрощения поиска.

Пример метаданных:

```
id: "61f0c404-5cb3-11e7-907b-000000000000"
name: "System Report Template"
description: "Шаблон для создания отчетов по метрикам системы и событиям"
type: report_template_schema
version: "1.0.0"
author: Ivan Ivanov <ivanov@example.ru>
data_source:
  - Windows
  - Linux
tags:
  - system
  - report
```

В этом примере:

- Схема шаблона отчета с уникальным идентификатором `61f0c404-5cb3-11e7-907b-000000000000` предназначена для создания отчетов по метрикам системы и событиям.
- Название схемы `System Report Template` описывает назначение шаблона.
- Поле `type` указывает, что тип элемента экспертизы (`report_template_schema`).
- Версия шаблона отчета задана как `1.0.0` в соответствии с правилами семантического версионирования.
- Автором схемы является `Ivan Ivanov`, чьи контактные данные указаны в поле `author`.
- В поле `data_source` перечислены источники данных, совместимые с отчетом — `Windows` и `Linux`.
- Теги `system` и `report` присвоены для облегчения поиска и классификации шаблона отчета в системе.

Поле `storagePeriod`

Поле `storagePeriod` определяет временной интервал, в течение которого данные остаются

доступными для включения в отчет.

- `unit`: единица времени, например, `MINUTE`, `HOUR`, `DAY`, `MONTH`.
- `value`: числовое значение для выбранной единицы времени.

Пример:

```
storagePeriod:
  unit: MINUTE
  value: 60
```

В этом примере:

- Поле `storagePeriod` задает временной интервал для сохранения данных, который равен 60 минутам. Это означает, что данные, собранные за последние 60 минут, будут включены в отчет.

Поле `schedule`

Поле `schedule` задает параметры расписания согласно формату cron для автоматического формирования отчетов.

- `hours`: часы, в которые будет формироваться отчет (формат `0-23`).
- `minutes`: минуты для формирования отчета (формат `0-59`).
- `seconds`: секунды для формирования отчета (формат `0-59`).
- `dayOfMonth`: день месяца для формирования отчета (`1-31` или `*` для ежедневного отчета).
- `dayOfWeek`: день недели для формирования отчета (`0-6`, где 0 — это воскресенье).
- `month`: месяц для формирования отчета (`1-12` или `*` для ежемесячного отчета).

Пример:

```
schedule:
  hours: "10"
  minutes: "0"
  seconds: "0"
  dayOfMonth: "*"
  dayOfWeek: "*"
  month: "*"
```

В этом примере:

- Отчет будет формироваться каждый день в 10:00:00 утра, так как `hours` установлено на "10", а `minutes` и `seconds` установлены на "0".
- Поле `dayOfMonth` установлено на `*`, что означает формирование отчета каждый день месяца.
- Поля `dayOfWeek` и `month` также установлены на `*`, что задает ежедневное и ежемесячное формирование отчета без ограничений по конкретным дням недели или месяцам.

Поле `variables`

Поле `variables` используется для передачи переменных в шаблон отчета.

- `key`: уникальный идентификатор переменной.
- `type`: тип источника данных. Допустимые значения:
  - `event_storage` — хранилище событий;
  - `active_list` — активный список;
  - `metric` — метрики;
  - `alert` — оповещения.

Пример:

```
variables:
  - key: VARIABLE_EVENT_STORAGE_ID_1
    type: event_storage
```

В этом примере:

- Переменная с ключом `VARIABLE_EVENT_STORAGE_ID_1` используется для обозначения хранилища событий в шаблоне отчета. Тип переменной установлен как `event_storage`, что указывает на использование хранилища событий в запросах данных.

Поле `layout`

Поле `layout` определяет размещение виджетов в шаблоне отчета. Каждый виджет имеет уникальный идентификатор и параметры, определяющие его положение и размеры.

- `id`: уникальный идентификатор виджета.
- `x`, `y`: координаты виджета на сетке (`x` — позиция по горизонтали, `y` — по вертикали).
- `w`, `h`: размеры виджета в ячейках (ширина `w` и высота `h`).
- `maxH`, `minH`, `minW`: ограничения по минимальной и максимальной высоте и ширине.

Пример:

```
layout:
  - id: WIDGET_KEY_1
    x: 0
    y: 1
    w: 6
    h: 3
    maxH: 5
    minH: 3
    minW: 6
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` размещен в позиции `x: 0`, `y: 1`, что означает его расположение в первом столбце и второй строке сетки шаблона отчета.
- Размер виджета составляет 6 ячеек по ширине (`w: 6`) и 3 ячейки по высоте (`h: 3`).
- Ограничения для виджета установлены: максимальная высота — 5 ячеек, минимальная высота — 3 ячейки, минимальная ширина — 6 ячеек.

Поле `widgets`

Поле `widgets` описывает каждый виджет, его параметры данных и тип отображаемой информации. Основные параметры виджетов:

- `id`: уникальный идентификатор виджета.
- `name`: название виджета.
- `description`: краткое описание функциональности виджета.
- `autoUpdate`: настройки автоматического обновления данных (единица времени и интервал).
- `last`: временной период отображаемых данных (например, последний день или месяц).
- `variables`: параметры запроса данных для виджета.

Пример:

```
widgets:
  - id: WIDGET_KEY_1
    name: "Host DB RAM"
    description: "Отображает текущее использование RAM базой данных"
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: DAY
      value: 1
    variables:
      source: metric
      metricId: c9c9e971-7d45-48eb-9ed5-000000000000
      queryArguments: []
      step: ONE_DAY
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` имеет название "Host DB RAM" и предназначен для отображения использования оперативной памяти базы данных.
- Описание виджета указано в поле `description` как "Отображает текущее использование RAM базой данных".
- Виджет настроен на автоматическое обновление каждые 1 минуту, что задано в поле `autoUpdate`.
- Поле `last` указывает, что виджет будет отображать данные за последний день.
- Источник данных для виджета — метрика (`source: metric`), с идентификатором метрики

`metricId: c9c9e971-7d45-48eb-9ed5-000000000000`. Дополнительные параметры запроса определены как `queryArguments: []`, и шаг агрегации данных установлен на один день (`step: ONE_DAY`).

Структура виджетов

Виджеты в шаблоне отчета могут быть нескольких типов в зависимости от отображаемых данных и формата представления. Каждый тип виджета имеет уникальные параметры.

Виджеты с метриками

Метрики отображают количественные данные, такие как использование ресурсов системы, количество событий и другие параметры, собираемые из источников.

Поля виджета с метриками:

- `source`: определяет источник данных для виджета (допустимые значения: `metric`, `SEARCH`, `alert`).
- `metricId`: идентификатор метрики, данные которой отображаются.
- `step`: интервал времени, за который агрегируются данные.

Пример виджета с метрикой:

```
widgets:
  - id: WIDGET_KEY_1
    name: "CPU Usage"
    description: "Отображает использование CPU системы"
    autoUpdate:
      unit: MINUTE
      value: 5
    last:
      unit: HOUR
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
      step: ONE_MIN
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` называется "CPU Usage" и предназначен для отображения текущей загрузки процессора системы.
- Поле `description` содержит описание "Отображает использование CPU системы".
- Виджет обновляется каждые 5 минут, что задано в поле `autoUpdate` (`unit: MINUTE`, `value: 5`).
- Поле `last` указывает, что данные за последний час будут агрегироваться и отображаться на виджете (`unit: HOUR`, `value: 1`).
- Источник данных для виджета определен в поле `variables`. Здесь указывается, что данные берутся из метрики (`source: metric`), с идентификатором метрики `metricId: "c9c9e971-7d45-`

48eb-9ed5-000000000000", а интервал агрегации данных установлен на одну минуту (step: ONE_MIN).

Виджеты с графиками

Графики визуализируют изменение данных во времени и поддерживают несколько типов отображения: линейные графики, столбчатые диаграммы, круговые диаграммы и др.

Поля виджета с графиком:

- chartType: тип графика (line_chart, bar_chart, pie_chart, table, total).
- rql: запрос на языке RQL для получения данных.
- nameMapping: сопоставление полей запроса с осями графика (например, x — время, y — количество).
- limit: лимит строк для виджета.
- searchEntity: идентификатор переменной и тип источника данных.

Пример виджета с линейным графиком:

```
widgets:
  - id: WIDGET_KEY_2
    name: "Event Storage Counter"
    description: "Отображает количество событий в хранилище"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: line_chart
      rql: "SELECT count(id) as cnt, toStartOfHour(timestamp) as ts GROUP BY ts"
      nameMapping:
        - key: x
          value: ts
        - key: y
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором WIDGET_KEY_2 называется "Event Storage Counter" и предназначен для отображения количества событий в хранилище.
- Поле description содержит описание "Отображает количество событий в хранилище".

- Виджет автоматически обновляется каждые 30 минут, что задано в поле `autoUpdate` (`unit: MINUTE`, `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний месяц (`unit: MONTH`, `value: 1`).
- В поле `variables` виджет использует источник данных типа `search` и тип графика `line_chart`, что означает, что данные будут отображаться в виде линейного графика.
- Запрос на языке RQL (`rql`) возвращает количество событий (`cnt`), сгруппированных по часам (`toStartOfHour(timestamp)`).
- Поле `nameMapping` используется для сопоставления полей запроса с осями графика: поле `ts` отображается на оси `x`, а количество событий `cnt` — на оси `y`.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Табличные виджеты

Табличные виджеты представляют данные в виде таблицы, показывая несколько столбцов с данными.

Поля табличного виджета:

- `chartType`: всегда имеет значение `table`.
- `rql`: запрос на языке RQL для получения данных, которые будут отображены в таблице.
- `nameMapping`: сопоставление полей со столбцами таблицы (например, `timestamp`, `count`).
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример табличного виджета:

```
widgets:
  - id: WIDGET_KEY_3
    name: "Event Table"
    description: "Отображает события с временными метками"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: DAY
      value: 1
    variables:
      source: search
      chartType: table
      rql: "SELECT timestamp, count(id) as cnt FROM event_table GROUP BY timestamp"
      nameMapping:
        - key: timestamp
          value: timestamp
        - key: count
```

```
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_3` называется "Event Table" и отображает события с указанием временных меток.
- Поле `description` содержит описание "Отображает события с временными метками".
- Виджет автоматически обновляется каждые 30 минут (`autoUpdate` с параметрами `unit: MINUTE` и `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний день (`unit: DAY`, `value: 1`).
- В поле `variables` указано, что виджет использует тип отображения данных `table` (`chartType: TABLE`), а данные запрашиваются из источника типа `search`.
- Запрос на языке RQL (`rql`) получает временные метки (`timestamp`) и количество событий (`cnt`), сгруппированных по времени.
- Поле `nameMapping` сопоставляет поля запроса со столбцами таблицы: поле `timestamp` отображается в столбце с временными метками, а поле `cnt` — в столбце с количеством событий.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Виджеты с оповещениями

Виджеты с оповещениями отображают оповещения и события, классифицированные по уровням угрозы, времени создания и другим параметрам.

Поля виджета с оповещениями:

- `source`: всегда указывается `alert`.
- `kind`: тип отображаемых оповещений (`active`, `new`, `last`, `distributionByField`, `distributionByTime`, `total`).
- `filter`: опциональные фильтры для отображаемых оповещений (например, по уровню угрозы).

Пример виджета с активными оповещениями:

```
widgets:
  - id: WIDGET_KEY_4
    name: "Active Alerts"
    description: "Отображает активные оповещения в системе"
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
```

```
      unit: HOUR
      value: 1
    variables:
      source: alert
      kind:
        kind: active
        active: {}
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_4` называется "Active Alerts" и предназначен для отображения активных оповещений в системе.
- Поле `description` содержит описание "Отображает активные оповещения в системе").
- Виджет обновляется каждую минуту (`autoUpdate` с параметрами `unit: MINUTE`, `value: 1`).
- Поле `last` указывает, что виджет будет отображать оповещения за последний час (`unit: HOUR`, `value: 1`).
- Источник данных для виджета — это оповещения (`source: alert`).
- Поле `kind` указывает, что отображаются только активные оповещения (`kind: active`), без дополнительной фильтрации (параметры в `active: {}` не указаны).

Виджеты со сводными данными

Сводные виджеты отображают итоговые значения по выбранным параметрам, такие как общее количество событий или использования ресурсов.

Поля сводного виджета:

- `chartType`: всегда имеет значение `total`.
- `rql`: запрос на языке RQL для получения сводных данных.
- `nameMapping`: сопоставление результата с отображаемым полем.
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример сводного виджета:

```
widgets:
  - id: WIDGET_KEY_5
    name: "Total Event Count"
    description: "Отображает общее количество событий"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
```

```
      source: search
      chartType: total
      rql: "SELECT count(id) as cnt FROM event_storage"
      nameMapping:
        - key: value
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_5` назван "Total Event Count" и отображает общее количество событий.
- Поле `description` содержит описание "Отображает общее количество событий".
- Виджет автоматически обновляется каждые 30 минут (`autoUpdate` с параметрами `unit: MINUTE` и `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний месяц (`unit: MONTH`, `value: 1`).
- Источник данных — хранилище событий (`source: search`), тип виджета — сводный (`chartType: total`).
- Запрос на языке RQL (`rql`) возвращает общее количество событий в хранилище (`SELECT count(id) as cnt FROM event_storage`).
- Поле `nameMapping` сопоставляет поле `cnt` с отображаемым полем виджета, обозначенным как `value`.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Пример схемы шаблона отчета

Пример схемы шаблона отчета "System Report Template", схема RObject

```
# Универсальный уникальный идентификатор схемы шаблона отчета.
id: 61f0c404-5cb3-11e7-907b-000000000000

# Название шаблона отчета.
name: System Report Template

# Тип элемента.
type: report_template_schema

# Версия схемы.
version: 1.0.0

# Описание назначения шаблона отчета.
description: Шаблон отчета для мониторинга системных метрик и событий
```

```
# Имя и контактные данные автора схемы шаблона отчета.
author: Ivan Ivanov <ivanov@example.ru>

# Список источников данных, с которыми совместима схема шаблона отчета.
data_source:
  - Windows
  - Linux

# Список тегов, присвоенных схеме шаблона отчета для классификации и быстрого поиска.
tags:
  - system
  - report

# Параметры хранения данных.
storagePeriod:
  unit: MINUTE
  value: 60

# Расписание формирования отчета согласно формату cron.
schedule:
  hours: "10"
  minutes: "0"
  seconds: "0"
  dayOfMonth: "*"
  dayOfWeek: "*"
  month: "*"

# Переменные для источников данных виджетов.
variables:
  - key: VARIABLE_EVENT_STORAGE_ID_1
    type: EVENT_STORAGE

# Размещение виджетов в отчете.
layout:
  - id: WIDGET_KEY_1
    x: 0
    y: 1
    w: 6
    h: 3
    maxH: 5
    minH: 3
    minW: 6
  - id: WIDGET_KEY_2
    x: 0
    y: 3
    w: 6
    h: 3
  - id: WIDGET_KEY_3
    x: 0
    y: 6
```

```
    w: 6
    h: 3

# Параметры виджетов в отчете.
widgets:
  - id: WIDGET_KEY_1
    name: Host DB RAM
    description: "Отображает текущее использование RAM базой данных"
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: DAY
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
      queryArguments: []

  - id: WIDGET_KEY_2
    name: Event Storage Counter (LineChart)
    description: "Линейный график для отображения количества событий в хранилище"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: line_chart
      rql: "SELECT count(id) as cnt, toStartOfHour(timestamp) as ts GROUP BY ts"
      nameMapping:
        - key: x
          value: ts
        - key: y
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage

  - id: WIDGET_KEY_3
    name: "Оповещения по уровню угрозы за месяц"
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: MINUTE
      value: 3
    variables:
```

```
      source: ALERT
      kind:
        kind: active
        active: {}
```

Интерпретация

Метаданные шаблона отчета:

- `id`: универсальный уникальный идентификатор схемы шаблона отчета — "61f0c404-5cb3-11e7-907b-000000000000".
- `name`: название шаблона отчета — "System Report Template".
- `type`: указывается как `report_template_schema`, что определяет тип схемы как шаблон отчета.
- `version`: версия схемы — "1.0.0".
- `description`: краткое описание назначения шаблона — "Шаблон отчета для мониторинга системных метрик и событий".
- `author`: имя и контактные данные автора шаблона — "Ivan Ivanov <ivanov@example.ru>".
- `data_source`: список совместимых источников данных — "Windows" и "Linux".
- `tags`: список тегов, присвоенных шаблону для классификации — "system" и "report".

Параметры хранения данных:

- `storagePeriod`: задает период хранения данных для формирования отчета. В данном примере данные собираются за последние 60 минут.

Расписание формирования отчета:

- `schedule`: отчет будет формироваться ежедневно в 10:00:00.

Переменные:

- `variables`: переменные используются для передачи данных в виджеты. В данном примере переменная `VARIABLE_EVENT_STORAGE_ID_1` используется как идентификатор хранилища событий (`event_storage`).

Разметка виджетов:

- `layout`: описывает размещение виджетов на странице отчета. Например, виджет с идентификатором `WIDGET_KEY_1` размещен в позиции `x: 0`, `y: 1` с шириной 6 и высотой 3.

Виджеты:

1. **Host DB RAM**: отображает текущее использование RAM базой данных. Виджет обновляется каждую минуту и отображает данные за последний день.
   - `source`: метрика (источник данных — `metric`).
   - `metricId`: идентификатор метрики — "c9c9e971-7d45-48eb-9ed5-000000000000".
2. **Event Storage Counter (LineChart)**: отображает количество событий в хранилище в виде линейного

графика за последний месяц. Обновляется каждые 30 минут.

- `source`: данные берутся из хранилища событий (`search`).
- `chartType`: тип графика — линейный (`line_chart`).
- `rql`: запрос на языке RQL возвращает количество событий (`cnt`), сгруппированных по часам.
- `nameMapping`: сопоставляет поле `ts` с осью `x`, а `cnt` с осью `y` на графике.

3. **Оповещения по уровню угрозы за месяц**: отображает активные оповещения за последний месяц. Обновление происходит каждые 3 минуты.
   - `source`: оповещения (`alert`).
   - `kind`: отображаются активные оповещения (`active`), фильтрации нет.

Таблицы обогащения

В данном разделе освещается концепция таблиц обогащения в системе, включая описание их назначения, принципы работы и процесса интеграции в процесс обработки событий. Управление таблицами обогащения происходит в разделе **Экспертиза**.

О таблицах обогащения

Таблица обогащения в системе представляет собой структурированный набор данных, используемый для дополнения и уточнения информации по событиям безопасности. Они служат для обогащения событий дополнительными данными, не содержащимися непосредственно в событиях, но позволяющими улучшить анализ и интерпретацию событий.

Структура таблиц обогащения включает в себя определения полей и типов данных, которые должны содержаться в каждой записи таблицы. Каждая запись в таблице обогащения может включать такие данные, как идентификаторы устройств, IP-адреса, имена пользователей, географические данные, данные о приложениях и любую другую информацию, которая может быть использована для обогащения событий.

Процесс обогащения событий с использованием таблиц обогащения заключается в сопоставлении данных из событий с записями в таблицах по определенным ключам. Это может быть выполнено автоматически системой во время обработки событий, когда обнаруживается совпадение по ключу, и дополнительные данные из таблицы включаются в информацию о событии.

Таблицы обогащения позволяют выполнить следующие задачи:

- Дополнение событий информацией, недоступной в исходных данных, для улучшения понимания контекста события.
- Уточнение и классификация событий на основе дополнительных параметров, содержащихся в таблицах.
- Повышение точности корреляции событий за счет дополнительных данных для сопоставления.

Для управления таблицами обогащения предусмотрены функции обогащения для получения и добавления записей, а также механизмы для автоматического обновления данных в таблицах с использованием внешних источников.

Работа с таблицей обогащения

Доступные операции над таблицей обогащения:

- Создание таблицы обогащения
- Изменение таблицы обогащения
- Удаление таблицы обогащения

Создание таблицы обогащения

Чтобы добавить таблицу обогащения:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Таблица обогащения**. Система отобразит окно создания таблицы обогащения.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля таблицы обогащения, чтобы определить логику ее работы.
4. Нажмите на кнопку **Опубликовать версию**, чтобы сохранить изменения и опубликовать созданную таблицу обогащения. Система создаст таблицу обогащения и отобразит соответствующее уведомление. Новая таблица появится в списке элементов экспертизы.



Чтобы таблица обогащения стала доступна на конвейерах событий, ее необходимо включить.

Вы можете создать черновик таблицы обогащения без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик таблицы обогащения и отобразит соответствующее уведомление. Новый черновик появится в списке элементов экспертизы.

Изменение таблицы обогащения

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление таблицы обогащения

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура таблицы обогащения

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор таблицы обогащения.	строка	да
`name`	Название таблицы обогащения. Допустимы строчные и прописные буквы латинского алфавита, цифры, подчеркивания и дефис. Пробелы не допускаются.	строка	да
`type`	Тип элемента. Всегда имеет значение `enrichment_table` для элементов этого типа.	строка	да
`version`	Версия таблицы обогащения, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора таблицы обогащения.	строка	нет
`description`	Описание таблицы обогащения, содержащее дополнительную информацию о ее назначении и функциональности.	строка	нет
`tags`	Список тегов, присвоенных таблице обогащения для классификации и быстрого поиска.	массив строк	нет
`kind`	Тип хранилища таблицы обогащения, указывающий на формат (`table`, `maxmind`).	строка	да
`location`	Способ коммуникации с хранилищем таблицы обогащения (`file`, `embedded`).	строка	да
`schema`	Схема, описывающая поля таблицы обогащения и их типы данных. Поддерживаются типы `string`, `bool`, `float`, `integer`, `date`, `timestamp`.	объект	да
`content`	Содержимое таблицы для режима `embedded`, включающее предопределенные данные. Обязательно только для таблиц с `location: embedded`.	массив объектов	условно (требуется только для `embedded`)
`update_frequency_min`	Частота обновления данных в минутах для режимов, отличных от `embedded`. Определяет, как часто данные должны обновляться из внешнего источника.	число	условно (для `file`)
`uri`	Расположение файла или источника данных.	строка	условно (для `file`)



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных

в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, которое позволяет уникально идентифицировать его результаты в выводе тестирования.	строка	да
`events`	Набор тестовых событий, имитирующих входные данные, которые будут обрабатываться правилом. Это позволяет проверить реакцию правила на разнообразные ситуации.	массив	да
`assertion`	Блок утверждений на языке VRL, применяемых к результатам обработки тестовых событий правилом. Функции `assert` и `assert_eq` используются для проверки соответствия фактических результатов обработки ожидаемым значениям.	строка	да

Эта структура позволяет определить тесты для проверки функциональности таблицы обогащения, включая корректность обработки данных и соответствие результатов обогащения заданным условиям.

Составление таблицы обогащения



Управление таблицами обогащения в системе осуществляется в разделе **Экспертиза**.

Для создания таблицы обогащения в системе необходимо определить ее ключевые элементы структуры. Эти элементы охватывают как метаданные таблицы, так и параметры, определяющие содержание и методы доступа к данным.

1. Определите метаданные таблицы:
    - `id`: уникальный идентификатор таблицы обогащения.
    - `name`: название таблицы обогащения. Допустимы строчные и прописные буквы латинского алфавита, цифры, точки, подчеркивания и дефис. Пробелы не допускаются.
    - `version`: версия таблицы, следующая стандартам Semantic Versioning.
    - `type`: тип элемента, указываемый как `enrichment_table`.
    - `author` (опционально): имя и контактные данные автора таблицы.
    - `description` (опционально): краткое описание назначения и функциональности таблицы.
    - `tags` (опционально): теги для классификации и поиска таблицы.
2. Настройте параметры доступа и структуру данных:
    - `kind`: тип хранилища данных (например, `table`, `maxmind`).
    - `location`: метод доступа к данным (например, `file`, `embedded`).
    - `schema`: схема данных, определяющая структуру и типы полей таблицы.
    - `content` (для `embedded`): содержимое таблицы, предоставленное непосредственно в конфигурации для встроенного хранения.
3. Установите параметры обновления данных (для режимов, отличных от `embedded`):
    - `update_frequency_min`: частота обновления данных в минутах.
    - `uri` (для `file`): расположение файла или источника данных.

Поля метаданных

Метаданные служат для идентификации и описания таблицы обогащения в системе. Заполнение метаданных должно производиться в соответствии со следующими указаниями:

- `id`: присвойте уникальный идентификатор таблицы обогащения. Идентификатор должен быть уникальным в пределах системы.
- `name`: укажите название таблицы обогащения. Допустимы строчные и прописные буквы латинского алфавита, цифры, точки, подчеркивания и дефис. Пробелы не допускаются.
- `version`: определите версию таблицы, следуя принципам Semantic Versioning (например, 1.0.0).
- `type`: укажите тип элемента как `enrichment_table`.
- `author` (опциональное): укажите имя и контактные данные автора таблицы.
- `description` (опциональное): предоставьте краткое описание целей и функционала таблицы обогащения.
- `tags` (опциональное): добавьте теги для удобства классификации и поиска таблицы в системе.

Поле `kind`

Поле `kind` в таблицах обогащения определяет тип источника данных, используемого для обогащения информации. Это поле позволяет системе определить, каким образом должны быть загружены и обработаны данные, а также какие операции поддерживаются для данного типа таблицы. Важность этого поля заключается в его влиянии на методы доступа к данным и способы их использования в процессе обогащения.

Типы источников данных

- `table`: указывает, что данные для обогащения хранятся в структурированной таблице, которая может быть загружена из файла, получена через API или встроена непосредственно в конфигурацию.
- `maxmind`: определяет использование специализированных геолокационных баз данных MaxMind для обогащения данных, например, для определения местоположения по IP-адресу.

Выбор типа источника данных

Выбор между `table` и `maxmind` зависит от конкретных задач обогащения и доступности данных. Для геолокационного обогащения чаще всего используется тип `maxmind`, тогда как для других видов обогащения может быть выбран тип `table`.

Пример использования поля `kind`:

```
kind: table
```

или

```
kind: maxmind
```



Выбор типа `kind` должен соответствовать целям обогащения и типу обрабатываемых данных. Важно учитывать, что выбор типа источника данных влияет на конфигурацию таблицы обогащения и ее дальнейшее использование в системе.



При планировании использования таблиц обогащения рекомендуется заранее определить требуемый тип источника данных, чтобы обеспечить оптимальную конфигурацию и эффективность процесса обогащения.

Поле `location`

Поле `location` в таблицах обогащения указывает на метод хранения и доступа к данным таблицы обогащения. Этот параметр критически важен для определения того, как система будет взаимодействовать с данными, включая их загрузку, обновление и использование в процессе обогащения.

Значения поля `location` и их описание

- `file`: данные таблицы обогащения располагаются в файле. Система обращается к файлу для чтения данных. Путь к файлу указывается отдельно в конфигурации.
- `embedded`: данные встроены непосредственно в конфигурацию таблицы обогащения. Этот метод подходит для статических данных, которые не требуют регулярных обновлений.

Выбор метода `location`

Пример:

```
location: file
```

или

```
location: embedded
```

Поле `schema`

Поле `schema` в таблицах обогащения определяет структуру данных, которая будет использоваться в таблице. Оно задает типы данных для каждого поля в таблице обогащения, обеспечивая корректную обработку и интерпретацию данных при их использовании.

Определите поля данных. Каждое поле в `schema` должно иметь указание своего типа данных, что позволяет системе правильно обрабатывать входящие значения.

Доступные типы данных:

- `string` — для текстовой информации.
- `bool` — для булевых значений: истина или ложь.
- `float` — для чисел с плавающей точкой.
- `integer` — для целочисленных значений.
- `date` — для дат.
- `timestamp` — для меток времени.

Пример:

```
schema:
  user_id: integer
  user_email: string
  login_time: timestamp
  is_active: bool
```

Используйте `schema` для точного определения ожидаемых данных. Это включает в себя не только типы полей, но и их назначение. Например, поле `user_email` должно содержать адрес электронной почты пользователя, а `login_time` — временную метку его последнего входа в систему.

Поле `content`

Поле `content` используется для хранения данных непосредственно в конфигурации таблицы, когда выбран режим `location: embedded`. Это поле позволяет интегрировать статические данные в таблицу обогащения без необходимости их загрузки из внешних источников. `content` обеспечивает немедленный доступ к данным для обогащения без задержек, связанных с их извлечением или обновлением.

Структура и заполнение поля `content`

`content` является массивом объектов, где каждый объект представляет собой запись данных в таблице. Структура каждого объекта должна соответствовать схеме, определенной в поле `schema`.

Пример:

```
content:
  - id: 1
    name: "Example Name 1"
    type: "Type1"
  - id: 2
    name: "Example Name 2"
    type: "Type2"
```



Использование `content` требует тщательного планирования структуры данных и их соответствия схеме таблицы. Данные должны быть актуальными и достоверными, поскольку они будут встроены в таблицу без возможности динамического обновления (за исключением ручного изменения конфигурации). `content` подходит для данных, которые не изменяются или обновляются редко, таких как списки контрольных значений, статические справочники и прочие.

Рекомендации по заполнению

- Перед включением данных в `content` убедитесь, что они точно соответствуют определенной в `schema` структуре. Некорректно структурированные данные могут привести к ошибкам при их обработке.
- Для удобства управления и возможного обновления рассмотрите возможность разделения больших объемов данных на несколько таблиц обогащения, особенно если данные относятся к различным категориям или имеют разное назначение.



Включение данных в `content` увеличивает независимость от внешних источников и сетевой инфраструктуры, однако требует аккуратности в подготовке и проверке этих данных для обеспечения корректности обогащения.

Поле `update_frequency_min`

Поле `update_frequency_min` определяет периодичность обновления данных в таблице обогащения в минутах. Этот параметр актуален для таблиц, метод доступа к данным которых определен как `file` в поле `location`. Значение поля позволяет настроить частоту, с которой система будет инициировать процесс обновления данных, чтобы поддерживать их актуальность.

Принципы использования

- Значение `update_frequency_min` должно быть выбрано с учетом баланса между необходимостью актуальности данных и минимизацией нагрузки на системные ресурсы.
- Для данных, подверженных частым изменениям, рекомендуется устанавливать меньшее значение, обеспечивая тем самым их своевременное обновление.

- Для статических данных, которые изменяются редко, целесообразно выбирать большее значение интервала обновления.

Пример:

```
update_frequency_min: 30
```

В данном примере данные в таблице обогащения будут обновляться каждые 30 минут.

Рекомендации по заполнению

- Оцените частоту изменений в данных источника и потребности бизнес-процессов в актуальной информации при настройке `update_frequency_min`.
- Учитывайте возможности инфраструктуры и избегайте чрезмерной нагрузки на систему, оптимизируя частоту обновлений.



Настройка `update_frequency_min` для каждой таблицы обогащения отдельно позволяет точнее контролировать актуальность данных в соответствии с их спецификой и обновлениями. Использование разных интервалов обновления дает возможность более гибкого реагирования на потребности в данных, оптимизируя ресурсы и улучшая эффективность обогащения.

Поле `uri`

Поле `uri` в таблицах обогащения указывает на расположение файла или источника данных, из которого таблица будет получать обновления. Это поле обязательно для метода доступа `file`, поскольку оно определяет, где именно хранятся данные и как они будут извлекаться.

Пример:

```
http://10.0.0.1/test.csv
```

Функции обогащения



При обращении к таблице обогащения в VRL-песочнице будет возвращено значение `null`.

Функции обогащения используются для поиска и извлечения данных из таблиц обогащения. Эти функции позволяют обогатить события безопасности дополнительной информацией, не содержащейся непосредственно в событиях. Взаимодействие с таблицами обогащения может быть настроено в любом поле, использующем язык VRL.

Для взаимодействия с таблицами обогащения предусмотрены VRL функции поиска значений в таблице обогащения `find_enrichment_table_records` и извлечения значения из таблицы обогащения `get_enrichment_table_record`.

Функция поиска значений в таблице обогащения

```
find_enrichment_table_records(таблица: <строка>, условие: <объект>, [select: <массив>,
case_sensitive: <логическое значение>])
:: <массив>
```

Поиск записей в таблице обогащения, которые соответствуют заданным условиям. Возвращает массив объектов, соответствующих критериям поиска.

Функция извлечения значения из таблицы обогащения

```
get_enrichment_table_record(table: <строка>, condition: <объект>, [select: <массив>,
case_sensitive: <логическое значение>])
:: <объект>, <ошибка>
```

Извлекает единственную запись из таблицы обогащения, точно соответствующую указанным условиям. В случае, если запись не найдена или найдено более одной записи, возвращается ошибка.

Функция	Описание	Возвращаемый тип
`find_enrichment_table_records`	Поиск множества записей в таблице обогащения по заданным критериям.	Массив объектов
`get_enrichment_table_record`	Поиск одной конкретной записи, точно соответствующей условиям поиска.	Объект или ошибка

Пример использования функции `find_enrichment_table_records`*:*

```
find_enrichment_table_records("geoip", {"ip": .source_ip})
```

Данная функция ищет записи в таблице обогащения `geoip`, соответствующие IP-адресу источника. Возвращает массив объектов, каждый из которых содержит информацию о геолокации по IP-адресу.

Пример использования функции `get_enrichment_table_record`*:*

```
get_enrichment_table_record("user_profiles", {"user_id": .user_id})
```

Функция ищет в таблице обогащения `user_profiles` запись, точно соответствующую идентификатору пользователя. Возвращает объект с профилем пользователя или ошибку, если запись не найдена или найдено несколько записей.

Пример использования в поле `on_correlate`*:*

```
on_correlate: !vrl |
  geo_info = find_enrichment_table_records!("geoip", {"ip": .source_ip}, select:
["country_name", "city_name"])
  if is_empty(geo_info) {
    .source_country = geo_info[0].country_name
    .source_city = geo_info[0].city_name
  }
```

```
  threat_info = get_enrichment_table_record!("threat_intelligence", {"ip":
.source_ip}, select: ["category", "score"])
  if !is_null(threat_info) {
    .threat_category = threat_info.category
    .threat_score = threat_info.score
  }
```

В этом примере:

- Выполняется поиск геолокационной информации по IP-адресу источника события в таблице `geoip`, и результаты добавляются к событию как `.source_country` и `.source_city`.
- Затем производится запрос к таблице `threat_intelligence` для извлечения информации о категории угрозы и оценке уровня угрозы, ассоциированных с IP-адресом источника. Эти данные добавляются к событию как `.threat_category` и `.threat_score`.

Пример таблицы обогащения

Пример таблицы обогащения "DDoSSuspiciousPatterns", схема RObject

```
# Уникальный идентификатор таблицы обогащения.
id: security/ddos_suspicious_patterns

# Название таблицы обогащения.
name: DDoSSuspiciousPatterns

# Тип элемента экспертизы.
type: enrichment_table

# Версия таблицы.
version: 1.0.0

# Описание назначения таблицы.
description: Содержит список подозрительных паттернов запросов и IP-адресов,
ассоциированных с DDoS атаками.

# Имя и контактные данные автора таблицы.
author: Ivan Ivanov <ivanov@example.ru>

# Теги, связанные с таблицей.
tags:
  - ddos
  - security

# Способ коммуникации с хранилищем таблицы.
location: embedded

# Тип хранилища таблицы.
kind: table
```

```
# Схема описывает поля таблицы обогащения.
schema:
  ip_address: string
  pattern: string
  risk_level: integer

# Содержимое таблицы для `embedded` режима.
content:
  - ip_address: "192.168.1.1"
    pattern: "/api/highload"
    risk_level: 5
  - ip_address: "192.168.100.100"
    pattern: "/api/highload"
    risk_level: 5
  - ip_address: "172.16.0.2"
    pattern: "/contact"
    risk_level: 2
  - ip_address: "192.168.100.132"
    pattern: "/api/highload"
    risk_level: 5
```

Интерпретация

Метаданные таблицы обогащения:

- `id`: `security/ddos_suspicious_patterns` — уникальный идентификатор, позволяющий идентифицировать таблицу в системе.
- `name`: `DDoSSuspiciousPatterns` — название таблицы, отражающее ее предназначение для хранения данных о подозрительных паттернах запросов, связанных с DDoS атаками.
- `type`: `enrichment_table` — указывает на тип объекта в системе, подтверждая его принадлежность к таблицам обогащения.
- `version`: `1.0.0` — версия таблицы обогащения, следует принципам Semantic Versioning, обеспечивая управление изменениями и совместимостью.
- `description`: описывает назначение таблицы, в данном случае содержание списка подозрительных паттернов запросов и IP-адресов, ассоциированных с DDoS атаками.
- `author`: `Ivan Ivanov <ivanov@example.ru>` — информация об авторе таблицы, включая контактные данные.
- `tags`: `ddos` и `security` — теги, упрощающие поиск и классификацию таблицы в системе.

Параметры доступа и хранения данных:

- `location`: `embedded` — указывает на то, что данные таблицы встроены непосредственно в конфигурацию, что упрощает доступ к ним и ускоряет процесс обогащения.
- `kind`: `table` — подтверждает, что данные организованы в виде структурированной таблицы, подходящей для поиска и обработки информации.

Структура данных (схема и содержимое):

- `schema` описывает структуру данных в таблице, включая поля `ip_address` (строка), `pattern` (строка) и `risk_level` (целое число), что позволяет систематически хранить информацию о подозрительных IP-адресах, соответствующих паттернах запросов и уровне риска.
- `content` содержит конкретные записи данных, например, IP-адреса с ассоциированными с ними паттернами запросов и уровнем риска, демонстрируя реальные примеры данных, используемых для обогащения событий.

Обогащение событий:

- Когда событие содержит IP-адрес или паттерн запроса, система проверяет наличие соответствующих записей в таблице "DDoSSuspiciousPatterns".

Сопоставление событий с записями в таблице:

- Для каждого события система ищет записи в таблице, где `ip_address` совпадает с IP-адресом источника события или `pattern` совпадает с паттерном запроса, указанным в событии.
- Если сопоставление успешно, информация о `risk_level` из таблицы добавляется к событию, указывая на уровень риска, связанный с обнаруженным IP-адресом или паттерном запроса.

Шаблоны поиска

Шаблоны поиска в системе используются для настройки конфигурации и условий RQL-запросов. Они применяются в разделе **Поиск** и виджетах дашбордов для отображения аналитических данных.

Управление конфигурациями шаблонов поиска осуществляется в разделе **Экспертиза**. Шаблоны позволяют задавать фильтры, временные интервалы и RQL-запросы для выборки событий, упрощая анализ данных и создание визуализаций.

О шаблонах поиска

Шаблон поиска — это структурированный объект данных в формате RObject (.ro), который определяет параметры и условия для выполнения запросов к данным в системе в разделе **Поиск** и виджетах дашбордов. Их можно создавать на основе запросов, введенных в этих разделах, или непосредственно через раздел Экспертиза. Шаблон поиска задает структуру фильтров, временной интервал и параметры выборки, которые могут быть использованы для выполнения поисковых запросов по событиям системы.

Шаблон поиска включает следующие компоненты: фильтры для настройки условий отбора данных, временные параметры и RQL-запрос, формирующий логику поиска. Фильтры определяют поля и условия, по которым данные отбираются, временные параметры указывают период выборки, а запрос на языке RQL задает основную логику обработки данных.

Каждый фильтр включает описание целевого поля данных, логический оператор сравнения и значения, по которым данные должны отбираться. В качестве условий могут использоваться операторы `EQ`, `BETWEEN`, `LIKE`, `IN`, `GT`, `GE`, `LT`, `LE`, и `NE`.

Шаблон поиска позволяет настраивать временные параметры, включая фиксированный период и интервальные значения. Это обеспечивает возможность как статических, так и динамических выборок данных на основе шаблона. Поля фильтров и временных настроек передаются в RQL-запрос, который используется для конечной выборки данных.

Работа с шаблоном поиска

Шаблоны поиска используются в разделе **Поиск**.

Доступные операции над шаблоном поиска:

- Создание шаблона
- Применение шаблона
- Поиск шаблона
- Обновление шаблона

Создание шаблона

Чтобы создать шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Нажмите на кнопку **Создать шаблон**. Отобразится окно сохранения шаблона.
4. Введите название шаблона.
5. При необходимости введите описание шаблона.
6. Выберите каталог раздела **Экспертиза**, в который будет помещен шаблон.
7. Чтобы сохранить в шаблоне текущий заданный период, установите флажок **Сохранить временной период**.
8. Чтобы сохранить в шаблоне имеющиеся пользовательские фильтры, установите флажок **Сохранить фильтры**.
9. Нажмите на кнопку **Создать**. Система создаст шаблон поиска и отобразит соответствующее уведомление. Новый шаблон появится на панели шаблонов.

Применение шаблона

Чтобы применить шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Выберите шаблон из списка. Система применит шаблон для поиска событий.

Поиск шаблона

Чтобы найти шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Введите название требуемого шаблона в поле поиска. На панели отобразятся шаблоны, в названии которых встречается введенный запрос.

Обновление шаблона

Вы можете обновить существующий шаблон. Для этого:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Вы можете отредактировать название и описание шаблона. Также можно установить или снять флажки **Сохранить временной период** и **Сохранить фильтры**.
4. Нажмите на кнопку **Обновить** и подтвердите обновление. Система сохранит изменения и отобразит соответствующее уведомление. В шаблоне будет сохранен поисковый запрос, введенный в поле запроса на момент редактирования.

Работа с RObject-конфигурацией шаблона поиска

- Создание шаблона поиска
- Изменение шаблона поиска
- Удаление шаблона поиска

Создание шаблона поиска

Чтобы создать шаблон поиска:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Шаблон поиска**. Система отобразит окно создания шаблона.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля шаблона поиска, чтобы задать параметры поиска и условия фильтрации.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст шаблон поиска и отобразит соответствующее уведомление. Новый шаблон появится в таблице раздела **Экспертиза**.



Чтобы шаблон поиска стал доступным для использования, его необходимо включить.

Вы можете создать черновик шаблона поиска без публикации, используя кнопку **Сохранить черновик**. Система создаст черновик и отобразит соответствующее уведомление. Черновик отобразится в таблице раздела **Экспертиза**.



Вы также можете создать конфигурацию шаблона поиска на основе существующего запроса.

Изменение шаблона поиска

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление шаблона поиска

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура шаблона поиска

Шаблон поиска содержит обязательные и опциональные поля, определяющие его логику и настройки для отбора данных.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор шаблона поиска. Должен быть уникальным в пределах экспертизы. Можно использовать UUID в качестве значения.	Строка	да
`name`	Название шаблона поиска. Допустимы латинские буквы, цифры, точки, подчеркивания и дефисы.	строка	да
`description`	Краткое описание назначения шаблона поиска.	строка	нет
`type`	Тип шаблона. Для шаблонов поиска всегда указывается `search_template`.	строка	да
`author`	Имя и контактные данные автора шаблона поиска.	строка	нет
`date`	Дата создания или изменения шаблона. Задается в формате `YYYY-MM-DD`, то есть "год-месяц-день". Поле не связано с полями **Дата создания** и **Дата изменения** в карточке элемента экспертизы.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`status`	Статус разработки текущей версии шаблона поиска. Может принимать любые текстовые значения. Примеры значений: • `debug` — версия в разработке; • `experimental` — экспериментальная версия; • `test` — версия для тестирования; • `stable` — стабильная версия.	строка	нет
`tags`	Список тегов, присвоенных шаблону поиска для классификации и быстрого поиска.	массив строк	нет
`version`	Версия шаблона поиска в формате Semantic Versioning.	строка	да
`time`	Временной интервал для выборки данных. Может включать период (`from`, `to`) или интервальное значение (`unit`, `value`).	объект	да
`rql`	RQL-запрос, определяющий логику поиска и выборки данных.	строка	да



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

Структура `filters`

Поле `filters` описывает настройки условий для отбора данных в шаблоне поиска. Каждый фильтр включает параметры для задания условий отбора и дополнительных параметров конфигурации.

Поле	Описание	Тип данных	Обязательное поле
`name`	Название целевого поля данных, к которому применяется фильтр.	строка	да
`operation`	Оператор сравнения, определяющий логику фильтрации. Допустимые значения: `EQ`, `NE`, `GT`, `LT`, `GE`, `LE`, `BETWEEN`, `LIKE`, `IN`.	строка	да
`values`	Массив значений, используемых для фильтрации. Для операций `IN` допускается указание нескольких значений, а для `BETWEEN` — строго два значения (границы интервала).	массив строк	да
`enabled`	Статус фильтра, указывающий, будет ли он применен (`true`) или отключен (`false`).	булево значение	да
`inverted`	Опция инверсии условия фильтрации. Если установлено в `true`, условие фильтрации будет инвертировано.	булево значение	нет
`customTitle`	Пользовательское название фильтра, отображаемое в интерфейсе на панели фильтров. Если значение не задано, используется значение поля `name`.	строка	нет

Структура `time`

Поле `time` определяет временной интервал, за который выполняется выборка данных в шаблоне поиска. Интервал может задаваться двумя способами: с помощью фиксированного периода (`from` и `to`) или через интервальные параметры (`unit` и `value`).

Поле	Описание	Тип данных	Обязательное поле
`from`	Начальная точка временного интервала в формате `YYYY-MM-DD HH:MM:SS`. Используется в паре с полем `to`.	строка	да (если `to` указано)
`to`	Конечная точка временного интервала в формате `YYYY-MM-DD HH:MM:SS`. Используется в паре с полем `from`.	строка	да (если `from` указано)
`unit`	Единица измерения временного интервала. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`. Применяется в паре с полем `value`.	строка	да (если `value` указано)

Поле	Описание	Тип данных	Обязательное поле
`value`	Количество единиц, задающих продолжительность временного интервала. Например, при значениях `unit: DAY` и `value: 7` интервал составит последние 7 дней.	число	да (если `unit` указано)

Составление шаблона поиска

Для создания шаблона поиска необходимо определить ряд обязательных и опциональных полей, которые формируют структуру и логику поиска.

Поля метаданных

Заполните поля метаданных шаблона поиска:

- `id`: уникальный идентификатор шаблона поиска. Должен быть уникальным в пределах экспертизы.
- `name`: название шаблона поиска. Допустимы латинские буквы, цифры, символы подчеркивания, точки и дефисы.
- `type`: всегда указывается `search_template` для элементов экспертизы данного типа.
- `description` (опционально): краткое описание назначения шаблона.
- `author` (опционально): имя и контактные данные автора шаблона поиска.
- `version`: версия шаблона поиска согласно стандарту Semantic Versioning.
- `status` (опционально): статус разработки шаблона. Может принимать текстовые значения, такие как:
  - `draft`: черновая версия.
  - `published`: опубликованная версия.
  - `deprecated`: устаревшая версия.
- `tags` (опционально): список тегов, присвоенных шаблону для классификации и быстрого поиска.
- `date` (опционально): дата создания или изменения шаблона. Формат значения: `YYYY-MM-DD`.

Пример метаданных:

```
id: 61f0c404-5cb3-11e7-907b-000000000000
name: Failed Logins
type: search_template
description: Шаблон поиска для неудачных попыток входа
author: Ivan Ivanov <ivanov@example.ru>
version: 1.0.0
status: published
date: 2024-01-15
tags:
  - security
```

```
  - login
```

В этом примере:

- Поле `id`: уникальный идентификатор шаблона поиска. В данном случае UUID — `61f0c404-5cb3-11e7-907b-000000000000`.
- Поле `name`: название шаблона — "Failed Logins".
- Поле `type`: тип шаблона — `search_template`.
- Поле `description`: краткое описание — "Шаблон поиска для неудачных попыток входа".
- Поле `author`: имя автора шаблона — "Ivan Ivanov <ivanov@example.ru>".
- Поле `version`: версия шаблона в формате `1.0.0`.
- Поле `status`: статус разработки шаблона — `published`.
- Поле `date`: дата создания шаблона — "2024-01-15".
- Поле `tags`: содержит теги `security` и `login` для классификации шаблона.

Поле `filters`

Поле `filters` используется для настройки условий фильтрации данных. Каждый фильтр определяет параметр для отбора, логический оператор и значения.

- `name`: название целевого поля данных для фильтрации.
- `operation`: логический оператор для сравнения значений (например, `EQ`, `BETWEEN`, `LIKE`, `IN`).
- `values`: массив значений для сравнения с полем.



Для операций `IN` допускается указание нескольких значений, а для `BETWEEN` — строго два значения (границы интервала).

- `inverted`: указывает, применяется ли логическое отрицание к результату фильтрации.
- `enabled`: указывает, включен ли фильтр для обработки.
- `customTitle` (опционально): заголовок фильтра, который отображается на панели фильтров.

Пример фильтра:

```
filters:
  - name: action
    values:
      - login
    operation: EQ
    inverted: false
    enabled: true
    customTitle: Action Filter
  - name: status
    values:
      - failed
    operation: EQ
```

```
    inverted: false
    enabled: true
    customTitle: Status Filter
```

В этом примере:

- Первый фильтр проверяет, равно ли значение поля `action` значению `login`.
- Второй фильтр проверяет, равно ли значение поля `status` значению `failed`.
- Оба фильтра включены для обработки (`enabled: true`), и логическое отрицание не применяется (`inverted: false`).
- В полях `customTitle` фильтрам присвоены заголовки `Action Filter` и `Status Filter`, которые отобразятся на панели фильтров.

Поле `time`

Поле `time` определяет временной интервал для выборки данных в шаблоне поиска. Может быть представлено либо фиксированным периодом, либо интервалом времени.

Фиксированный период

- `from`: начало временного интервала.
- `to`: окончание временного интервала.

Пример с фиксированным периодом:

```
time:
  from: 2024-10-25 04:00:00
  to: 2024-10-26 03:59:59
```

В этом примере данные будут отбираться за фиксированный временной период с 25 октября 2024 года, начиная с 04:00:00, по 26 октября 2024 года до 03:59:59.

Интервальный тип

- `unit`: единица измерения интервала времени (например, `DAY`, `HOUR`).
- `value`: длительность интервала в указанных единицах.

Пример с интервальным типом:

```
time:
  unit: DAY
  value: 1
```

В этом примере задается временной интервал в 1 день (`unit: DAY`, `value: 1`). Это позволяет динамически выбирать данные, например, за последние 24 часа от момента выполнения запроса.

Поле `rql`

Поле `rql` определяет запрос на языке RQL для формирования условий поиска в шаблоне.

Пример RQL-запроса:

```
rql: sourceIp = '192.0.2.1' AND dvendor = 'Linux'
```

В этом примере:

- Запрос отбирает записи, где:
  - `sourceIp`: значение поля равно `192.0.2.1`.
  - `dvendor`: значение поля равно `Linux`.

Пример шаблона поиска

Пример шаблона поиска "Failed Login Attempts", схема RObject

```
# Уникальный идентификатор шаблона поиска.
id: a1f0d234-6ac2-4d7e-bf93-000000000000

# Название шаблона поиска.
name: Failed Login Attempts

# Тип элемента экспертизы.
type: search_template

# Версия шаблона.
version: 1.0.0

# Описание назначения шаблона поиска.
description: Шаблон для поиска неудачных попыток входа в систему

# Автор шаблона.
author: Ivan Ivanov <ivanov@example.ru>

# Фильтры для настройки условий поиска.
filters:
  - name: action
    values:
      - login
    operation: EQ
    inverted: false
    enabled: true
    customTitle: Action Filter
  - name: status
    values:
      - failed
    operation: EQ
```

```
    inverted: false
    enabled: true
    customTitle: Status Filter

# Временной период для выборки данных.
time:
  unit: HOUR
  value: 1

# RQL-запрос для выборки данных.
rql: sourceIp = '192.0.2.1' AND dvendor = 'Linux'
```

Интерпретация

Метаданные шаблона поиска:

- `id`: уникальный идентификатор шаблона поиска. В данном случае UUID — `a1f0d234-6ac2-4d7e-bf93-000000000000`.
- `name`: название шаблона поиска — "Failed Login Attempts".
- `type`: тип элемента — всегда указывается как `search_template`.
- `version`: версия шаблона — `1.0.0`.
- `description`: описание назначения шаблона — "Шаблон для поиска неудачных попыток входа в систему".
- `author`: автор шаблона — "Ivan Ivanov <ivanov@example.ru>".

Фильтры:

- `filters`: фильтры позволяют задавать условия для отбора данных:
  - Фильтр `action` с оператором `EQ` задает условие, при котором значение поля `action` должно быть равно `login`.
  - Фильтр `status` также с оператором `EQ` отбирает события, где значение поля `status` равно `failed`.
  - Фильтры отобразятся на панели фильтров с заголовками `Action Filter` и `Status Filter`, которые заданы в полях `customTitle`.

Временной интервал:

- `time`: интервал, в пределах которого будут искаться события. В примере указан интервал в 1 час (`unit` установлен на `HOUR`, `value` равно `1`).

RQL-запрос:

- `rql`: запрос на языке RQL, определяющий логику поиска.
  - Запрос отбирает записи, где:
    - `sourceIp`: значение равно `192.0.2.1`.
    - `dvendor`: значение равно `Linux`.

Глобальные функции

В данном разделе представлен обзор структуры, принципов работы и процесса создания глобальных функций в системе. Управление глобальными функциями в системе осуществляется в разделе Экспертиза.

О глобальных функциях

Глобальная функция задает переиспользуемый блок кода на языке VRL, который можно применять при создании правил корреляции и нормализации. Такие функции позволяют настраивать фильтрацию событий, выполнять вычисления или преобразования данных.

Для использования глобальной функции в конструкторе правиле корреляции необходимо выбрать ее из списка доступных типов условий и указать значения для аргументов.

Глобальные функции помогают упростить настройку правил, избегая дублирования кода, и обеспечивают возможность многократного использования одной логики в различных сценариях.

Работа с глобальной функцией

Доступные операции над глобальной функцией:

- Создание глобальной функции
- Изменение глобальной функции
- Удаление глобальной функции

Создание глобальной функции

Чтобы создать глобальную функцию:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите в выпадающем списке пункт **Глобальная функция**. Система отобразит окно создания функции.



При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля глобальной функции, чтобы определить ее логику и параметры.
4. Нажмите на кнопку **Опубликовать версию**, чтобы сохранить изменения и опубликовать созданную глобальную функцию. Система создаст глобальную функцию и отобразит соответствующее уведомление. Новая функция отобразится в таблице раздела **Экспертиза**.



Чтобы глобальная функция стала доступной для использования в условиях корреляции, ее необходимо включить.

Вы можете создать черновик глобальной функции без публикации, используя кнопку **Сохранить черновик**. Система создаст черновик и отобразит соответствующее уведомление. Черновик отобразится в таблице раздела **Экспертиза**.

Изменение глобальной функции

Способ изменения идентичен для всех элементов экспертизы и приведен в разделе Изменение элемента экспертизы.

Удаление глобальной функции

Способ удаления идентичен для всех элементов экспертизы и приведен в разделе Удаление элемента экспертизы.

Структура глобальной функции

Глобальная функция содержит обязательные и опциональные поля, определяющие ее логику и параметры для обработки данных.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор глобальной функции. Должен быть уникальным в пределах экспертизы. Можно использовать UUID в качестве значения.	строка	да
`name`	Название глобальной функции. Должно начинаться с латинской буквы. В названии допустимы строчные и прописные буквы латинского алфавита, цифры, подчеркивания и дефис. Пробелы не допускаются.	строка	да
`description`	Краткое описание назначения глобальной функции.	строка	нет
`data_source`	Список источников данных, с которыми совместима глобальная функция. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет

Поле	Описание	Тип данных	Обязательное поле
`status`	Статус разработки текущей версии глобальной функции. Может принимать любые текстовые значения. Примеры значений: • `debug` — версия в разработке; • `experimental` — экспериментальная версия; • `test` — версия для тестирования; • `stable` — стабильная версия.	строка	нет
`date`	Дата создания или изменения глобальной функции. Задается в формате `YYYY-MM-DD`, то есть "год-месяц-день". Поле не связано с полями **Дата создания** и **Дата изменения** в карточке элемента экспертизы.	строка	нет
`type`	Тип объекта. Для глобальных функций всегда указывается `global_function`.	строка	да
`version`	Версия функции в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора функции.	строка	нет
`source`	Код глобальной функции, реализованный на языке VRL.	строка	да
`arguments`	Набор аргументов, определяющих входные данные функции. Каждый аргумент имеет тип и имя.	массив объектов	да
`fallible`	Определяет, может ли функция возвращать ошибки. По умолчанию `true`.	логический	нет
`tests`	Набор тестов для проверки корректности работы функции.	массив объектов	нет



Строки могут записываться в разных стилях YAML. Для каждого стиля действуют свои правила экранирования:

- **Без кавычек:** экранирование не требуется. Строки без кавычек не могут начинаться с символов решетки, кавычек, дефиса, пробела и других, указанных в спецификации.

```
path: \Software\App\'Users'\Settings
```

- **В одиночных кавычках:** одиночные кавычки экранируются удвоением.

```
path: '\Software\App\''Users''\Settings'
```

- **В двойных кавычках:** обратная косая черта и двойные кавычки экранируются обратной косой чертой.

```
path: "\\Software\\App\\'Users'\\Settings"
```

В блоках VRL используются правила экранирования строк VRL.

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform>
- <source>
- <eventid_1>
- <eventid_2>
```

Здесь:

- `<platform>` — платформа или информационная система.
- `<source>` — источник событий.
- `<eventid_N>` — идентификаторы событий (EventID) или пути к журналу событий.

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `arguments`

Поле	Описание	Тип данных	Обязательное поле
`keyword`	Имя аргумента, которое используется в коде функции.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`kind`	Тип аргумента. Допустимые значения: `bytes`, `integer`, `float`, `boolean`, `object`, `array`, `timestamp`, `regex`.	строка или объект	да

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его при проверке работы глобальной функции.	строка	да
`events`	Набор входных данных для тестирования в виде массива событий, передаваемых в функцию.	массив	да
`assertion`	Блок проверок на языке VRL, выполняемых на каждом событии из `events`. Включает `assert` и `assert_eq` для валидации ожидаемых результатов.	строка	да

Составление глобальной функции

Для создания глобальной функции необходимо определить ряд обязательных и опциональных полей, которые формируют ее поведение и логику обработки данных.

Поля метаданных

Заполните поля метаданных глобальной функции:

- `id`: уникальный идентификатор глобальной функции. Должен быть уникальным в пределах экспертизы. Можно использовать UUID в качестве значения.
- `name`: название глобальной функции. Допустимы латинские буквы, цифры и символы подчеркивания.
- `type`: всегда указывается `global_function` для элементов экспертизы данного типа.
- `description` (опционально): краткое описание назначения функции.
- `author` (опционально): имя и контактные данные автора глобальной функции.
- `version`: версия глобальной функции согласно стандарту Semantic Versioning.
- `status` (опционально): статус разработки текущей версии глобальной функции. Может принимать текстовые значения, такие как:
  - `debug`: версия в разработке.
  - `experimental`: экспериментальная версия.
  - `test`: версия для тестирования.
  - `stable`: стабильная версия.

- `date` (опционально): дата создания или изменения глобальной функции. Формат значения: `YYYY-MM-DD` (год-месяц-день). Поле не связано с полями **Дата создания** и **Дата изменения** в карточке элемента экспертизы.
- `tags` (опционально): список тегов, присвоенных глобальной функции для классификации и быстрого поиска в системе.

Пример метаданных:

```
id: a123b456-c789-0def-1112-000000000000
name: multiply_values
type: global_function
description: Функция для умножения двух чисел
author: Ivan Ivanov <ivanov@example.ru>
version: 1.0.0
status: stable
date: 2025-01-23
tags:
  - math
  - operations
```

В этом примере:

- Глобальная функция с уникальным идентификатором `a123b456-c789-0def-1112-000000000000` предназначена для выполнения операции умножения.
- Название функции `multiply_values` указывает на ее функциональность.
- Поле `type` имеет значение `global_function`, что обозначает тип элемента экспертизы.
- Версия функции указана как `1.0.0` в соответствии с правилами семантического версионирования.
- Поле `status` имеет значение `stable`, что указывает на завершение разработки функции.
- Поле `date` содержит дату последнего изменения или создания функции: `2025-01-23`.
- Автором функции является `Ivan Ivanov`, чьи контактные данные указаны в поле `author`.
- Теги `math` и `operations` присвоены функции для облегчения поиска и классификации в системе.

Поле `source`

Поле `source` содержит основной код глобальной функции, написанный на языке VRL. Этот код реализует логику выполнения функции и использует аргументы, указанные в поле `arguments`. Код функции должен быть валидирован, а возвращаемое значение должно соответствовать ожидаемому типу, заданному в конфигурации функции.

Основные аспекты работы `source`

- **Входные данные**: аргументы, передаваемые функции, доступны внутри `source` под именами, указанными в `arguments`.
- **Логика выполнения**: последовательность операций, выполняемых над входными данными, таких как вычисления, условные конструкции и вызовы других функций.

- **Возвращаемое значение**: итоговый результат выполнения логики, указанный в `source`. Это значение возвращается функцией и должно соответствовать требованиям типов данных, заданных в конфигурации функции.

Пример реализации `source`:

```
source: !vrl |
  # Умножение значений двух аргументов.
  argA * argB
```

В этом примере:

- `argA` и `argB`: аргументы, передаваемые в функцию.
- `*`: операция умножения, результат которой возвращается как итог выполнения функции.
- Возвращаемое значение: произведение значений `argA` и `argB`.

Поддерживаемые элементы в `source`

Поле `source` поддерживает все возможности языка VRL. Подробное описание доступных выражений и функций приведено в документации по выражениям и функциям.

Основные возможности включают:

- **Операции с аргументами**: работа с переданными данными, такими как арифметические операции, манипуляции со строками и преобразования типов.
- **Условные конструкции**: использование логических выражений и операторов `if`/`else` для ветвления логики выполнения.
- **Встроенные функции VRL**: вызов встроенных VRL-функций для обработки данных (например, преобразования, фильтрации, агрегации).
- **Работа с объектами**: доступ к полям сложных объектов, переданных в функцию, через их ключи.

Примеры значений в `source`

Простое арифметическое выражение

```
source: !vrl |
  argA * argB
```

Выполняется умножение двух числовых аргументов `argA` и `argB`.

Условное ветвление

```
source: !vrl |
  if arg > 10 {
    "Value is greater than 10"
  } else {
    "Value is 10 or less"
```

```
  }
```

Функция возвращает строку в зависимости от значения аргумента `arg`.

Использование встроенной функции

```
source: !vrl |
  to_string(arg)
```

Функция преобразует числовое значение аргумента `arg` в строку.

Работа с объектом

```
source: !vrl |
  slice(arg."field_name", 0, 5)
```

Функция извлекает подстроку из значения поля `field_name` объекта `arg`.

Поле `arguments`

Поле `arguments` задает параметры, передаваемые в глобальную функцию при ее вызове. Оно определяет структуру, тип данных и имена параметров, доступных внутри функции.

Элементы структуры `arguments`

- `keyword`: уникальное имя параметра, используемое в коде функции для ссылки на аргумент.
- `kind`: тип данных параметра.

Пример структуры `arguments`

```
arguments:
  - keyword: input_data
    kind: object
  - keyword: multiplier
    kind: integer
```

В этом примере:

- Аргумент `input_data` имеет тип `object`, что означает, что он представляет объект с произвольной структурой.
- Аргумент `multiplier` имеет тип `integer`, что указывает на ожидание целого числа.

Типы данных

Поле `kind` может принимать следующие значения типов данных:

- `bytes`: бинарные данные.
- `integer`: целые числа.

- `float`: числа с плавающей точкой.
- `boolean`: логические значения (`true` или `false`).
- `object`: объект (структура ключ-значение).
- `array`: массив.
- `timestamp`: временные метки в формате RFC 3339.
- `regex`: регулярное выражение.

Дополнительные требования

1. Имена параметров (`keyword`) должны быть уникальными в рамках одной функции.
2. Тип данных (`kind`) должен быть совместим с ожидаемыми значениями, передаваемыми при вызове функции.
3. Если аргументу передается объект, каждое поле в его структуре должно быть строго типизировано.

Поле `fallible`

Поле `fallible` в глобальной функции определяет, может ли функция возвращать ошибки.

По умолчанию в поле установлено значение `true` и функция может возвращать ошибки. При вызове такой глобальной функции нужно обработать ошибку времени выполнения путем одной из следующих операций на языке VRL:

- Присваивание
- Объединение
- Вызов функции

Если в поле установлено значение `false`, функция не может возвращать ошибок. При использовании такой функции не следует выполнять присваивание, объединение или вызов ошибки.



Глобальная функция может быть помечена как не возвращающая ошибок (`fallible: false`), только если в ее коде нет вызова функций с прерыванием `!`.

Тестирование глобальной функции



При обращении к глобальной функции в VRL-песочнице будет возвращено значение `null`.

Тестирование глобальной функции проверяет ее корректность и точность выполнения заданных вычислений или трансформаций на основе входных данных. Это позволяет убедиться, что функция возвращает ожидаемые результаты, корректно обрабатывает исключительные ситуации и ошибки, а также стабильно работает в продуктивной среде.

Если глобальная функция взаимодействует с внешними данными (например, активными списками или таблицами обогащения), рекомендуется настроить искусственные данные через поле `setup_tests`. Это позволяет моделировать работу функции в условиях, близких к продуктивным, без обращения к реальной инфраструктуре.

Поле `tests`

Поле `tests` определяет набор тестов для проверки работы функции. Каждый тест содержит описание входных данных, условий проверки, и ожидаемого результата.

Элементы структуры `tests`:

- `name`: название теста для его идентификации.
- `events`: список входных данных, которые подаются на вход функции в процессе тестирования. Входные данные состоят из следующих обязательных полей:
  - `$arguments` — список передаваемых аргументов.
  - `$event` — массив тестовых событий. Эти события моделируют данные, которые передаются функции при ее реальной работе.
- `assertion`: выражение или набор выражений на языке VRL, проверяющих корректность результата выполнения функции.

Пример структуры `tests`

```
tests:
  - name: Тест умножения чисел
    events:
      - { "$arguments": { "argA": 50, "argB": 10 }, "$event": {} }
    assertion: !vrl |
      assert_eq!(.[0], 50, "Ожидается произведение 50")
```

В этом примере:

- `name`: тест называется "Тест умножения чисел".
- `events`: входное событие включает два аргумента, `argA` и `argB`.
- `assertion`: проверяется, что вызов функции возвращает значение `50`.

Тестовые события

Массив `events` содержит данные, передаваемые в функцию в процессе тестирования. Каждый элемент массива представляет собой объект, где ключи соответствуют именам аргументов функции, а значения — типам данных, указанным в `arguments`.



Для работы с временными метками (например, типом `TIMESTAMP`) значения в событиях должны быть указаны в формате `t'<значение>'`. Например: `t'2024-05-15 06:28:15'`. Это гарантирует корректную обработку времени.

Условия проверки

Поле `assertion` задает выражения на языке VRL, которые проверяют результат работы функции. Условия могут включать проверки значений, использование логических операторов, или проверку обработки исключительных ситуаций.

Создание и выполнение тестов

Для проверки глобальной функции создается набор тестов, каждый из которых включает следующие элементы:

1. Описание теста (`name`): краткое описание цели и ожидаемого результата теста.
2. События (`events`): список входных данных, которые подаются на вход функции в процессе тестирования. Для входных данных необходимо заполнить следующие поля:
   - `$arguments` — список передаваемых аргументов.
   - `$event` — массив тестовых событий.
3. Утверждения (`assertions`): логика проверки результата работы функции на основе переданных событий и условий.

Пример:

```
tests:
  - name: Проверка функции деления
    events:
      - { $arguments: { dividend: 10, divisor: 2 }, $event: {}}
    assertion: !vrl |
        assert_eq!(.[0], 5, "Ожидается результат 5")
```

Логика проверки (`assertions`) позволяет убедиться, что возвращаемый функцией результат соответствует ожидаемому. Если утверждение выполняется, это подтверждает корректность работы функции. Если же результат отличается от ожидаемого, это может указывать на необходимость доработки функции.



- Перед запуском тестов убедитесь, что значения в `events` соответствуют формату и типам, ожидаемым функцией.
- Используйте результаты тестов для оптимизации логики и повышения устойчивости функции.

Поле `setup_tests`



Если в элементе экспертизы используются внешние зависимости, то есть активные списки, таблицы обогащения или глобальные функции, обязательно заполните поле `setup_tests`.

Поле `setup_tests` в конфигурации предназначено для определения тестового окружения, включая предварительную настройку искусственных данных для активных списков, таблиц обогащения и глобальных функций, используемых в элементе экспертизы. Это обеспечивает проверку правила на корректность выполнения и взаимодействие с внешними данными до его применения в операционной среде.

Поле `setup_tests` используется для:

- Имитации взаимодействия с активными списками, таблицами обогащения и глобальными

функциями.

- Проверки логики правила или глобальной функции с учетом ожидаемых данных обогащения.

Процедура настройки `setup_tests`:

1. В корне конфигурации укажите секцию `setup_tests`.
2. Внутри секции `setup_tests` через ключ `enrichment_tables`, `active_lists` или `global_functions` задайте искусственные таблицы обогащения, активные списки и глобальные функции, используемые правилом.
3. Для каждого искусственного элемента укажите его структуру и набор искусственных данных, которые будут использоваться при тестировании.



Искусственные данные должны соответствовать ожидаемым данным из реальных таблиц обогащения, активных списков или глобальных функций и позволять полноценно проверить работу элемента экспертизы.

Структура `setup_tests` для активного списка:

```
setup_tests:
  active_lists:
    <active_list_name>:
      <field>:
        - [<field_value_1>]
        - [<field_value_2>]
```

Здесь:

- `<active_list_name>` — имя активного списка.
- `<field>` — поле активного списка.
- `<field_value_N>` — значения поля активного списка.

Структура `setup_tests` для таблицы обогащения:

```
setup_tests:
  enrichment_tables:
    <enrichment_table_name>:
      fields:
        - <field_1>
        - <field_2>
      data:
        - [<field_1_value_1>, <field_2_value_1>]
        - [<field_1_value_2>, <field_2_value_2>]
```

Здесь:

- `<enrichment_table_name>` — имя таблицы обогащения.
- `<field_N>` — поля таблицы обогащения.

- `<field_N_value_M>` — значения полей таблицы обогащения.

Структура `setup_tests` для глобальной функции:

```
setup_tests:
  global_functions:
    <global_function_name>:
      arguments:
        - keyword: <parameter_1_name>
          kind: <parameter_1_type>
        - keyword: <parameter_2_name>
          kind: <parameter_2_type>
      values_map:
        - argument: <argument>
          event: <event>
          return: <result>
      default: <default_result>
      fallible: <bool_value>
```

Здесь:

- `<global_function_name>` — имя глобальной функции.
- `<parameter_N_name>` — имена параметров глобальной функции.
- `<parameter_N_type>` — типы параметров глобальной функции.
- `<argument>`, `<event>` — аргументы глобальной функции.
- `<result>` — результат глобальной функции.
- `<default_result>` — результат глобальной функции по умолчанию.
- `<bool_value>` — логическое значение.

При вызове искусственная глобальная функция возвращает значение, заданное в `values_map`, если найдена запись с совпадающими аргументами. Если подходящей записи нет, возвращается значение из поля `default`.

Пример глобальной функции

Пример глобальной функции "Multiplication Function", схема RObject

```
# Универсальный уникальный идентификатор функции.
id: d2f0c404-5cb3-11e7-907b-000000000000

# Название глобальной функции.
# Должно быть уникальным в пределах коллектора.
name: multiplication_example

# Тип элемента экспертизы. Всегда имеет значение `global_function` для глобальных
функций.
type: global_function
```

```
# Версия функции.
version: 1.0.0

# Краткое описание назначения глобальной функции.
description: Функция умножения двух чисел

# Имя и контактные данные автора глобальной функции.
author: Ivan Ivanov <ivanov@example.ru>

# Статус разработки функции.
# Возможные варианты: `debug`, `experimental`, `test`, `stable`.
status: stable

# Дата создания или последнего изменения функции, формат YYYY-MM-DD.
date: 2025-01-23

# Категории, по которым функции группируются.
tags:
  - math
  - example

# Исполняемый код функции на языке VRL.
source: !vrl |
  # Функция возвращает произведение двух аргументов.
  if is_integer(argA) && (is_integer(argB)) {
    argA * argB
  } else {
    0
  }

# Параметры функции.
arguments:
  # Имя первого аргумента.
  - keyword: argA
    # Тип первого аргумента.
    kind: integer
  # Имя второго аргумента.
  - keyword: argB
    # Тип второго аргумента.
    kind: integer

# Опциональное. Определяет, может ли функция возвращать ошибки. По умолчанию 'true'.
fallible: true

# Список тестов функции.
tests:
  # Наименование теста, уникальное для каждого теста.
  - name: Multiplication Test 1
    # Тестовые данные.
    events:
      - { $arguments: { argA: 10, argB: 2 }, $event: {}}
```

```
    # Проверка для тестовых данных.
    assertion: !vrl |
      assert_eq!(.[0], 20, "Ожидается произведение 20")

  # Наименование теста для проверки обработки ошибок.
  - name: Multiplication Test Invalid Data
    # Тестовые данные с некорректным форматом.
    events:
      - { $arguments: { argA: "invalid", argB: 2 }, $event: {}}
    # Проверка для тестовых данных.
    assertion: !vrl |
      assert_eq!(.[0], 0, "При некорректном вводе ожидается результат 0")
```

Интерпретация

Метаданные глобальной функции:

- `id`: уникальный идентификатор функции — `d2f0c404-5cb3-11e7-907b-000000000000`.
- `name`: название функции — `multiplication_example`.
- `type`: тип объекта экспертизы — всегда `global_function` для глобальных функций.
- `version`: версия функции — `1.0.0`.
- `description`: описание назначения функции — "Функция умножения двух чисел".
- `author`: автор функции — `Ivan Ivanov <ivanov@example.ru>`.
- `status`: статус разработки функции — `stable`, что означает завершенную разработку и готовность к использованию.
- `date`: дата создания или последнего изменения функции — `2025-01-23`.
- `tags`: категории функции, используемые для классификации — `math` и `example`.

Код функции:

- `source`: основной код функции, написанный на языке VRL, реализует умножение значений аргументов `argA` и `argB` при условии, что в оба аргумента переданы целые числа. В противном случае в качестве результата функции возвращается значение `0`.

Параметры функции:

- `arguments`: список параметров, передаваемых функции.
- `keyword`: имя аргумента. В данном случае это `argA` и `argB`.
- `kind`: тип данных аргумента. В данном случае это `integer` (целое число).

Тесты:

- `tests`: набор тестов для проверки правильности выполнения функции.
- Тест 1:
  - `name`: название теста — "Multiplication Test 1".
  - `events`: данные для теста содержат два значения: `argA = 10` и `argB = 2`.

- `assertion`: проверяется, что функция `multiplication_example` вернет `20` при переданных аргументах.
- Сообщение об ошибке, возвращаемое в случае несоответствия: `"Ожидается произведение 20"`.

- Тест 2:
  - `name`: название теста — "Multiplication Test Invalid Data".
  - `events`: данные содержат некорректное значение `argA = "invalid"` и корректное `argB = 2`.
  - `assertion`: проверяется, что функция обрабатывает невалидные данные и возвращает `0`, если вход некорректен.
  - Сообщение об ошибке, возвращаемое в случае несоответствия: "При некорректном вводе ожидается результат 0".

Тестирование правил экспертизы с помощью языка VRL

В данном разделе рассматриваются основные аспекты создания тестов, включая использование заглушек, проверку полноты тестов, охват всех возможных сценариев и проведение тестов на производительность.



Инструкции по созданию тестов доступны в руководствах по созданию RObject-конфигураций элементов экспертизы.

Заглушки



Заглушки — это специальные заменители, которые используются в тестах для изоляции конкретных участков кода. Они имитируют работу частей системы, которые не поддаются тестированию в данном контексте, например, внешние API или несуществующие компоненты.

В тестах на основе заглушек важно гарантировать, что они правильно настроены, чтобы их использование не привело к ложным срабатываниям и некорректным результатам. Например, если заглушка возвращает неправильные данные, это может исказить результат теста, и его проверка не будет корректной.



Заглушки не должны создавать ложных срабатываний или неправильных выводов в процессе выполнения теста.

Заглушки используются в следующих случаях:

- Изоляция сложных зависимостей: заглушки могут заменить сложные или зависимые компоненты, которые сложно или невозможно протестировать, как, например, внешние веб-сервисы.
- Ускорение тестов: заглушки могут значительно ускорить выполнение тестов, поскольку вместо реальных вызовов к базе данных, файловой системе или сети используется имитация их работы.
- Тестирование ошибок и исключений: заглушки могут быть настроены так, чтобы имитировать ошибки, что полезно для проверки обработки исключительных ситуаций.

Пример теста с заглушкой

```
...
tests:
  - name: Privilege escalation from admin to root
    events:
       - { "uid": "root", "suid": "admin", "cmd": "pytohn", "sproc":"python",
"dproduct": "Ubuntu", "dvchost": "ubnt.land", "dvendor": "Linux", "mode": "0100600",
"msg": [ "audit(1694421703.884:27001658):" ], "name": "", "params": [ "-m" ],
"sourceIp": "10.150.50.115", "suser": "1000", "syscall": "221", "duid": "0"}
    assertion: !vrl |
      true
```

В данном примере заглушка используется для тестирования случая повышения привилегий с пользователя admin до root. В контексте данного теста не происходит выполнения реальных операций с привилегиями или вызовов системы, а проверка сводится к простому утверждению `true`.

Полнота написания тестов

Тесты должны быть достаточно полными, чтобы гарантировать проверку всех ключевых параметров и состояний, особенно при работе с более сложными функциями или обработкой данных.

Ниже приведен пример неполного теста, который проверяет только одно условие (параметр `appname`):

Пример неполного теста

```
...
tests:
  - name: "Test"
    events:
      - { "raw": {"appname": "testapp", "facility": "kern", "message": "test message"}
}
    assertion: !vrl |
      assert_eq!(.[0].appname, "testapp")
```

Этот тест проверяет только значение поля `appname`, но не включает другие важные параметры, такие как `facility` и `message`. Это может привести к пропуску ошибок в данных.

Оптимизированный тест должен включать проверки для всех важных параметров, таких как `facility` и `message`:

Пример оптимизированного теста

```
...
tests:
  - name: "Test"
    events:
      - { "raw": {"appname": "testapp", "facility": "kern", "message": "test message"}
```

```
}
    assertion: !vrl |
      assert_eq!(.[0].appname, "testapp")
      assert_eq!(.[0].facility, "kern")
      assert_eq!(.[0].message, "test message")
```

Рассмотрим пример теста нормализации, где необходимо проверить несколько полей.

Пример неполного теста для проверки нормализации

```
...
normalizer: !vrl |
  .dproduct = "metrics"
  .dvendor = to_string(.raw.namespace) ?? ""
  .name = to_string(.raw.tags.component_id) ?? ""
  .cnt = to_string(.raw.counter.value) ?? ""
  .cat = to_string(.raw.name) ?? ""
tests:
  - name: add_resource1
    events:
     - { "raw":
{"appname":"testtag","facility":"kern","hostname":"host001","message":"13.03.2017
17:54.25 '# UserGate Server# ' 192.168.30.71 -> 188.40.238.250:80
0/440","procid":14884,"severity":"info","timestamp":"2024-02-27T16:14:41Z"}}
    assertion: !vrl |
      assert_eq!(.[0].bytesOut, null)
```

Для того чтобы исправить тест, нужно добавить проверку для каждого поля:

Пример оптимизированного теста

```
...
tests:
  - name: "testing"
    events:
      - raw:
{"counter":{"value":1751.0},"kind":"absolute","name":"component_received_event_bytes_t
otal","namespace":"vector","tags":{"component_id":"unique-id-
12345","component_kind":"transform","component_type":"remap","host":"collector-host-
01"},"timestamp":"2024-05-29T10:26:42.688089484Z"}
    assertion: !vrl |
      assert_eq!(.[0].cat, "component_received_event_bytes_total")
      assert_eq!(.[0].cnt, "1751")
      assert_eq!(.[0].dvendor, "vector")
      assert_eq!(.[0].name, "unique-id-12345")
      assert_eq!(.[0].dproduct, "metrics")
```

Проверка всех сценариев

Тесты должны обеспечивать надежность и предсказуемость правил, чтобы исключить нежелательные срабатывания или пропуски.

Тестирование всех сценариев выполнения правил включает проверку корректной обработки событий, обработки некорректных данных, обработки исключений и проверки работы на граничных условиях.

Тесты можно разделить на несколько категорий:

- **Позитивные сценарии**: проверка, что правило корректно обрабатывает события, соответствующие условиям.
- **Негативные сценарии**: проверка, что некорректные или неподходящие события не приводят к срабатыванию правила.
- **Граничные случаи**: тестирование минимальных, максимальных значений и редких комбинаций.



При описании тестов рекомендуется указывать цель проверки, чтобы упростить сопровождение кода.

Для достижения полной проверки сценариев следует:

- Убедиться, что тесты охватывают все логические ветки правил.
- Проверить реакции правил на отсутствующие, лишние или неверные данные.
- Включить тесты, моделирующие исключительные ситуации, например, пустые события, неожиданные форматы данных.

Пример теста

```
tests:
  - name: "Scenario 1: Basic network connection"
    description: Проверка корректного определения сетевого подключения.
    events:
      - { "type": "network", "action": "connect", "source_ip": "192.0.2.1" }
    assertion: !vrl |
      assert_eq!(.[0].type, "network") &&
      assert_eq!(.[0].action, "connect") &&
      assert_eq!(.[0].source_ip, "192.0.2.1")

  - name: "Eventid 7: wmic process list using wmiscript.xsl"
    description: Проверка обработки события с использованием `wmic` и `wmiscript.xsl`.
    events:
      - { "dvchost": "host1.domain.local", "fname":
"C:\\Windows\\System32\\wmiscript.dll" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].fname, "C:\\Windows\\System32\\wmiscript.dll")

  - name: "Eventid 7: msxsl processing msxscript.xsl"
    description: Проверка корректного распознавания использования `msxsl` и
```

```
`msxscript.xsl`.
    events:
      - { "dvchost": "host1.domain.local", "fname":
"C:\\Windows\\System32\\msxscript.dll" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].fname, "C:\\Windows\\System32\\msxscript.dll")

  - name: "Eventid 4688: msxsl loading xsl over network"
    description: Проверка загрузки `.xsl` через SMB-сетевой путь.
    events:
      - { "dvchost": "host1.domain.local", "fname": "\\x.x.x.x\\mysmbshare\\file1.xsl"
}
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].fname, "\\x.x.x.x\\mysmbshare\\file1.xsl")

  - name: "Eventid 1: wmic process execution"
    description: Проверка выполнения команды через `wmic`.
    events:
      - { "dvchost": "host1.domain.local", "fname": "C:\\Windows\\System32\\cmd.exe" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].fname, "C:\\Windows\\System32\\cmd.exe")

  - name: "Eventid 1: msxsl executing calc.exe"
    description: Проверка выполнения `calc.exe` через `msxsl`.
    events:
      - { "dvchost": "host1.domain.local", "sproc": "C:\\Windows\\System32\\cmd.exe" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].sproc, "C:\\Windows\\System32\\cmd.exe")

  - name: "Eventid 1: msxsl.exe from known hash"
    description: Проверка выполнения `msxsl.exe` из известного пути.
    events:
      - { "dvchost": "host1.domain.local", "dproc": "C:\\System\\Payloads\\msxsl.exe"
}
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].dproc, "C:\\System\\Payloads\\msxsl.exe")

  - name: "Eventid 1: msxsl.exe from random file"
    description: Проверка выполнения произвольного файла вместо ожидаемого.
    events:
      - { "dvchost": "host1.domain.local", "dproc": "C:\\System\\Payloads\\456.exe" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].dproc, "C:\\System\\Payloads\\456.exe")
```

Тесты на производительность

Производительность правил должна проверяться с помощью специализированных тестов, направленных на выявление узких мест в обработке большого объема событий. Это позволяет гарантировать, что система будет эффективно работать даже при высокой нагрузке и большом объеме данных.

Тесты на производительность обычно включают в себя создание набора событий, который имитирует реальные условия работы системы, и их многократную обработку для оценки времени выполнения правил. Для этого используется параметр `repetitions`, который задает количество повторений для каждого теста, что помогает оценить стабильность работы в условиях высоких нагрузок.

Пример теста на производительность

```
...
benches:
  - name: "Performance Test"
    repetitions: 10000
    events:
      - { "raw": "2022-08-11T12:32:41.000Z 5.86.210.12 - user=user2 act=file op=del
host=pc1 name=pomodoro.txt" }
...
```

В данном примере выполняется тест с 10 000 повторений, где для каждого события генерируется строка, содержащая информацию о файле и операции с ним. Это позволяет проверить, насколько эффективно правило обрабатывает множество таких событий в течение длительного времени.

Оценка покрытия техник MITRE ATT&CK правилами экспертизы

Данный раздел описывает процесс работы с оценкой покрытия MITRE ATT&CK в системе. Работа осуществляется в разделе **Инструменты** → **Покрытие MITRE ATT&CK** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О MITRE ATT&CK

MITRE ATT&CK — это база знаний, содержащая сведения о видах и подвидах приемов, которые могут использовать злоумышленники. Такие виды и подвиды называются **техниками** и **сабтехниками**.

Раздел **Инструменты** → **Покрытие MITRE ATT&CK** предназначен для оценки того, какая доля техник и сабтехник MITRE ATT&CK обнаруживается правилами корреляции. Из раздела можно также перейти к правилам для конкретной техники или сабтехники.

Интерфейс раздела

Панель инструментов раздела содержит следующие компоненты:

- Процент покрытия отображает долю техник и сабтехник MITRE ATT&CK, для которых в системе существуют правила корреляции.
- Переключатель **Только покрытые техники** позволяет отобразить только те техники и сабтехники, которые покрыты правилами корреляции.

В рабочей области раздела отображается список техник и сабтехник:

- Техники распределены по тактикам — этапам атаки. Список техник внутри тактики упорядочен по алфавиту.
- В поле каждой техники отображаются ее название и номер. Например, `Access Token Manipulation` — `T1134`.
- Если для техники выделены сабтехники, то в ее поле отображается кнопка  для раскрытия списка. Чтобы скрыть список сабтехник, нажмите на кнопку  .

  Список сабтехник внутри техники упорядочен по номерам.
- В поле каждой сабтехники отображаются ее название и номер. Например, `Create Process with Token` — `T1134.002`.

Состояние покрытия техник и сабтехник

Статус покрытия обозначается цветом в полях техник и сабтехник:

- голубой — сабтехники и техники, полностью покрытые правилами;
- желтый — техники, в которых покрыта только часть сабтехник.



Чтобы отобразить только те техники и сабтехники, для которых существуют правила, установите переключатель **Только покрытые техники** на панели инструментов в активное положение.

Переход к правилам для техники или сабтехники

Чтобы перейти к правилам, покрывающим конкретную технику или сабтехнику, наведите курсор на ее поле и нажмите на появившуюся кнопку  . На новой вкладке откроется раздел **Экспертиза** с фильтрацией по тегу техники или сабтехники.

Справочник по языку запросов RQL

Данный раздел содержит информацию о языке запросов R-Vision Query Language (далее также — RQL). RQL — это декларативный язык запросов, созданный для выполнения операций поиска, анализа и извлечения информации о событиях безопасности. Язык разработан для обработки данных в контексте информационной безопасности и адаптирован для работы с типами данных и моделями данных системы.

Работа с RQL-запросами

В данном разделе описано, как использовать язык запросов RQL для анализа событий и записей активных списков. Описано создание запросов, которые включают работу с проекциями, использование условий, управление выводом, сортировку и группировку результатов, а также заполнение данных. Также описывается использование фильтров и временных ограничений в поиске.



В данном разделе при описании выражений приняты следующие обозначения:

- Необязательные компоненты запросов заключаются в квадратные скобки.
- `<expression>` — выражение, то есть применение оператора, функция, идентификатор, литерал, арифметическое выражение или выражение в скобках.

Содержание раздела:

- Синтаксис запросов
- Работа с проекциями (оператор `SELECT`)
  - Обращение к вложенным данным в полях типа `JSON`
  - Использование псевдонимов (оператор `AS`)
- Использование условий (оператор `WHERE`)
  - Операторы сравнения
    - Использование операторов с полями `JSON`
  - Операторы работы с множествами (`IN`) и проверки на пустое значение (`IS NULL`)
    - Поиск по подсетям с оператором `IN`
  - Комбинирование условий (операторы `AND`, `OR`, `NOT`)
  - Работа с настраиваемыми полями универсальных моделей
  - Использование фильтров и ограничений по времени
- Группировка (оператор `GROUP BY`)
  - Фильтрация результатов группировки (оператор `HAVING`)
- Сортировка возвращаемых записей (оператор `ORDER BY`)
  - Заполнение пропусков в списке возвращаемых результатов (модификатор WITH FILL)
- Работа с активными списками (операторы `JOIN` и `ENRICH`)

    - Работа с оператором `JOIN` в режиме базового поиска
    - Работа с оператором `JOIN` в режиме продвинутого поиска
    - Работа с оператором `ENRICH`
- Преобразование значений с помощью условной логики (функция `if()`)
- Максимальное количество и пропуск возвращаемых записей (операторы `LIMIT` и `OFFSET`)

Синтаксис запросов

RQL-запросы используются для поиска и извлечения информации о событиях безопасности и записях активных списков.

Работа с событиями в R-Vision осуществляется только в рамках определенного хранилища. Поэтому, в отличие от SQL, в запросах не требуется указывать таблицу базы данных в компоненте `FROM <имя таблицы>`.

RQL-запросы используются в разделах **Дашборды** и **Поиск**. В разделе **Поиск** доступны два режима поиска — базовый и продвинутый, в последнем случае доступны использование проекций и группировка результатов.

В режиме базового поиска могут опускаться компонент `SELECT *` и оператор `WHERE` при указании условия:

```
[SELECT *] [<expression>]
[(INNER | LEFT) JOIN `active_list_name` [AS alias_name] ON <expression> [ENRICH
target_field = source_field]]
[ORDER BY field_name [ASC | DESC] [WITH FILL [FROM <expression> TO <expression>] STEP
<expression>]]
[LIMIT <m> [OFFSET <n>]]
```

```
SELECT <expression> [AS alias_name][, <expression>]
[(INNER | LEFT | RIGHT | FULL) JOIN `active_list_name` [AS alias_name] ON <expression>
[ENRICH target_field = source_field]]
[WHERE <expression>]
[GROUP BY field_name [HAVING <expression>]]
[ORDER BY field_name [ASC | DESC] [WITH FILL [FROM <expression> TO <expression>] STEP
<expression>]]
[LIMIT <m> [OFFSET <n>]]
```

Соответствие запросов SQL и RQL

Запрос в SQL	Запрос в системе
`SELECT tenantId FROM table`	`SELECT tenantId`

Запрос в SQL	Запрос в системе
`SELECT tenantId FROM table WHERE length(tenantId) = 5`	`SELECT tenantId WHERE length(tenantId) = 5`
`SELECT tenantId, count(*) FROM table GROUP BY tenantId`	`SELECT tenantId, count(*) GROUP BY tenantId`
`SELECT * FROM table WHERE id = 'abcdef'`	`id = 'abcdef'`

Фильтры в разделе **Поиск** и ограничения по времени применяются как дополнительные условия для запроса.



Если в RQL-запросе необходимо обратиться к полю, название которого совпадает с ключевым словом ClickHouse, например, `SELECT`, `FROM`, `WHERE`, то название поля необходимо заключать в обратные апострофы.

Пример запроса к полю с ключевым словом в названии

```
SELECT `FROM` FROM table
```

Работа с проекциями (оператор SELECT)

В системе для каждого хранилища событий выбирается модель событий, а для активного списка — схема активного списка, которые определяют структуру данных. По умолчанию система возвращает результат со **всеми** полям текущей модели данных или схемы активного списка независимо от того, какие поля отображаются в списке. Поэтому, если в результат необходимо включить все поля, в базовом поиске компонент запроса `SELECT *` может опускаться.

Оператор `SELECT` используется при работе с проекциями, то есть для указания определенных полей, которые будут включены в результат запроса. Этот оператор часто используется в виджетах и метриках, которые создаются на дашбордах. Имена полей в запросе отделяются друг от друга запятыми.

```
SELECT <expression>[, <expression>]
```



Не ставьте разделитель после последнего поля, включенного в запрос.

Оператор `SELECT` является обязательным при использовании оператора `WHERE` и группировке результатов.

Соответствие запросов SQL и RQL при использовании проекций

Запрос в SQL	Запрос в системе
`SELECT tenantId, raw FROM table`	`SELECT tenantId, raw`
`SELECT tenantId, raw FROM table WHERE length(tenantId) = 5`	`SELECT tenantId, raw WHERE length(tenantId) = 5`

Обращение к вложенным данным в полях типа JSON

В RQL-запросах доступно обращение ко вложенным объектам и массивам в полях типа JSON с помощью оператора $.



Поиск по полям внутри JSON приводит к снижению производительности, так как внутренние поля не индексируются.

Обращение к полям JSON

Чтобы обратиться к вложенному полю, используйте полный путь в JSON:

```
SELECT $<field_name>.<key_1>.<key_2>⋯.<key_N>
```

Здесь:

- <field_name> — ключ поля модели событий.
- <key_1>, …, <key_N> — ключи JSON. Ключи в пути перечисляются в порядке вложенности и разделяются точкой.



При обращении к первому уровню полей оператор $ можно опустить. Например, запросы data.referer вместо $data.referer.

Обращение к вложенным объектам JSON

Чтобы обратиться к вложенному объекту типа JSON, используйте оператор ^ перед путем:

```
SELECT $<field_name>.^<object_key>
```

Здесь:

- <field_name> — ключ поля модели событий.
- <object_key> — ключ объекта JSON.

Обращение к массивам JSON

Чтобы обратиться к плоскому массиву, где все элементы — простые значения, используйте путь в JSON:

```
SELECT $<field_name>.<arr_key>
```

Здесь:

- <arr_key> — ключ поля, которое является плоским массивом.

Чтобы обратиться к массиву со вложенными объектами, используйте оператор `[]`:

```
$<field_name>.<nested_arr_key>[]
```

Здесь:

- <nested_arr_key> — ключ поля-массива, которое содержит объекты.

Чтобы искать ключ во всех элементах массива, используйте следующий синтаксис:

```
SELECT $<field_name>.<arr_key>[].<nested_key>
```

Здесь:

- `<field_name>` — ключ поля модели событий;
- `<arr_key>` — ключ массива JSON;
- `<nested_key>` — ключ JSON, который будет найден во всех элементах массива `<key_2>`.

Обращение к массиву можно сочетать с обращением к вложенному объекту:

```
SELECT $<field_name>.<arr_key>[].^<object_key>
```



Обращение к отдельному элементу массива JSON недоступно.

Примеры обращения к полям `JSON`

Рассмотрим обращение к полям `JSON` на примере события, которое имеет поле `data` со следующим содержанием:

```
{
  "timestamp": "2025-08-15T15:07:89Z",
  "source": {
    "system": "firewall",
    "ip": "192.0.2.0"
   },
  "events": [
    {
      "event_id": "evt-001",
      "actor": {
```

```
        "user": "admin"
      }
    },
    {
      "event_id": "evt-002",
      "actor": {
        "process": "nginx"
      }
    },
    {
      "event_id": "evt-003",
      "actor": {
        "process": "powershell.exe"
      }
     }
  ],
  "params": [
    [
      "name",
      "value"
    ],
    [
      "result",
      "success"
    ]
  ]
}
```

Пример 1. Обращение к вложенному полю `JSON`

```
SELECT $data.source.ip
```

Этот запрос вернет значение поля `ip`, которое находится внутри `source`:

```
"192.0.2.0"
```

Пример 2. Обращение к вложенному объекту `JSON`

```
SELECT $data.^source
```

Этот запрос вернет объект в поле `source`:

```
{"system":"firewall","ip":"192.0.2.0"}
```

Пример 3. Обращение к массивам внутри массива

```
SELECT $data.params
```

Этот запрос вернет элементы массивов внутри массива `params`:

```
name,value,result,success
```

Пример 4. Поиск ключей внутри элементов массива

```
SELECT $data.events[].event_id
```

Этот запрос вернет массив значения полей `event_id` внутри элементов массива `events`:

```
evt-001,evt-002,evt-003
```

Пример 5. Поиск вложенных объектов внутри элементов массива

```
SELECT $data.events[].^actor
```

Этот запрос вернет массив значений полей `actor` внутри элементов массива `events`:

```
{"user": "admin""},{"process":"nginx"},"process":"powershell.exe"}
```

Использование псевдонимов (оператор `AS`)

При редактировании виджетов и метрик удобно использовать псевдонимы, которые позволяют сократить названия компонентов запроса в секции **Сопоставление полей**.

```
SELECT <expression> AS alias_name
```

Пример использования псевдонима для функции

При создании виджета типа **Таблица** вводится запрос:

Запрос с использованием псевдонима для функции `count()`

```
SELECT collectorId, count(*) AS cnt WHERE collectorId LIKE '%' GROUP BY collectorId
```

Здесь для функции `count(*)` задается псевдоним `cnt`, который можно указать в секции сопоставления полей.

Использование условий (оператор WHERE)

В базовом поиске при задании условия оператор WHERE в запросе опускается.

При использовании проекций оператор WHERE позволяет применять к указанному полю запрос с выборкой строк по определенному условию.

```
SELECT <expression> WHERE <expression>
```

В условиях используются операторы сравнения, работы с множествами и проверки на пустое значение. Для проверки полей типа Array используйте функцию has.

Пример выборки строк по условию

Запрос с оператором WHERE

```
SELECT tenantId WHERE length(tenantId) = 5
```

Этот запрос возвращает события с количеством символов в поле tenantId, равным 5.

Операторы сравнения

Оператор	Значение	Типы данных
=	Равно	Все типы
!=	Неравно	Все типы
<	Меньше	Числовые типы и даты
<=	Меньше или равно	Числовые типы и даты
>	Больше	Числовые типы и даты
>=	Больше или равно	Числовые типы и даты
LIKE	Соответствует заданному шаблону.	Строковые типы
NOT LIKE	Не соответствует заданному шаблону.	Строковые типы
BETWEEN	Входит в диапазон. a BETWEEN b AND c равнозначно выражению a >= b AND a <= c.	Числовые типы и даты
NOT BETWEEN	Не входит в диапазон. a NOT BETWEEN b AND c равнозначно выражению a < b OR a > c.	Числовые типы и даты



Шаблон может включать обычные символы и следующие метасимволы:

- % — обозначает любое количество произвольных символов или отсутствие символов.

- `_` — обозначает один произвольный символ.
- `\` — используется для экранирования символов `%`, `_` и `\`.

Использование операторов с полями `JSON`

Чтобы обратиться в условии к вложенным полям `JSON`, используйте соответствующий синтаксис:

Запрос с оператором `WHERE` и вложенным полем `JSON`

```
SELECT $data WHERE $data.source.ip = "192.0.2.0"
```

Для приведения типа используйте соответствующую функцию преобразования типов:

Запрос с оператором `WHERE` и вложенным полем `JSON`

```
SELECT $data WHERE toUInt64($data.process_id) = 2776
```

Например, чтобы выполнить поиск по полю `JSON` с оператором `LIKE`, сначала приведите его к строке с помощью функции `toString`:

Запрос с оператором `LIKE` и приведением к строке

```
SELECT * WHERE toString(data) LIKE '192.0.2.0%'
```

Данный запрос вернет записи, где в поле `data` встречается последовательность `192.0.2.0`.

Запрос с оператором `LIKE` и приведением к строке

```
SELECT * WHERE toString(data) like '%{"active":true,"ip":"192.0.2.0"}%'
```

Данный запрос вернет записи, где в поле `data` содержатся вложенные поля в указанном порядке:

```
{
  "active": true,
  "ip": "192.0.2.0"
}
```

Операторы работы с множествами (`IN`) и проверки на пустое значение (`IS NULL`)

Оператор	Значение
`IN`	Входит во множество
`NOT IN`	Не входит во множество
`IS NULL`	Значение является пустым (NULL)
`IS NOT NULL`	Значение не является пустым (NULL)

Поиск по подсетям с оператором IN

С помощью оператора IN можно осуществлять поиск данных по подсетям. Формат записи подсети: <netmask>/<prefix>. Здесь:

- <subnet> — маска подсети.
- <prefix> — сетевой префикс. Допустимые значения зависят от версии IP-протокола:
  - для IPv4: 0—32;
  - для IPv6: 0—128.

Поиск по IPv4-подсетям с оператором IN

```
sourceIp IN ':192.0.2.0/24'
```

Поиск по IPv6-подсетям с оператором IN

```
sourceIp IN '::ffff:192.0.2.0/64'
```



Поиск по подсетям также можно осуществлять с помощью функции isIPAddressInRange.

Проверка наличия элемента в массиве (функция has)

Чтобы проверить наличие элемента в поле с типом Array, используйте функцию has:

```
SELECT * WHERE has(field_name, value)
```

Здесь value — значение, которое нужно найти среди элементов массива. Может использоваться значение NULL.

Пример

Использование оператора has

```
SELECT * WHERE has(array_field1, "securityEvent")
```

Этот запрос возвращает события, где поле array_field1 содержит элемент "securityEvent".

Комбинирование условий (операторы AND, OR, NOT)

Совмещение нескольких условий происходит с использованием операторов:

Оператор	Значение
AND	Конъюнкция ("И")
OR	Дизъюнкция ("ИЛИ")
NOT	Логическое отрицание

Пример запроса с комбинированным условием

Рассмотрим запрос, возвращающий события, у которых значения в поле `tenantId` начинаются с символа n, а количество символов в поле `raw` превышает 10.

Запрос в SQL

```
SELECT * FROM table WHERE tenantId LIKE 'n%' AND length(raw) > 10
```

Запрос в системе

```
tenantId LIKE 'n%' AND length(raw) > 10
```

Работа с настраиваемыми полями универсальных моделей

В RQL-запросах доступно обращение к регулярным настраиваемым полям универсальной модели события. Обращение к таким полям может осуществляться как напрямую, так и посредством специального синтаксиса.



Обращение посредством специального синтаксиса доступно, если выполняются следующие условия:

- Имя поля начинается с одного из следующих префиксов:
  - `cn`;
  - `cs`;
  - `c6a`;
  - `deviceCustomDate`.
- Существует парное поле, имя которого содержит `Label`.

Синтаксис

```
<custom_field>:<custom_field_label> <expression>
```

Здесь:

- `<custom_field>` — ключ значения настраиваемого поля, например: `cn1`, `deviceCustomDate1`.

  Вместо ключа конкретного поля можно использовать его семейство, зависящее от типа данных значения в этом поле:

  - `cs` — `LCString`;
  - `cn` — `UInt64`;
  - `c6a` — `IPv6`;
  - `deviceCustomDate` — `DateTime64`.



Семейство полей рекомендуется использовать в случаях, когда требуется

проверять условие <expression> по всем полям этого семейства.

- <custom_field_label> — поле метаданных, хранящее метку, которая указывает на содержание и назначение связанного с ним поля <custom_field>. Эта метка играет роль названия для поля <custom_field> и позволяет интерпретировать хранящиеся в нем данные.

  Примеры меток:

  - cn1Label — для поля cn1;
  - deviceCustomDate1Label — для поля deviceCustomDate1;

- <expression> — выражение с условием поиска по полю значения, использующее операции сравнения (=, !=, <>, <, >, <=, >=) и/или операторы IN, LIKE, NOT LIKE.



- Обращение к полю напрямую рекомендуется применять, если достоверно известно, какая информация хранится в этом поле, и требуется получить информацию строго из этого поля.
- Обращение к полям посредством специального синтаксиса рекомендуется применять в случаях, когда оно позволяет упростить RQL-запрос. Например:
  - Когда одно и то же поле универсальной модели содержит различные данные для разных событий.
  - Когда искомые данные распределены по различным полям универсальной модели нескольких событий.

  В этих случаях представленный синтаксис позволяет заменить множественные условия с операторами AND и OR коротким RQL-запросом.

Примеры обращения к настраиваемым полям

Пример обращения к полю напрямую

Допустим, имеется поток событий, у которых в поле cs1 хранится адрес электронной почты пользователя, совершившего подозрительное действие. Известно, что часть событий является тестовой и приходит от разных пользователей с одним и тем же электронным адресом test@example.com.

Требуется исключить из рассматриваемого потока тестовые события. Ниже представлен вариант соответствующего RQL-запроса с непосредственным обращением к полю cs1.

Обращение к настраиваемому полю напрямую

```
cs1 != "test@example.com"
```

Запрос выдаст события, поле cs1 которых не содержит значение test@example.com.

Пример обращения к полю по его ключу

Допустим, имеется поток событий от разных источников. Для событий от некоторых источников в поле cs1 хранится адрес электронной почты пользователя, совершившего подозрительное действие. В соответствующем поле метки cs1Label этих событий хранится значение email. Известно, что часть

событий от этих источников является тестовой и приходит от разных пользователей с одним и тем же электронным адресом `test@example.com`.

Требуется получить поток событий, для которых указаны электронные адреса пользователей, и исключить из него тестовые события. Ниже представлен RQL-запрос, построенный на основе представленного ранее синтаксиса. Обращение к полю `cs1` выполняется по его ключу.

Обращение к настраиваемому полю по его ключу

```
cs1:email != "test@example.com"
```

Запрос выдаст события, для которых одновременно выполняются следующие условия:

1. Поле `cs1` не содержит значение `test@example.com`.
2. Поле `cs1Label` содержит название `email`.

Представленный запрос эквивалентен следующему запросу:

Обращение к настраиваемому полю и его полю метки

```
cs1 != "test@example.com" AND cs1Label = "email"
```

Пример обращения к полю по его семейству

Допустим, имеется поток событий из разных источников. Для событий некоторых источников в полях семейства `cs` (например, `cs1`, `cs2`, `cs3`) хранится адрес электронной почты пользователя, совершившего подозрительное действие. В соответствующих полях меток (например, `cs1Label`, `cs2Label`, `cs3Label`) этих событий хранится значение `email`. Известно, что часть событий от этих источников является тестовой и приходит от разных пользователей с одним и тем же электронным адресом `test@example.com`.

Требуется получить поток событий, для которых указаны электронные адреса пользователей, и исключить из него тестовые события. Ниже представлен RQL-запрос, построенный на основе представленного ранее синтаксиса. Обращение к настраиваемым полям универсальной модели события выполняется по их семейству `cs`.

Обращение к настраиваемым полям по их семейству

```
cs:email != "test@example.com"
```

Запрос выдаст события, для которых одновременно выполняются следующие условия:

1. Ни одно из полей значений семейства `cs` не содержит значение `test@example.com`.
2. Поле метки, соответствующее полю значений семейства `cs`, содержит название `email`.

Представленный запрос эквивалентен следующему запросу:

Обращение к настраиваемым полям и их полям меток

```
(cs1 != "test@example.com" AND cs1Label = "email") OR (cs2 != "test@example.com" AND
```

```
cs2Label = "email") OR (cs3 != "test@example.com" AND cs3Label = "email") OR (cs4 !=
"test@example.com" AND cs4Label = "email") OR (cs5 != "test@example.com" AND cs5Label
= "email") OR (cs6 != "test@example.com" AND cs6Label = "email")
```

Использование фильтров и ограничений по времени

После выполнения запроса в разделе **Поиск** можно отфильтровать возвращенные результаты с помощью кнопки **Добавить фильтр**. В окне добавления фильтра указывается поле, по которому проводится фильтрация, оператор сравнения и значение для сравнения.

Вместе с фильтрами удобно использовать функциональность ограничения запроса по времени. Это ограничение аналогично добавлению к запросу компонента, который ограничивает результаты запроса определенным временным интервалом.

Система воспринимает применяемые фильтры и ограничения по времени как дополнительные условия оператора `WHERE`, присоединяемые оператором `AND`. Переключатель **Инвертировать (NOT)** интерпретируется как оператор `NOT`.

Примеры использования фильтров и ограничения по времени

Пример 1

Вы ввели в строке поиска запрос `tenantId LIKE 'n%' AND length(raw) > 10`. Затем добавили фильтр по полю `originalTimestamp` с оператором `=` и значением `05.05.20236 05:00:00`. Последовательность этих действий равнозначна следующему запросу в строке поиска:

```
tenantId LIKE 'n%' AND length(raw) > 10 AND originalTimestamp = '2023.05.05 05:00:00'
```

Пример 2

Вы хотите ограничить результат запроса `tenantId LIKE 'n%' AND length(raw) > 10` датами `01.01.2023` и `12.12.2023`. Для этого после выполнения запроса выберите значение **Задать период** в поле периода и укажите диапазон периода в полях справа. Эти действия аналогичны следующему запросу:

```
tenantId LIKE 'n%' AND length(raw) > 10 AND (timestamp >= 2023.01.01 AND timestamp <=
2023.12.12)
```

Таким образом, применение фильтров и ограничений по времени существенно упрощает создание поисковых запросов в системе.

Группировка (оператор `GROUP BY`)

Объединять результаты запроса по одному или нескольким полям можно с помощью оператора `GROUP BY`. С оператором `GROUP BY` могут применяться агрегатные функции: `count()`, `sum()`, `avg()`, `max()`, `min()`.

```
SELECT <expression> GROUP BY field_name
```



При использовании оператора GROUP BY необходимо указать, к какой проекции будет применяться объединение. Например, запрос вида `GROUP BY tenantID` воспринимается системой как `SELECT * GROUP BY tenantID`, и поэтому некорректен. Корректный запрос с указанной проекцией имеет вид `SELECT tenantID GROUP BY tenantID`.

Фильтрация результатов группировки (оператор `HAVING`)

Если требуется отфильтровать группы данных, полученные после объединения результатов, используйте оператор `HAVING`.

```
SELECT <expression> GROUP BY field_name HAVING <expression>
```

Пример использования оператора HAVING

```
SELECT tenantID, count(*) AS cnt GROUP BY tenantID HAVING cnt > 10
```

Этот запрос выбирает события, группируя их по `tenantID`, и возвращает только те группы, в которых количество событий больше 10.

Сортировка возвращаемых записей (оператор `ORDER BY`)

Результаты запроса можно сортировать по любому полю с помощью оператора `ORDER BY`.

```
[SELECT <expression>] ORDER BY field_name [ASC | DESC]
```

Направление сортировки задается модификаторами:

- `ASC` — по возрастанию;
- `DESC` — по убыванию.

Примеры сортировки

Сортировка в базовом поиске

```
ORDER BY collectorId
```

Этот запрос выводит события с сортировкой по полю `collectorId` в порядке возрастания.

Использование сортировки с агрегатной функцией `count()`

```
SELECT tenantId, count(*) ORDER BY tenantId ASC
```

Этот запрос возвращает события, упорядоченные по возрастанию значения в поле `tenantId` и указанием количества событий в каждой группе.

Заполнение пропусков в списке возвращаемых результатов (модификатор WITH FILL)

Чтобы заполнить пропуски в списке возвращаемых результатов при сортировке и группировке записей, используйте модификатор `WITH FILL`. Пропуски заполняются значениями с указанным шагом в указанном промежутке, либо по умолчанию.



`WITH FILL` можно применять только после имени столбца в `ORDER BY`. Допускается использование нескольких `WITH FILL` для разных столбцов.

```
SELECT <expression> ORDER BY field_name WITH FILL [FROM <expression> TO <expression>]
STEP <expression>]]
```



Для случая с несколькими полями `ORDER BY field2 WITH FILL, field1 WITH FILL` порядок заполнения будет соответствовать порядку полей в секции `ORDER BY`.

Здесь:

- `field_name` — имя поля, по которому происходит сортировка.
- `FROM <expression>` — начальное значение интервала, который будет заполнен. Если не указано, будет использовано наименьшее значение в поле `field_name`.
- `TO <expression>` — конечное значение. Если не указано, будет использовано наибольшее значение в поле `field_name`.
- `STEP <expression>` — шаг, с которым будет заполняться интервал. Если не указан, используется значение `1.0` для числовых типов, и 1 секунда для метки времени.

Значение параметра `<expression>` должен принадлежать к одному из следующих типов:

- числовой литерал (целое или дробное число);
- метка времени;
- арифметическая операция с числами, меткой времени, интервалом или функцией;
- функция с возвращаемым значением в виде числа или метки времени.

Примеры заполнения пропусков

Заполнение пропусков для временных интервалов

```
SELECT toStartOfHour(timestamp) as ts, sum(cnt) as sum_cnt GROUP BY ts HAVING sum_cnt
> 0 ORDER BY ts WITH FILL FROM now() - INTERVAL 1 day TO now() STEP INTERVAL 1 hour
```

Данный запрос выполняет следующую последовательность операций:

- `SELECT toStartOfHour(timestamp) as ts, sum(cnt) as sum_cnt`: выбираются значения поля `timestamp`, округленные до начала каждого часа, и вычисляется сумма поля `cnt` для каждой

группы записей с одинаковым значением `ts`.

- `GROUP BY ts`: результаты группируются по полю `ts`, которое представляет округленные значения.
- `HAVING sum_cnt > 0`: применяется фильтрация, чтобы оставить только те группы записей, у которых сумма поля `cnt` больше нуля. Это означает, что в результирующем наборе будут только записи, для которых существуют записи с положительными значениями `cnt` внутри группы.
- `ORDER BY ts WITH FILL FROM now() - INTERVAL 1 day TO now() STEP INTERVAL 1 hour`: результаты сортируются по полю `ts`. Здесь используется модификатор `WITH FILL`, который обеспечивает заполнение пропущенных интервалов с шагом в 1 час от текущего времени минус 1 день до текущего времени. То есть, если в исходных данных отсутствуют записи для определенных интервалов времени, то они будут включены в результирующий набор со значениями по умолчанию.

Заполнение пропусков для временных интервалов с фильтрацией

```
SELECT toStartOfHour(timestamp) as ts, sum(cnt) as sum_cnt GROUP BY ts HAVING sum_cnt
> 0 AND ts >= now() - INTERVAL 1 day AND ts <= now() ORDER BY ts WITH FILL FROM now()
- INTERVAL 1 day TO now() STEP INTERVAL 1 hour
```

Здесь выполняются те же самые операции, что и в первом запросе, но добавляется дополнительный фильтр в выражении `HAVING` и выражении `AND`. Этот фильтр ограничивает результаты только записями, у которых поле `ts` находится в интервале от текущего времени минус 1 день до текущего времени. Таким образом, данный запрос дополнительно фильтрует записи по временному интервалу перед применением сортировки и заполнением пропущенных интервалов.

В результате выполнения этого запроса будет получен отсортированный список, где каждая запись будет содержать округленное значение `timestamp`, сумму `cnt` для соответствующей группы записей, а также пропущенные интервалы времени со значениями по умолчанию, если таковые имеются. В результирующем наборе будут только записи, удовлетворяющие условиям фильтрации по сумме `cnt` и временному интервалу.

Работа с активными списками (операторы `JOIN` и `ENRICH`)

Данные можно обогащать и фильтровать через активные списки с помощью операторов `JOIN` и `ENRICH`:

- В режиме базового поиска для фильтрации записей по данным активных списков используется оператор `JOIN`, который позволяет уточнить результаты запросов на основе совпадения данных без их обогащения. Оператор `ENRICH` используется в комбинации с `JOIN` и позволяет обогащать эти результаты данными из активных списков. Поддерживаются операции `INNER JOIN` и `LEFT JOIN`.
- В режиме продвинутого поиска для объединения данных из активных списков с событиями из хранилища с целью обогащения информации используется оператор `JOIN`. Поддерживаются операции `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN` и `FULL JOIN`.

Работа с оператором `JOIN` в режиме базового поиска

В режиме базового поиска раздела **Поиск** оператор `JOIN` служит для фильтрации записей на основе данных из активных списков. Это позволяет детализировать результаты запросов путем анализа

совпадений значений полей без обогащения событий. При этом данные из хранилища событий и активные списки рассматриваются как отдельные таблицы: хранилище событий является "левой таблицей", а активный список — "правой".

```
(INNER | LEFT) JOIN `<active_list_name>` [AS alias_name] ON <expression>
```

Здесь:

- `(INNER | LEFT)` указывает на тип операции `JOIN`:
  - `INNER JOIN` отбирает строки при наличии совпадения данных между обеими таблицами.
  - `LEFT JOIN` отображает все строки из левой таблицы (хранилище событий), дополняя их совпадающими данными из правой таблицы (активный список), если таковые имеются. Применяется совместно с оператором `ENRICH`.
- `<active_list_name>` обозначает имя активного списка, который используется в качестве правой таблицы для операции `JOIN`.

  Имя активного списка необходимо заключать в обратные апострофы.

- `AS alias_name` предоставляет возможность задать псевдоним для активного списка, упрощая дальнейшее обращение к данным в запросе.
- `<expression>` формулирует условие объединения, используя операции сравнения и логические операторы для определения совпадений данных между таблицами.

  При обращении к полям таблицы без указания ее названия система автоматически начинает поиск поля в хранилище событий. Если поле в хранилище событий не найдено, поиск поля продолжается в активном списке.

  Обращение к полю в записях событий может выполняться одним из следующих способов:

  - **Указание только названия поля**. Рекомендуется использовать данный способ, так как поиск полей по умолчанию начинается в "левой" таблице (хранилище событий).
  - **Указание имени таблицы из БД ClickHouse и названия поля**. Имя таблицы ClickHouse, связанной с хранилищем событий, необходимо заключать в обратные апострофы, например: `` `clickhouse_table_name`.field ``.

    Использование имени хранилища событий вместо имени ассоциированной с ним таблицы ClickHouse не допускается.

  Обращение к полю активного списка может выполняться одним из следующих способов:

  - **Указание только названия поля**. Данный способ допустим только в тех случаях, когда в записях событий отсутствует поле с таким же названием. В противном случае произойдет обращение к полю в записях событий.
  - **Указание имени активного списка и названия поля**. Имя активного списка

необходимо заключать в обратные апострофы, например: `` `active_list_name`.field ``.

- **Указание псевдонима активного списка и названия поля**. Псевдоним (alias) активного списка допускается задавать как с обратными апострофами, так и без них, например: `` `alias_name`.field `` или `alias_name.field`.

Пример использования `INNER JOIN` для фильтрации событий

Оператор `INNER JOIN` в разделе **Поиск** может быть использован для фильтрации записей, позволяя ограничить выборку событий на основе соответствия условиям, заданным для данных из активных списков. Этот механизм не обогащает данные событий новой информацией, но позволяет исключить из результата те события, которые не соответствуют заданным критериям.

Пример запроса ниже демонстрирует использование `INNER JOIN` для фильтрации событий по списку угроз `threat_list`, исключая из результатов все события, не соответствующие IP-адресам, занесенным в этот список. Цель запроса — идентифицировать потенциально вредоносную активность, происходящую с IP-адресов, указанных в списке угроз.

```
INNER JOIN `threat_list` ON sourceIp = `threat_list`.ip_address
```

Этот запрос исключит из результата все события, источник которых не совпадает с IP-адресами, указанными в активном списке `threat_list`. Таким образом, анализ будет сосредоточен исключительно на событиях, источник которых был заранее классифицирован как потенциальная угроза. Поскольку объединение данных не происходит, в результатах запроса отображаются только поля исходных событий, соответствующих условиям фильтрации.

Работа с оператором `JOIN` в режиме продвинутого поиска

В режиме продвинутого поиска раздела **Поиск** оператор `JOIN` позволяет объединять данные из активных списков и хранилища событий, обогащая информацию о событиях. Поддерживаются операции `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN` и `FULL JOIN`, каждая из которых имеет свои особенности при объединении данных. При этом события хранилища и записи активных списков анализируются как данные в двух разных таблицах: хранилище событий служит "левой таблицей", а активный список — "правой".

```
[SELECT <expression>] (INNER | LEFT | RIGHT | FULL) JOIN `<active_list_name>` [AS
alias_name] ON <expression>
```

Здесь:

- `(INNER | LEFT | RIGHT | FULL)` определяет тип операции `JOIN`, применяемой для объединения данных из левой таблицы (хранилища событий) и правой (активного списка):
  - `INNER JOIN` выбирает строки, где присутствует совпадение по заданным условиям в обеих таблицах.
  - `LEFT JOIN` отображает все строки из левой таблицы, дополняя их данными из правой таблицы при наличии совпадений.

- `RIGHT JOIN` аналогичен `LEFT JOIN`, но выводит все строки из правой таблицы, дополняя их данными из левой таблицы при наличии совпадений.
- `FULL JOIN` объединяет методы `LEFT JOIN` и `RIGHT JOIN`, показывая все строки из обеих таблиц.

- `<active_list_name>` обозначает имя активного списка, который используется в качестве правой таблицы.



Имя активного списка необходимо заключать в обратные апострофы.

- `[AS alias_name]` позволяет назначить псевдоним активному списку, упрощая дальнейшую работу с запросом.
- `<expression>` обозначает условие для объединения, которое может включать операции сравнения и логические операторы, для определения критериев совпадения данных между таблицами.

При обращении к полям таблицы без указания ее названия система автоматически начинает поиск поля в хранилище событий. Если поле в хранилище событий не найдено, поиск поля продолжается в активном списке. Обращение к полю в записях событий может выполняться одним из следующих способов:

- **Указание только названия поля**. Рекомендуется использовать данный способ, так как поиск полей по умолчанию начинается в "левой" таблице (хранилище событий).
- **Указание имени таблицы из БД ClickHouse и названия поля**. Имя таблицы ClickHouse, связанной с хранилищем событий, необходимо заключать в обратные апострофы, например: `` `clickhouse_table_name`.field ``.



Использование имени хранилища событий вместо имени ассоциированной с ним таблицы ClickHouse не допускается.

Обращение к полю активного списка может выполняться одним из следующих способов:

- **Указание только названия поля**. Данный способ допустим только в тех случаях, когда в записях событий отсутствует поле с таким же названием. В противном случае произойдет обращение к полю в записях событий.
- **Указание имени активного списка и названия поля**. Имя активного списка необходимо заключать в обратные апострофы, например: `` `active_list_name`.field ``.
- **Указание псевдонима активного списка и названия поля**. Псевдоним (alias) активного списка допускается задавать как с обратными апострофами, так и без них, например: `` `alias_name`.field `` или `alias_name.field`.

Примеры использования `JOIN` в продвинутом поиске

Пример 1. Использование `INNER JOIN`

Оператор `INNER JOIN` может быть использован для объединения данных о событиях с активными

списками, что позволит идентифицировать события, исходящие с IP-адресов, внесенных в список блокировки.

Пример запроса ниже демонстрирует использование `INNER JOIN` для соединения данных событий с активным списком `blacklist_ips` по IP-адресу источника (`sourceIp`), с целью выявления событий с блокированными IP-адресами.

```
SELECT sourceIp, timestamp, action, ip_address
INNER JOIN `blacklist_ips` ON sourceIp = `blacklist_ips`.ip_address
```

Визуализация ожидаемого результата запроса:

event.sourceIp	event.timestamp	event.action	blacklist_ips.ip_address
192.0.2.1	2022-07-01T12:00:00Z	Blocked	192.0.2.1
192.0.2.2	2022-07-02T12:00:00Z	Blocked	192.0.2.2

В этой таблице отображаются события, произошедшие с IP-адресов, указанных в системе (`event`), и соответствующие IP-адресам в активном списке `blacklist_ips`. Используя `INNER JOIN`, выборка ограничивается событиями, источники которых находятся в списке блокировки.

Пример 2. Использование `LEFT JOIN`

Оператор `LEFT JOIN` может быть использован для объединения данных о событиях входа в систему с информацией о пользователях из активного списка. Это позволяет получить расширенный контекст активности пользователей, включая отсутствие событий для определенных пользователей. В случаях, когда соответствующая запись в активном списке отсутствует, результатом объединения для данной записи события будут значения `NULL` в полях, предназначенных для данных из активного списка.

Пример запроса ниже демонстрирует использование `LEFT JOIN` для соединения событий с записями активного списка о пользователях `user_info` по идентификатору пользователя. Цель — включить в анализ всех пользователей, включая тех, по которым отсутствуют данные о событиях входа.

```
SELECT sourceIp, timestamp, ip_address, userName
LEFT JOIN `user_info` ON userId = `user_info`.user_id
```

Визуализация ожидаемого результата запроса:

event.sourceIp	event.timestamp	user_info.ip_address	user_info.userName
192.0.2.1	2022-07-01T12:00:00Z	192.0.2.1	Sergey
192.0.2.2	2022-07-02T12:00:00Z	192.0.2.2	Ivan
192.0.2.3	2022-07-03T12:00:00Z	NULL	NULL

В этой таблице показаны все попытки входа пользователей, зафиксированные в системе (`event`), дополненные информацией о них из активного списка `user_info`. Последняя строка демонстрирует случай, когда в таблице событий имеются данные о входе с IP-адреса 192.0.2.3, но информация о

пользователе, выполнившем вход, отсутствует в активном списке `user_info`.

Пример 3. Использование `RIGHT JOIN`

Оператор `RIGHT JOIN` может быть применен для объединения информации об устройствах из активного списка с данными о событиях, позволяя анализировать данные обо всех устройствах независимо от наличия связанных с ними событий. В случаях, когда соответствующая запись в хранилище отсутствует, результатом объединения для данной записи события будут значения `NULL` в полях, предназначенных для данных из хранилища.

Пример запроса ниже демонстрирует использование `RIGHT JOIN` для соединения информации об устройствах из активного списка `device_info` с данными о событиях на основе совпадения серийных номеров устройств (`serial_number`) с внешними идентификаторами устройств в событиях (`deviceExternalId`). Цель — предоставить полный обзор по всем устройствам, включая те, по которым отсутствуют события.

```
RIGHT JOIN `device_info` ON deviceExternalId = `device_info`.serial_number
```

Визуализация ожидаемого результата запроса:

event.deviceExternalId	event.action	device_info.serial_number	device_info.device_name
12345	Alert	12345	DeviceA
NULL	NULL	67890	DeviceB

В этой таблице отображается информация об устройствах из активного списка `device_info`, дополненная данными о событиях из системы. Запись устройства `DeviceB` демонстрирует случай, когда для устройства нет зарегистрированных событий.

Пример 4. Использование `FULL JOIN`

Оператор `FULL JOIN` можно использовать для комплексного анализа данных, чтобы объединить информацию о событиях с данными из активных списков, не исключая записи без взаимных соответствий. Это обеспечивает возможность анализа, включая изучение как связанных, так и независимых данных событий и элементов активных списков.

Пример запроса ниже иллюстрирует использование `FULL JOIN` для объединения данных о событиях с информацией из активного списка `network_segments` на основе соответствия между IP-адресами устройств (`deviceIp`) и сегментами сети (`segment_ip`). Цель — предоставить полный обзор сетевой активности и конфигурации сети, включая события и сетевые сегменты, которые прямо не связаны.

```
FULL JOIN `network_segments` ON deviceIp = `network_segments`.segment_ip
```

Визуализация ожидаемого результата запроса:

event.deviceIp	event.action	network_segments.segment_ip	network_segments.segment_name
198.51.100.1	Access_Granted	198.51.100.1	Office_Network
NULL	NULL	198.51.100.1	Data_Center
203.0.113.24	Unauthorized_Access	NULL	NULL

В этой таблице представлена информация, объединяющая события сетевой активности и конфигурацию сетевых сегментов. Запись для сегмента `Data_Center` показывает сетевой сегмент без зарегистрированных событий, тогда как запись для IP-адреса 203.0.113.24 демонстрирует событие, для которого нет соответствующего сегмента сети в активном списке.

Работа с оператором `ENRICH`

Оператор `ENRICH` дает возможность заменять или добавлять значения в полях событий на основе данных из соответствующих записей активного списка. Он используется в сочетании с оператором `JOIN`. `ENRICH` применяется только после того, как условие соединения, заданное в `JOIN`, будет выполнено.

```
[SELECT <expression>]
(INNER | LEFT | RIGHT | FULL) JOIN `<active_list_name>` ON <join_condition>
ENRICH <target_field> = <source_field>
```

Здесь:

- `<active_list_name>` — имя активного списка, данные из которого используются для обогащения.

  Имя активного списка необходимо заключать в обратные апострофы.

- `<join_condition>` — условие соединения, определяющее правила поиска совпадений между данными событий и записями в активном списке.

  При обращении к полям таблицы без указания ее названия система автоматически начинает поиск поля в хранилище событий. Если поле в хранилище событий не найдено, поиск поля продолжается в активном списке. Обращение к полю в записях событий может выполняться одним из следующих способов:

  - **Указание только названия поля**. Рекомендуется использовать данный способ, так как поиск полей по умолчанию начинается в "левой" таблице (хранилище событий).
  - **Указание имени таблицы из БД ClickHouse и названия поля**. Имя таблицы ClickHouse, связанной с хранилищем событий, необходимо заключать в обратные апострофы, например: `` `clickhouse_table_name`.field ``.

    Использование имени хранилища событий вместо имени ассоциированной с ним таблицы ClickHouse не допускается.

Обращение к полю активного списка может выполняться одним из следующих способов:

- **Указание только названия поля**. Данный способ допустим только в тех случаях, когда в записях событий отсутствует поле с таким же названием. В противном случае произойдет обращение к полю в записях событий.
- **Указание имени активного списка и названия поля**. Имя активного списка необходимо заключать в обратные апострофы, например: `` `active_list_name`.field ``.
- **Указание псевдонима активного списка и названия поля**. Псевдоним (alias) активного списка допускается задавать как с обратными апострофами, так и без них, например: `` `alias_name`.field `` или `alias_name.field`.

- `<target_field>` — поле в записях событий, для которого будет выполнено обогащение.

Обращение к полю в записях событий может выполняться одним из следующих способов:



- **Указание только названия поля**. Рекомендуется использовать данный способ, так как поиск полей по умолчанию начинается в "левой" таблице (хранилище событий).
- **Указание имени таблицы из БД ClickHouse и названия поля**. Имя таблицы ClickHouse, связанной с хранилищем событий, необходимо заключать в обратные апострофы, например: `` `clickhouse_table_name`.field ``.

  Использование имени хранилища событий вместо имени ассоциированной с ним таблицы ClickHouse не допускается.

- `<source_field>` — поле в активном списке, значения из которого используются для обогащения данных событий.

Обращение к полю активного списка может выполняться одним из следующих способов:



- **Указание только названия поля**. Данный способ допустим только в тех случаях, когда в записях событий отсутствует поле с таким же названием. В противном случае произойдет обращение к полю в записях событий.
- **Указание имени активного списка и названия поля**. Имя активного списка необходимо заключать в обратные апострофы, например: `` `active_list_name`.field ``.
- **Указание псевдонима активного списка и названия поля**. Псевдоним (alias) активного списка допускается задавать как с обратными апострофами, так и без них, например: `` `alias_name`.field `` или `alias_name.field`.

Оператор `ENRICH` позволяет обогащать данные событий из нескольких активных списков в рамках одного запроса. Для точного указания полей из конкретных активных списков следует использовать полное имя поля, включающее имя активного списка и имя поля, разделенные точкой.

Примеры использования ENRICH

Обогащение данных события значениями из одного активного списка

Использование ENRICH

```
LEFT JOIN `user_info` ON userId = `user_info`.name
ENRICH id = `user_info`.age
```

В данном примере запрос обогащает значение поля `id` в записях событий значением поля `age` из активного списка `user_info`, если значение поля `userId` события совпадает с полем `name` в активном списке.

Обогащение данных события дополнительными данными из активных списков

Применение запросов с `INNER JOIN` и `ENRICH` в разделе **Поиск** позволяет обогатить данные событий дополнительными данными из активных списков при условии точного совпадения между записями событий и данными активного списка. В отличие от `LEFT JOIN`, `INNER JOIN` выбирает только те записи, для которых существуют совпадающие данные в обеих таблицах, и далее `ENRICH` используется для обогащения этих данных.



Этот метод подходит для сценариев, где необходимо ограничить анализ событиями, которые имеют прямое соответствие в активных списках, и одновременно обогатить эти события дополнительной информацией из этих списков.

Использование `INNER JOIN` *с* `ENRICH`

```
INNER JOIN `device_info` ON deviceId = `device_info`.device_id
ENRICH deviceType = `device_info`.type
```

Обращение к нескольким активным спискам

Использование `LEFT JOIN` *с* `ENRICH`

```
SELECT id
LEFT JOIN `admin_list` ON userId = `admin_list`.user_id
LEFT JOIN `security_events` ON eventCode = `security_events`.code
ENRICH adminName = `admin_list`.name, securityLevel = `security_events`.level
```

В данном примере запрос обогащает данные событий, используя информацию из двух разных активных списков — `admin_list` и `security_events`. Оператор `ENRICH` применяется для замены (или добавления, если такого поля нет) в записях событий: значения поля `adminName` на основании данных об имени администратора из списка `admin_list` и значения поля `securityLevel` на основании уровня безопасности из списка `security_events`. Оба обогащения основываются на совпадении условий: `userId` с `user_id` для списка администраторов и `eventCode` с `code` для событий.

Преобразование значений с помощью условной логики (функция `if()`)

Функция `if()` позволяет отображать в результатах запроса преобразованные значения вместо

исходных. Это удобно для категоризации результатов или более наглядного представления данных на дашбордах и в отчетах.

```
SELECT if(cond, then, else) AS field1,
...
[GROUP BY field1
ORDER BY field1]
```

Здесь:

- `cond` — проверяемое условие;
- `then` — выражение, результат которого функция вернет, если `cond` истинно;
- `else` — выражение, результат которого возвращается, если `cond` ложно.

В качестве `else` можно использовать вложенную функцию `if()`. Для компактного представления нескольких условий можно использовать функцию `multiIf()`:

```
multiIf(cond_1, then_1, cond_2, then_2, ... else)
```

Здесь:

- `cond_N` — условие.
- `then_N` — результат функции при выполнении условия `cond_N`.
- `else` — результат функции, если ни одно из условий не выполнено.

Пример использования условных функций

Рассмотрим запрос, который возвращает данные, сгруппированные по нескольким полям, вместе с количеством записей:

```
SELECT
    ipAddress,
    deviceName,
    status,
    count(*) AS total_checks
GROUP BY ipAddress, deviceName, status
ORDER BY ipAddress, deviceName, status;
```

Визуализация ожидаемого результата запроса:

ipAddress	deviceName	status	total_checks
192.0.2.0	DeviceA	PASSED	597
192.0.2.0	DeviceA	NOT_PASSED	543
192.0.2.0	DeviceA	NOT_COMPLETED	1116

ipAddress	deviceName	status	total_checks
192.0.2.1	DeviceB	PASSED	56
192.0.2.1	DeviceB	NOT_PASSED	68
192.0.2.1	DeviceB	NOT_COMPLETED	147

Чтобы отобразить в столбце `status` значения на русском языке, преобразуем запрос следующим образом:

```
SELECT
  ipAddress,
  deviceName,
  if(
      equals(status, "PASSED"),
      "Пройденные",
      if(
          equals(status, "NOT_PASSED"),
          "Непройденные",
          if(
              equals(status, "NOT_COMPLETED"),
              "Неприменимые результаты",
              "-"
              )
        )
    ) AS status,
count(*) AS total_checks
GROUP BY ipAddress, deviceName, status
ORDER BY ipAddress, deviceName, status;
```

Здесь равенство поля `status` и указанного значения проверяется с помощью функции `equals()`. Если `equals()` возвращает ложное значение, происходит переход ко вложенному `if()`.

Визуализация ожидаемого результата запроса:

ipAddress	deviceName	status	total_checks
192.0.2.0	DeviceA	Пройденные	597
192.0.2.0	DeviceA	Непройденные	543
192.0.2.0	DeviceA	Неприменимые результаты	1116
192.0.2.1	DeviceB	Пройденные	56
192.0.2.1	DeviceB	Непройденные	68
192.0.2.1	DeviceB	Неприменимые результаты	147

Чтобы упростить запись, используем функцию `multiIf()`:

```
SELECT
  ipAddress,
  deviceName,
  multiIf(
      equals(status, "PASSED"), "Пройденные",
      equals(status, "NOT_PASSED"), "Непройденные",
      equals(status, "NOT_COMPLETED"), "Неприменимые результаты",
      "_"
      ) AS status,
count(*) AS total_checks
GROUP BY ipAddress, deviceName, status
ORDER BY ipAddress, deviceName, status;
```

Максимальное количество и пропуск возвращаемых записей (операторы `LIMIT` и `OFFSET`)

По умолчанию в списке результатов поискового запроса отображается 500 записей. Однако вы можете ограничить количество возвращаемых записей с помощью оператора `LIMIT`.

```
LIMIT <m> [OFFSET <n>]
```

Например, чтобы ограничить выдачу двадцатью записями, используется запрос:

```
LIMIT 20
```

Также можно добавить команду `OFFSET`, которая указывает, сколько строк необходимо пропустить перед началом вывода. Это полезно, если вы хотите начать вывод данных с определенной строки. Например, чтобы пропустить первые 10 строк и вывести следующие 20, используется запрос:

```
LIMIT 20 OFFSET 10
```

Функции

В RQL функции делятся на два основных типа: обычные функции и агрегатные функции, которые обладают различным поведением и областью применения.

Обычные функции применяются к каждой строке индивидуально, вычисляя результат для каждой строки без учета других строк данных. В отличие от них, агрегатные функции обрабатывают данные сразу по нескольким строкам, аккумулируя значения и вычисляя общий результат для набора строк.

Типизация в RQL является строгой, что означает отсутствие неявных преобразований между типами данных. Все функции определены для конкретных типов аргументов, следовательно, иногда может потребоваться использование функций преобразования типов для корректной работы.

В RQL выражения с одинаковым абстрактным синтаксическим деревом (AST) в рамках одного

запроса исполняются однократно. Это включает в себя одинаковые подзапросы, которые также оптимизируются путем их объединения и однократного выполнения.

Функции в RQL возвращают строго одно значение, тип которого определяется типами аргументов, но не их конкретными значениями. Исключением является функция `toFixedString`.

При работе с функциями важно учитывать обработку значений `NULL`. Если хотя бы один из аргументов функции является `NULL`, то результатом функции, как правило, также будет `NULL`, если иное не указано в описании функции.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Агрегатные функции

Агрегатные функции в RQL позволяют проводить расчеты над множествами значений, предоставляя доступ к вычислению общих сумм, средних, максимальных и минимальных значений, а также других агрегатов из больших объемов данных. Их ключевая особенность — это возможность эффективно обработать информацию, полученную из различных строк данных.

Обработка `NULL` в агрегатных функциях предполагает исключение `NULL` значений из обработки, то есть, все `NULL` значения пропускаются и не участвуют в агрегации.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.



Система не поддерживает функции ClickHouse, использующие синтаксис с двумя парами круглых скобок, например `quantile(0.95)(value)`.

Агрегатные функции

Функция	Описание
`any()`	Возвращает произвольное значение из столбца (`any(column)`). Может быть использована для оптимизации, когда точное значение неважно.

Функция	Описание
`anyHeavy()`	Возвращает наиболее часто встречающееся значение в столбце (`anyHeavy(column)`).
`anyLast()`	Возвращает последнее значение в столбце (`anyLast(column)`).
`argMax()`	Возвращает значение аргумента, при котором достигается максимальное значение выражения (`argMax(arg, val)`).
`argMin()`	Возвращает значение аргумента, при котором достигается минимальное значение выражения (`argMin(arg, val)`).
`avg()`	Вычисляет среднее значение для столбца, содержащего числовые данные (`avg(column)`).
`avgWeighted()`	Вычисляет взвешенное среднее значение (`avgWeighted(value, weight)`).
`boundingRatio()`	Вычисляет отношение ограничивающего объема (`boundingRatio(column)`), что может использоваться для оценки плотности данных.
`categoricalInformationValue()`	Вычисляет информационную ценность для категориальных переменных (`categoricalInformationValue(variable, outcome)`), позволяет оценить значимость переменных в предсказании исхода.
`contingency()`	Вычисляет таблицу сопряженности для двух категориальных переменных (`contingency(variable1, variable2)`), используется для анализа взаимосвязи между переменными.
`corr()`	Вычисляет коэффициент корреляции Пирсона между двумя столбцами (`corr(column1, column2)`).
`count()`	Подсчитывает количество строк, удовлетворяющих условиям запроса (`count()`). Если условие не указано, подсчитывает все строки в наборе данных.
`covarPop()`	Вычисляет ковариацию для всей совокупности (`covarPop(column1, column2)`).
`covarSamp()`	Вычисляет ковариацию на основе выборки из генеральной совокупности (`covarSamp(column1, column2)`).
`cramersV()`	Вычисляет коэффициент Крамера для измерения ассоциации между двумя категориальными переменными (`cramersV(variable1, variable2)`), позволяет оценить силу связи между переменными.
`cramersVBiasCorrected()`	Вычисляет скорректированный на смещение коэффициент Крамера (`cramersVBiasCorrected(variable1, variable2)`), обеспечивает более точную оценку ассоциации при малых выборках.
`deltaSum()`	Вычисляет сумму разностей последовательных значений в столбце (`deltaSum(column)`), полезно для анализа изменений значений во времени.
`deltaSumTimestamp()`	Вычисляет сумму разностей между последовательными временными метками (`deltaSumTimestamp(timestampColumn)`), используется для вычисления общего времени между событиями.

Функция	Описание
`entropy()`	Вычисляет энтропию для столбца (`entropy(column)`), что может использоваться для оценки степени разнообразия или неопределенности в данных.
`first_value()`	Возвращает первое значение в порядке сортировки (`first_value(column)`).
`groupArray()`	Агрегирует значения в массив (`groupArray(column)`), обеспечивает возможность работы с массивом значений как с единой сущностью.
`groupArrayInsertAt()`	Вставляет значение в массив по указанному индексу (`groupArrayInsertAt(value, index)`), позволяет модифицировать массивы при агрегации.
`groupArrayIntersect()`	Возвращает массив, содержащий только элементы, присутствующие во всех входных массивах (`groupArrayIntersect(arrays)`), полезно для поиска общих элементов.
`groupArraySorted()`	Сортирует элементы массива (`groupArraySorted(column)`), обеспечивает порядок данных в массиве.
`groupBitAnd()`	Выполняет побитовую операцию "И" над всеми значениями столбца (`groupBitAnd(column)`), применяется для агрегации битовых флагов.
`groupBitmap()`	Агрегирует значения в битовую карту (`groupBitmap(column)`), эффективно для подсчета уникальных значений.
`groupBitmapAnd()`	Выполняет операцию "И" над битовыми картами (`groupBitmapAnd(bitmap1, bitmap2)`), используется для поиска общих уникальных значений между наборами.
`groupBitmapOr()`	Выполняет операцию "ИЛИ" над битовыми картами (`groupBitmapOr(bitmap1, bitmap2)`), применяется для объединения уникальных значений из нескольких наборов.
`groupBitmapXor()`	Выполняет операцию исключающего "ИЛИ" над битовыми картами (`groupBitmapXor(bitmap1, bitmap2)`), полезно для определения уникальных значений, отличающихся между наборами.
`groupBitOr()`	Выполняет побитовую операцию "ИЛИ" над всеми значениями столбца (`groupBitOr(column)`), используется для объединения битовых флагов.
`groupBitXor()`	Выполняет побитовую операцию исключающее "ИЛИ" над всеми значениями столбца (`groupBitXor(column)`), полезно для вычисления различий между битовыми флагами.
`groupUniqArray()`	Агрегирует уникальные значения в массив (`groupUniqArray(column)`), используется для удаления дубликатов из набора данных.
`intervalLengthSum()`	Суммирует длины интервалов между значениями в столбце (`intervalLengthSum(startColumn, endColumn)`), что может применяться для вычисления общего времени или расстояния.
`kurtPop()`	Вычисляет коэффициент эксцесса для всей выборки (`kurtPop(column)`), позволяет оценить, насколько плоское или острое распределение у всей совокупности.

Функция	Описание
`kurtSamp()`	Вычисляет выборочный коэффициент эксцесса (`kurtSamp(column)`), используется для анализа "остроты" пика распределения данных.
`largestTriangleThreeBuckets()`	Применяет алгоритм Largest Triangle Three Buckets для уменьшения числа точек во временных рядах (`largestTriangleThreeBuckets(time, value, buckets)`), используется для оптимизации отображения графиков больших временных рядов.
`last_value()`	Возвращает последнее значение в порядке сортировки (`last_value(column)`).
`max()`	Возвращает максимальное значение в столбце (`max(column)`). Поддерживает такие же типы данных, как и функция `min()`.
`maxIntersections()`	Находит максимальное количество пересечений между наборами значений (`maxIntersections(set1, set2)`), используется для определения наибольшего перекрытия между группами.
`maxIntersectionsPosition()`	Находит позицию максимального количества пересечений (`maxIntersectionsPosition(set1, set2)`), позволяет определить, где происходит наибольшее перекрытие между наборами данных.
`maxMap()`	Находит максимальные элементы входных массивов, группируя по ключам (`maxMap(keysArray, valuesArray)`), применяется для определения максимального значения по ключу.
`meanZTest()`	Выполняет Z-тест для среднего значения (`meanZTest(sampleMean, populationMean, stdDev, sampleSize)`), применяется для проверки гипотезы о равенстве средних в генеральной совокупности и выборке.
`median()`	Вычисляет медиану значений в столбце (`median(column)`).
`min()`	Возвращает минимальное значение в столбце (`min(column)`). Может применяться к числовым, строковым и некоторым другим типам данных.
`minMap()`	Находит минимальные элементы входных массивов, группируя по ключам (`minMap(keysArray, valuesArray)`), используется для определения минимального значения по ключу.
`rankCorr()`	Вычисляет ранговую корреляцию между двумя переменными (`rankCorr(column1, column2)`).
`simpleLinearRegression()`	Вычисляет параметры простой линейной регрессии для двух переменных (`simpleLinearRegression(y, x)`), позволяя оценить линейную зависимость между ними.
`skewPop()`	Вычисляет коэффициент асимметрии для всей совокупности (`skewPop(column)`), полезно для анализа формы распределения на всей выборке.
`skewSamp()`	Вычисляет выборочный коэффициент асимметрии для набора данных (`skewSamp(column)`), используется для оценки асимметрии распределения.

Функция	Описание
`sparkBar()`	Генерирует текстовое представление гистограммы для распределения значений (`sparkBar(column)`), предоставляет визуализацию распределения в текстовом формате.
`stddevPop()`	Вычисляет стандартное отклонение для всей выборки (`stddevPop(column)`). Поддерживается только для числовых данных.
`stddevSamp()`	Вычисляет выборочное стандартное отклонение (`stddevSamp(column)`). Используется для оценки стандартного отклонения на основе выборки из генеральной совокупности.
`stochasticLinearRegression()`	Вычисляет параметры стохастической линейной регрессии (`stochasticLinearRegression(y, x)`), используется для моделирования линейной зависимости с учетом случайного компонента.
`stochasticLogisticRegression()`	Вычисляет параметры стохастической логистической регрессии (`stochasticLogisticRegression(y, x)`), применяется для моделирования вероятности события на основе одной или нескольких независимых переменных.
`studentTTest()`	Выполняет t-тест Стьюдента для сравнения средних двух выборок. Используется без параметра `confidenceLevel` (`studentTTest(sample1, sample2)`).
`sum()`	Суммирует значения по столбцу (`sum(column)`). Применима только к числовым типам данных.
`sumCount()`	Вычисляет сумму и количество значений одновременно (`sumCount(column)`), полезно для получения общей суммы и размера выборки в одном запросе.
`sumKahan()`	Суммирует числа с использованием алгоритма Кахана для уменьшения ошибки округления (`sumKahan(column)`).
`sumMap()`	Суммирует элементы входных массивов, группируя по ключам (`sumMap(keysArray, valuesArray)`), полезно для агрегации данных по ключу.
`sumWithOverflow()`	Суммирует значения, позволяя переполнение (`sumWithOverflow(column)`), применяется для обработки очень больших сумм, где может произойти переполнение.
`theilsU()`	Вычисляет индекс Тейла для измерения ассоциации между двумя категориальными переменными (`theilsU(variable1, variable2)`), предоставляет альтернативную метрику для оценки взаимосвязей.
`uniq()`	Подсчитывает количество уникальных значений в столбце (`uniq(column)`), используется для определения количества различных элементов.
`uniqCombined()`	Приблизительно подсчитывает количество уникальных значений с помощью алгоритма HyperLogLog (`uniqCombined(column)`), эффективно для больших объемов данных.

Функция	Описание
`uniqCombined64()`	Приблизительно подсчитывает количество уникальных значений с помощью 64-битной версии алгоритма HyperLogLog (`uniqCombined64(column)`), предназначено для очень больших наборов данных.
`uniqExact()`	Точно подсчитывает количество уникальных значений (`uniqExact(column)`), предпочтительно для небольших наборов данных, где требуется точность.
`uniqHLL12()`	Приблизительно подсчитывает количество уникальных значений с использованием алгоритма HyperLogLog с точностью 12 бит (`uniqHLL12(column)`), используется для оптимизации расчета уникальности в больших наборах данных.
`uniqTheta()`	Приблизительно подсчитывает количество уникальных значений с использованием алгоритма Theta Sketch (`uniqTheta(column)`), предлагает улучшенную производительность и точность на больших объемах данных.
`varPop()`	Вычисляет дисперсию для всей выборки (`varPop(column)`).
`varSamp()`	Вычисляет дисперсию выборки (`varSamp(column)`), что полезно для анализа вариативности данных в выборке.
`welchTTest()`	Выполняет t-тест Уэлча, модификацию t-теста Стьюдента, предназначенную для использования с выборками разного размера и дисперсии. Используется без параметра `confidenceLevel` (`welchTTest(sample1, sample2)`).

Условные функции

Условные функции в RQL обеспечивают выполнение логических и сравнительных операций, позволяя определить выполнение различных выражений в зависимости от заданных условий.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Условные функции

Функция	Описание
`if(cond, then, else)`	Выполняет условное ветвление. Если условие `cond` истинно (не равно нулю), возвращает результат выражения `then`. Если `cond` ложно (равно нулю) или NULL, возвращает результат выражения `else`. Типы выражений `then` и `else` должны быть схожими.
`multiIf(cond_1, then_1, cond_2, then_2, ⋯, else)`	Позволяет компактно записать несколько условий. Функция принимает 2N+1 аргументов: условия `cond_N` и соответствующие результаты `then_N`, а также результат `else` для случая, когда ни одно из условий не выполнено.

Функции сравнения

Типы данных, которые могут быть сравнены:

- числа;
- строки и фиксированные строки;
- даты;
- даты со временем.

Только значения в рамках одной группы могут быть сравнены (например, `UInt16` и `UInt64`), но не между группами (например, `UInt16` и `DateTime`).



Строки сравниваются по байтам. Если одна из строк содержит символы в кодировке UTF-8, занимающие несколько байт, это может привести к неожиданным результатам.

Строка `S1`, которая имеет другую строку `S2` в качестве префикса, считается длиннее, чем `S2`.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции сравнения

Функция	Описание
`equals`, `=`, `==`	Проверяет равенство двух значений. Возвращает 1 (истина), если значения равны, и 0 (ложь) в противном случае.
`greater`, `>`	Проверяет, больше ли первое значение, чем второе. Возвращает `1` (истина), если первое значение больше второго, и `0` (ложь) в противном случае.

Функция	Описание
`greaterOrEquals`, `>=`	Проверяет, больше ли первое значение, чем второе, или они равны. Возвращает `1` (истина), если первое значение больше второго или равно ему, и `0` (ложь) в противном случае.
`greatest(⋯)`	Возвращает наибольшее значение из списка значений. Все элементы списка должны быть сопоставимыми типами. Тип возвращаемого значения может быть продвинут до более общего типа для сравнения.
`least(⋯)`	Возвращает наименьшее значение из списка значений. Аналогично `greatest`, все элементы списка должны быть сопоставимыми типами, и тип возвращаемого значения может быть продвинут.
`less`, `<`	Проверяет, меньше ли первое значение, чем второе. Возвращает `1` (истина), если первое значение меньше второго, и `0` (ложь) в противном случае.
`lessOrEquals`, `<=`	Проверяет, меньше ли первое значение, чем второе, или они равны. Возвращает `1` (истина), если первое значение меньше второго или равно ему, и `0` (ложь) в противном случае.
`notEquals`, `!=`, `<>`	Проверяет, неравны ли два значения. Возвращает `1` (истина), если значения неравны, и `0` (ложь), если равны.

Логические функции

Нижеперечисленные функции выполняют логические операции над аргументами произвольных числовых типов. Они возвращают либо 0, либо 1 типа `UInt8` или в некоторых случаях `NULL`.

Ноль в качестве аргумента рассматривается как ложь, ненулевые значения считаются истиной.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Логические функции

Функция	Описание
`and(val1, val2⋯)`	Вычисляет результат логической конъюнкции (логического "И") между двумя или более значениями. Возвращает `1`, если все аргументы истинны (ненулевые), `0` — если хотя бы один аргумент ложен (ноль), и `NULL`, если среди аргументов нет нулей, но есть хотя бы один `NULL`.

Функция	Описание
`not(val)`	Вычисляет логическое отрицание ("НЕ") аргумента. Возвращает `1`, если аргумент ложен (ноль), `0` — если аргумент истинен (ненулевой), и NULL, если аргумент NULL.
`or(val1, val2⋯)`	Вычисляет результат логической дизъюнкции (логического "ИЛИ") между двумя или более значениями. Возвращает `1`, если хотя бы один аргумент истинен (ненулевой), `0` — если все аргументы ложны (нули), и `NULL`, если среди аргументов нет ненулевых значений и есть `NULL`.
`xor(val1, val2⋯)`	Вычисляет результат логической исключающей дизъюнкции (XOR) между двумя или более значениями. Для пары значений возвращает `1`, если одно из значений истинно (ненулевое), а другое ложно (ноль). Возвращает `0`, если оба значения одинаковы (оба нули или оба ненулевые). Для более чем двух значений применяется рекурсивно. Возвращает `NULL`, если среди аргументов хотя бы один `NULL`.

Функции преобразования типов



В RQL поведение функций `to<type>` и `cast` в некоторых случаях различается, например, при работе с `LowCardinality`: функция `cast` убирает признак `LowCardinality`, в то время как функции `to<type>` его не убирают. То же самое касается `Nullable`. Такое поведение не соответствует стандарту SQL, но его можно изменить с помощью настройки `cast_keep_nullable`.

Возможна потеря данных при конвертации значений из типа данных в меньший тип данных (например, с `Int64` на `Int32`) или между несовместимыми типами данных (например, с `String` на `Int`). Убедитесь, что результат конвертации соответствует ожиданиям.

Функции преобразования типов в RQL позволяют преобразовывать значения из одного типа данных в другой.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции преобразования в заданные типы

Функции преобразования в заданные типы

Функция	Описание
`accurateCast(x, T)`	Преобразует значение `x` в тип `T` с проверкой на переполнение и потерю данных.
`accurateCastOrDefault(x, T, default_value)`	Преобразует значение `x` в тип `T` с проверкой на переполнение и потерю данных, возвращая значение по умолчанию при неудаче.
`accurateCastOrNull(x, T)`	Преобразует значение `x` в тип `T` с проверкой на переполнение и потерю данных, возвращая `NULL` при неудаче.
`CAST(x, T)`	Явное преобразование значения `x` в тип `T`.
`reinterpretAs<Type>`	Преобразует значение в заданный тип без изменения битового представления значения.

Функции преобразования в численные типы

Имена функций, преобразующих значения в численные типы, имеют следующую структуру:

```
to<Type><Precision>[<ErrorHandling>]
```

Здесь:

- `<Type>` — численный тип;
- `<Precision>` — битность или точность числа;
- `<ErrorHandling>` — результат, возвращаемый при неудаче. Если поведение не указано, генерируется исключение.

Конкретные функции перечислены далее.

Функции преобразования в целочисленные типы

Функции преобразования в целочисленные типы

Функция	Описание
`toInt8`	Преобразует значение в тип `Int8`. В случае неудачи генерируется исключение.
`toInt8OrDefault`	Преобразует значение в тип `Int8`. В случае неудачи возвращает значение по умолчанию.
`toInt8OrNull`	Преобразует значение в тип `Int8`. В случае неудачи возвращает `NULL`.
`toInt8OrZero`	Преобразует значение в тип `Int8`. В случае неудачи возвращает `0`.
`toInt16`	Преобразует значение в тип `Int16`. В случае неудачи генерируется исключение.
`toInt16OrDefault`	Преобразует значение в тип `Int16`. В случае неудачи возвращает значение по умолчанию.
`toInt16OrNull`	Преобразует значение в тип `Int16`. В случае неудачи возвращает `NULL`.
`toInt16OrZero`	Преобразует значение в тип `Int16`. В случае неудачи возвращает `0`.

Функция	Описание
`toInt32`	Преобразует значение в тип `Int32`. В случае неудачи генерируется исключение.
`toInt32OrDefault`	Преобразует значение в тип `Int32`. В случае неудачи возвращает значение по умолчанию.
`toInt32OrNull`	Преобразует значение в тип `Int32`. В случае неудачи возвращает `NULL`.
`toInt32OrZero`	Преобразует значение в тип `Int32`. В случае неудачи возвращает `0`.
`toInt64`	Преобразует значение в тип `Int64`. В случае неудачи генерируется исключение.
`toInt64OrDefault`	Преобразует значение в тип `Int64`. В случае неудачи возвращает значение по умолчанию.
`toInt64OrNull`	Преобразует значение в тип `Int64`. В случае неудачи возвращает `NULL`.
`toInt64OrZero`	Преобразует значение в тип `Int64`. В случае неудачи возвращает `0`.
`toInt128`	Преобразует значение в тип `Int128`. В случае неудачи генерируется исключение.
`toInt128OrDefault`	Преобразует значение в тип `Int128`. В случае неудачи возвращает значение по умолчанию.
`toInt128OrNull`	Преобразует значение в тип `Int128`. В случае неудачи возвращает `NULL`.
`toInt128OrZero`	Преобразует значение в тип `Int128`. В случае неудачи возвращает `0`.
`toInt256`	Преобразует значение в тип `Int256`. В случае неудачи генерируется исключение.
`toInt256OrDefault`	Преобразует значение в тип `Int256`. В случае неудачи возвращает значение по умолчанию.
`toInt256OrNull`	Преобразует значение в тип `Int256`. В случае неудачи возвращает `NULL`.
`toInt256OrZero`	Преобразует значение в тип `Int256`. В случае неудачи возвращает `0`.

Функции преобразования в беззнаковые целочисленные типы

Функции преобразования в беззнаковые целочисленные типы

Функция	Описание
`toUInt8`	Преобразует значение в тип `UInt8`. В случае неудачи генерируется исключение.
`toUInt8OrDefault`	Преобразует значение в тип `UInt8`. В случае неудачи возвращает значение по умолчанию.
`toUInt8OrNull`	Преобразует значение в тип `UInt8`. В случае неудачи возвращает `NULL`.
`toUInt8OrZero`	Преобразует значение в тип `UInt8`. В случае неудачи возвращает `0`.
`toUInt16`	Преобразует значение в тип `UInt16`. В случае неудачи генерируется исключение.
`toUInt16OrDefault`	Преобразует значение в тип `UInt16`. В случае неудачи возвращает значение по умолчанию.

Функция	Описание
toUInt16OrNull	Преобразует значение в тип UInt16. В случае неудачи возвращает NULL.
toUInt16OrZero	Преобразует значение в тип UInt16. В случае неудачи возвращает 0.
toUInt32	Преобразует значение в тип UInt32. В случае неудачи генерируется исключение.
toUInt32OrDefault	Преобразует значение в тип UInt32. В случае неудачи возвращает значение по умолчанию.
toUInt32OrNull	Преобразует значение в тип UInt32. В случае неудачи возвращает NULL.
toUInt32OrZero	Преобразует значение в тип UInt32. В случае неудачи возвращает 0.
toUInt64	Преобразует значение в тип UInt64. В случае неудачи генерируется исключение.
toUInt64OrDefault	Преобразует значение в тип UInt64. В случае неудачи возвращает значение по умолчанию.
toUInt64OrNull	Преобразует значение в тип UInt64. В случае неудачи возвращает NULL.
toUInt64OrZero	Преобразует значение в тип UInt64. В случае неудачи возвращает 0.
toUInt128	Преобразует значение в тип UInt128. В случае неудачи генерируется исключение.
toUInt128OrDefault	Преобразует значение в тип UInt128. В случае неудачи возвращает значение по умолчанию.
toUInt128OrNull	Преобразует значение в тип UInt128. В случае неудачи возвращает NULL.
toUInt128OrZero	Преобразует значение в тип UInt128. В случае неудачи возвращает 0.
toUInt256	Преобразует значение в тип UInt256. В случае неудачи генерируется исключение.
toUInt256OrDefault	Преобразует значение в тип UInt256. В случае неудачи возвращает значение по умолчанию.
toUInt256OrNull	Преобразует значение в тип UInt256. В случае неудачи возвращает NULL.
toUInt256OrZero	Преобразует значение в тип UInt256. В случае неудачи возвращает 0.

Функции преобразования в типы с плавающей запятой

Функции преобразования в типы с плавающей запятой

Функция	Описание
toFloat32	Преобразует значение в тип Float32. В случае неудачи генерируется исключение.
toFloat32OrDefault	Преобразует значение в тип Float32. В случае неудачи возвращает значение по умолчанию.
toFloat32OrNull	Преобразует значение в тип Float32. В случае неудачи возвращает NULL.

Функция	Описание
toFloat32OrZero	Преобразует значение в тип Float32. В случае неудачи возвращает 0.
toFloat64	Преобразует значение в тип Float64. В случае неудачи генерируется исключение.
toFloat64OrDefault	Преобразует значение в тип Float64. В случае неудачи возвращает значение по умолчанию.
toFloat64OrNull	Преобразует значение в тип Float64. В случае неудачи возвращает NULL.
toFloat64OrZero	Преобразует значение в тип Float64. В случае неудачи возвращает 0.

Функции преобразования в десятичные типы

Функции преобразования в десятичные типы

Функция	Описание
toDecimal32	Преобразует значение в тип Decimal32. В случае неудачи генерируется исключение.
toDecimal32OrDefault	Преобразует значение в тип Decimal32. В случае неудачи возвращает значение по умолчанию.
toDecimal32OrNull	Преобразует значение в тип Decimal32. В случае неудачи возвращает NULL.
toDecimal32OrZero	Преобразует значение в тип Decimal32. В случае неудачи возвращает 0.
toDecimal64	Преобразует значение в тип Decimal64. В случае неудачи генерируется исключение.
toDecimal64OrDefault	Преобразует значение в тип Decimal64. В случае неудачи возвращает значение по умолчанию.
toDecimal64OrNull	Преобразует значение в тип Decimal64. В случае неудачи возвращает NULL.
toDecimal64OrZero	Преобразует значение в тип Decimal64. В случае неудачи возвращает 0.
toDecimal128	Преобразует значение в тип Decimal128. В случае неудачи генерируется исключение.
toDecimal128OrDefault	Преобразует значение в тип Decimal128. В случае неудачи возвращает значение по умолчанию.
toDecimal128OrNull	Преобразует значение в тип Decimal128. В случае неудачи возвращает NULL.
toDecimal128OrZero	Преобразует значение в тип Decimal128. В случае неудачи возвращает 0.
toDecimal256	Преобразует значение в тип Decimal256. В случае неудачи генерируется исключение.
toDecimal256OrDefault	Преобразует значение в тип Decimal256. В случае неудачи возвращает значение по умолчанию.

Функция	Описание
`toDecimal256OrNull`	Преобразует значение в тип `Decimal256`. В случае неудачи возвращает `NULL`.
`toDecimal256OrZero`	Преобразует значение в тип `Decimal256`. В случае неудачи возвращает `0`.

Функции преобразования типов даты и времени

Функции преобразования типов даты и времени

Функция	Описание
`dateTime64ToSnowflake(value)`	Преобразует значение `DateTime64` в первый идентификатор Snowflake ID на текущий момент.
`dateTimeToSnowflake(value)`	Преобразует значение `DateTime` в первый идентификатор Snowflake ID на текущий момент.
`fromUnixTimestamp64Micro(value [, timezone])`	Преобразует значение `Int64`, представляющее количество микросекунд с начала эпохи Unix, в значение `DateTime64` с точностью до микросекунд и опционально применяет часовой пояс.
`fromUnixTimestamp64Milli(value [, timezone])`	Преобразует значение `Int64`, представляющее количество миллисекунд с начала эпохи Unix, в значение `DateTime64` с точностью до миллисекунд и опционально применяет часовой пояс.
`fromUnixTimestamp64Nano(value [, timezone])`	Преобразует значение `Int64`, представляющее количество наносекунд с начала эпохи Unix, в значение `DateTime64` с точностью до наносекунд и опционально применяет часовой пояс.
`parseDateTime32BestEffort`	Аналогично `parseDateTimeBestEffort`, но преобразует строку в тип `DateTime32`.
`parseDateTime32BestEffortOrNull`	Аналогично `parseDateTimeBestEffortOrNull`, но преобразует строку в тип `DateTime32`, возвращая `NULL` при неудаче.
`parseDateTime32BestEffortOrZero`	Аналогично `parseDateTimeBestEffortOrZero`, но преобразует строку в тип `DateTime32`, возвращая нулевую дату и время при неудаче.
`parseDateTime64BestEffort`	Работает так же, как `parseDateTimeBestEffort`, но также принимает миллисекунды и микросекунды, возвращая тип данных `DateTime64` с заданной точностью.
`parseDateTime64BestEffortOrNull`	Работает так же, как `parseDateTime64BestEffort`, но возвращает `NULL`, если формат даты не может быть обработан.
`parseDateTime64BestEffortOrZero`	Работает так же, как `parseDateTime64BestEffort`, но возвращает нулевую дату и время (`1970-01-01 00:00:00.000`), если формат даты не может быть обработан.
`parseDateTime64BestEffortUS`	Работает аналогично `parseDateTime64BestEffort`, но предполагает американский формат даты (MM/DD/YYYY и подобные) в случае неоднозначности.

Функция	Описание
parseDateTime64BestEffortUSOrNull	Работает аналогично parseDateTime64BestEffortUS, но возвращает NULL, если входная строка не может быть преобразована в тип данных DateTime64.
parseDateTime64BestEffortUSOrZero	Работает аналогично parseDateTime64BestEffortUS, но возвращает нулевую дату и время (1970-01-01 00:00:00.000), если входная строка не может быть преобразована.
parseDateTimeBestEffort	Пытается преобразовать строку в тип DateTime, распознавая множество форматов даты и времени, включая ISO 8601, RFC 1123, формат даты и времени ClickHouse, а также другие форматы. Поддерживает разбор строковых представлений Unix timestamp, различные строковые форматы даты и времени, а также строковые представления с указанием часового пояса.
parseDateTimeBestEffortOrNull	Работает так же, как parseDateTimeBestEffort, но возвращает NULL, если строка не может быть преобразована в дату и время.
parseDateTimeBestEffortOrZero	Работает так же, как parseDateTimeBestEffort, но возвращает нулевую дату и время (1970-01-01 00:00:00), если строка не может быть преобразована.
parseDateTimeBestEffortUS	Ведет себя как parseDateTimeBestEffort для стандартных форматов даты, но также предпочитает американский формат даты (MM/DD/YYYY и подобные) в случае неоднозначности. Если месяц и день не могут быть однозначно идентифицированы, функция возвращается к поведению parseDateTimeBestEffort.
parseDateTimeBestEffortUSOrNull	Работает так же, как parseDateTimeBestEffortUS, но возвращает NULL, если строка не может быть преобразована в дату и время.
parseDateTimeBestEffortUSOrZero	Работает так же, как parseDateTimeBestEffortUS, но возвращает нулевую дату и время (1970-01-01 00:00:00), если строка не может быть преобразована.
snowflakeToDateTime(value [, timezone])	Извлекает время из Snowflake ID и преобразует его в формат DateTime, принимая во внимание часовой пояс, если он указан.
snowflakeToDateTime64(value [, timezone])	Извлекает время из Snowflake ID и преобразует его в формат DateTime64, принимая во внимание часовой пояс, если он указан.
toDate	Преобразует значение в тип даты. В случае неудачи генерируется исключение.
toDate32	Преобразует значение в тип даты с расширенным диапазоном. В случае неудачи генерируется исключение.
toDate32OrDefault	Преобразует значение в тип даты с расширенным диапазоном. В случае неудачи возвращает значение по умолчанию.
toDate32OrNull	Преобразует значение в тип даты с расширенным диапазоном. В случае неудачи возвращает NULL.
toDate32OrZero	Преобразует значение в тип даты с расширенным диапазоном. В случае неудачи возвращает минимально возможную дату.

Функция	Описание
toDateOrDefault	Преобразует значение в тип даты. В случае неудачи возвращает значение по умолчанию.
toDateOrNull	Преобразует значение в тип даты. В случае неудачи возвращает NULL.
toDateOrZero	Преобразует значение в тип даты. В случае неудачи возвращает '0000-00-00'.
toDateTime	Преобразует значение в тип даты со временем. В случае неудачи генерируется исключение.
toDateTime64	Преобразует значение в тип даты со временем с высокой точностью. В случае неудачи генерируется исключение.
toDateTime64OrDefault	Преобразует значение в тип даты со временем с высокой точностью. В случае неудачи возвращает значение по умолчанию.
toDateTime64OrNull	Преобразует значение в тип даты со временем с высокой точностью. В случае неудачи возвращает NULL.
toDateTime64OrZero	Преобразует значение в тип даты со временем с высокой точностью. В случае неудачи возвращает '0000-00-00 00:00:00.000'.
toDateTimeOrDefault	Преобразует значение в тип даты со временем. В случае неудачи возвращает значение по умолчанию.
toDateTimeOrNull	Преобразует значение в тип даты со временем. В случае неудачи возвращает NULL.
toDateTimeOrZero	Преобразует значение в тип даты со временем. В случае неудачи возвращает '0000-00-00 00:00:00'.
toLowCardinality(expr)	Преобразует входные данные в версию LowCardinality того же типа данных, оптимизируя обработку и хранение данных за счет снижения количества уникальных значений.
toUnixTimestamp64Micro(value)	Преобразует значение DateTime64 в значение Int64, представляющее количество микросекунд с начала эпохи Unix. Входное значение округляется соответствующим образом.
toUnixTimestamp64Milli(value)	Преобразует значение DateTime64 в значение Int64, представляющее количество миллисекунд с начала эпохи Unix (1970-01-01 00:00:00 UTC). Входное значение округляется соответствующим образом.
toUnixTimestamp64Nano(value)	Преобразует значение DateTime64 в значение Int64, представляющее количество наносекунд с начала эпохи Unix. Входное значение округляется соответствующим образом.

Функции преобразования в интервалы времени

Функции преобразования в интервалы времени

Функция	Описание
toIntervalDay	Преобразует числовое значение в тип IntervalDay.
toIntervalHour	Преобразует числовое значение в тип IntervalHour.

Функция	Описание
`toIntervalMinute`	Преобразует числовое значение в тип IntervalMinute.
`toIntervalMonth`	Преобразует числовое значение в тип IntervalMonth.
`toIntervalQuarter`	Преобразует числовое значение в тип IntervalQuarter.
`toIntervalSecond`	Преобразует числовое значение в тип IntervalSecond.
`toIntervalWeek`	Преобразует числовое значение в тип IntervalWeek.
`toIntervalYear`	Преобразует числовое значение в тип IntervalYear.

Функции преобразования в строковые типы

Функции преобразования в строковые типы

Функция	Описание
`formatRow(format, x, y, …)`	Преобразует произвольные выражения в строку заданного формата, например, `CSV` или `TSV`, и может включать префикс или суффикс в каждой строке в зависимости от настроек.
`formatRowNoNewline(format, x, y, …)`	Аналогично функции `formatRow`, но удаляет завершающий перевод строки в конце отформатированной строки, если он есть.
`toDecimalString(number, scale)`	Преобразует числовое значение в строку с указанным числом знаков после десятичной точки.
`toFixedString(s, N)`	Преобразует строку `s` в фиксированную строку заданной длины `N`. Строка короче `N` дополняется нулевыми байтами.
`toString`	Преобразует значение в строковый тип. В случае неудачи генерируется исключение.
`toStringCutToZero(s)`	Обрезает строку до первого нулевого байта.

Функции для работы с Nullable-аргументами

Функции для работы с Nullable-аргументами в RQL предназначены для управления и обработки данных, которые могут содержать `NULL` значения.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции для работы с Nullable-аргументами

Функция	Описание
`assumeNotNull(x)`	Приводит значение типа `Nullable` к не `Nullable`, если значение не `NULL`. Возвращает неспецифицированный результат, если исходное значение — `NULL`.
`coalesce(x, ⋯)`	Последовательно слева направо проверяет, являются ли переданные аргументы `NULL`, и возвращает первый не `NULL` аргумент. Если все аргументы — `NULL`, возвращает `NULL`.
`ifNull(x, alt)`	Возвращает альтернативное значение `alt`, если основной аргумент `x` — `NULL`. В противном случае возвращает значение `x`.
`isNotNull(x)`	Проверяет, не является ли аргумент `x` `NULL`. Возвращает `0`, если `x` — `NULL`, и `1`, если `x` — не `NULL`.
`isNull(x)`	Проверяет, является ли аргумент `x` `NULL`. Возвращает `1`, если `x` — `NULL`, и `0`, если `x` — не `NULL`. Синоним: `ISNULL`.
`nullIf(x, y)`	Возвращает `NULL`, если аргументы `x` и `y` равны. В противном случае возвращает `x`.
`toNullable(x)`	Преобразует тип аргумента `x` к `Nullable`. Возвращает входное значение с типом `Nullable`.

Математические функции

Все функции возвращают числа типа `Float64`.

Арифметические функции рассмотрены отдельно.



Точность получаемых результатов приближена к максимально возможной. Полученные значения могут не совпадать точно с ближайшими к ним числами, представимыми в машинном представлении.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции математических констант и проверки знака

Функция	Описание
`e()`	Возвращает число *e* (основание натурального логарифма), приблизительно равное 2.718.
`pi()`	Возвращает число π ("пи"), приблизительно равное 3.14159.

Функция	Описание
`sign(x)`	Возвращает знак числа x: -1, 0 или 1.

Функции возведения в степень и извлечения корня

Функция	Описание
`cbrt(x)`	Возвращает кубический корень из x.
`exp(x)`	Возвращает значение e в степени x.
`exp10(x)`	Возвращает 10 в степени x.
`exp2(x)`	Возвращает 2 в степени x.
`intExp10(x)`	Аналог `exp10`, но возвращает UInt64.
`intExp2(x)`	Аналог `exp` с возвращением значения UInt64.
`pow(x, y)`	Возвращает x в степени y.
`sqrt(x)`	Возвращает квадратный корень из x.

Логарифмические функции

Функция	Описание
`log(x)`	Возвращает натуральный логарифм x.
`log10(x)`	Возвращает десятичный логарифм x.
`log1p(x)`	Возвращает логарифм (1 + x).
`log2(x)`	Возвращает двоичный логарифм x.

Тригонометрические функции

Функция	Описание
`acos(x)`	Возвращает арккосинус `x`.
`acosh(x)`	Возвращает обратный гиперболический косинус `x`.
`asin(x)`	Возвращает арксинус `x`.
`asinh(x)`	Возвращает обратный гиперболический синус `x`.
`atan(x)`	Возвращает арктангенс `x`.
`atan2(y, x)`	Возвращает арктангенс двух переменных `y` и `x`.
`atanh(x)`	Возвращает обратный гиперболический тангенс `x`.
`cos(x)`	Возвращает косинус угла `x`, заданного в радианах.
`cosh(x)`	Возвращает гиперболический косинус угла `x`.
`degrees(x)`	Конвертирует радианы в градусы.
`hypot(x, y)`	Возвращает длину гипотенузы прямоугольного треугольника с катетами `x` и `y`.

Функция	Описание
`radians(x)`	Конвертирует градусы в радианы.
`sin(x)`	Возвращает синус угла `x`, заданного в радианах.
`sinh(x)`	Возвращает гиперболический синус угла `x`.
`tan(x)`	Возвращает тангенс угла `x`, заданного в радианах.

Статистические функции

Функция	Описание
`erf(x)`	Возвращает функцию ошибок для `x`.
`erfc(x)`	Возвращает дополнительную функцию ошибок для `x`.
`width_bucket(operand, low, high, count)`	Возвращает номер интервала, в который попадает значение `operand`, в гистограмме с равномерно распределенными интервалами от `low` до `high`, число которых равно `count`.

Гамма-функция и факториал

Функция	Описание
`factorial(n)`	Возвращает факториал n.
`lgamma(x)`	Возвращает натуральный логарифм гамма-функции от x.
`tgamma(x)`	Возвращает гамма-функцию от x.

Арифметические функции

Для всех арифметических функций в RQL тип результата вычисляется как минимальный числовой тип, который может вместить результат, если такой тип существует. Выбор минимального типа происходит с учетом числа бит, знаковости и наличия плавающей точки. Если минимальных бит не хватает, то используется тип с максимальной битностью.

Примеры использования:

```
SELECT toTypeName(0), toTypeName(0 + 0), toTypeName(0 + 0 + 0), toTypeName(0 + 0 + 0 +
0)
```

Визуализация ожидаемого результата запроса

toTypeName(0)	toTypeName(plus(0, 0))	toTypeName(plus(plus(0, 0), 0))	toTypeName(plus(plus(plus(0, 0), 0), 0))
UInt8	UInt16	UInt32	UInt64

Арифметические функции работают для любой пары типов из `UInt8`, `UInt16`, `UInt32`, `UInt64`, `Int8`, `Int16`, `Int32`, `Int64`, `Float32`, `Float64`. Переполнение обрабатывается аналогично C++.



Использование функций `multiplyDecimal` и `divideDecimal` предпочтительнее в

ситуациях, когда необходима высокая точность вычислений и контроль за размером дробной части. В то же время следует учитывать, что эти функции могут работать медленнее стандартных арифметических функций `multiply` и `divide`.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.



Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Арифметические функции

Функция	Описание
`abs(a)`	Возвращает абсолютное значение числа `a`. Для беззнаковых типов ничего не изменяет.
`divide(a, b)`	Вычисляет частное чисел `a` и `b`. Результат всегда с плавающей запятой. Для целочисленного деления используйте `intDiv`.
`divideDecimal(a, b[, result_scale])`	Делит два числа типа `Decimal`, позволяя контролировать размер дробной части результата через `result_scale`.
`gcd(a, b)`	Вычисляет наибольший общий делитель чисел `a` и `b`.
`intDiv(a, b)`	Вычисляет целочисленное частное чисел `a` и `b` с округлением вниз.
`intDivOrZero(a, b)`	Аналог `intDiv`, но возвращает ноль при делении на ноль или при определенных условиях переполнения.
`lcm(a, b)`	Вычисляет наименьшее общее кратное чисел `a` и `b`.
`max2(value1, value2)`	Сравнивает числа `value1` и `value2`, возвращая максимальное значение. Результат приводится к `Float64`.
`min2(value1, value2)`	Сравнивает числа `value1` и `value2`, возвращая минимальное значение. Результат приводится к `Float64`.
`minus(a, b)`	Вычисляет разность чисел `a` и `b`. Поддерживает вычитание целых чисел из даты и даты с временем, аналогично `plus`.
`modulo(a, b)`	Вычисляет остаток от деления числа `a` на `b`. Результат целочисленный для целых чисел и с плавающей точкой, если хотя бы один из аргументов таковым является.
`moduloOrZero(a, b)`	Аналог `modulo`, но возвращает ноль при делении на ноль.
`multiply(a, b)`	Вычисляет произведение чисел `a` и `b`.
`multiplyDecimal(a, b[, result_scale])`	Умножает два числа типа `Decimal`, позволяя контролировать размер дробной части результата через `result_scale`.
`negate(a)`	Вычисляет число, обратное по знаку к `a`.

Функция	Описание
`plus(a, b)`	Вычисляет сумму чисел `a` и `b`. Поддерживает сложение целых чисел с датой и датой с временем, где прибавление к дате означает дни, а к дате с временем — секунды.

Битовые функции

Битовые функции работают для любой пары типов из `UInt8`, `UInt16`, `UInt32`, `UInt64`, `Int8`, `Int16`, `Int32`, `Int64`, `Float32`, `Float64`.

Тип результата — целое число, битность которого равна максимальной битности аргументов. Если хотя бы один аргумент знаковый, то результат — знаковое число. Если аргумент — число с плавающей запятой, оно приводится к Int64.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Битовые функции

Функция	Описание
`bitAnd(a, b)`	Выполняет побитовую операцию "И" (AND) над двумя числами `a` и `b`. Тип результата — целое число, битность которого равна максимальной битности аргументов.
`bitCount(x)`	Подсчитывает количество бит, установленных в 1, в числе `x`. Возвращает количество этих бит.
`bitHammingDistance(int1, int2)`	Возвращает расстояние Хэмминга между двумя целыми числами `int1` и `int2`, то есть количество позиций, в которых соответствующие биты различаются.
`bitNot(a)`	Выполняет побитовую операцию "НЕ" (NOT) над числом `a`, инвертируя каждый бит. Тип результата совпадает с типом входного значения.
`bitOr(a, b)`	Выполняет побитовую операцию "ИЛИ" (OR) над двумя числами `a` и `b`. Тип результата совпадает с типом входных значений.
`bitShiftLeft(a, b)`	Сдвигает бинарное представление числа `a` на `b` бит влево. Поддерживаются числа и строки. Тип результата совпадает с типом сдвигаемого значения.
`bitShiftRight(a, b)`	Сдвигает бинарное представление числа `a` на `b` бит вправо. Поддерживаются числа и строки. Тип результата совпадает с типом сдвигаемого значения.

Функция	Описание
`bitTest(number, index)`	Возвращает значение бита на позиции `index` в числе `number`. Отсчет начинается с 0 справа налево. Возвращает `1`, если бит установлен, и `0`, если бит сброшен.
`bitTestAll(number, index1, index2, ⋯)`	Проверяет, все ли указанные биты в позициях `index1, index2, ⋯` установлены в 1 в числе `number`. Возвращает `1`, если все указанные биты установлены, и `0` в противном случае.
`bitTestAny(number, index1, index2, ⋯)`	Проверяет, установлен ли хотя бы один из указанных битов в позициях `index1, index2, ⋯` в числе `number` в 1. Возвращает `1`, если хотя бы один указанный бит установлен, и `0` в противном случае.
`bitXor(a, b)`	Выполняет побитовую операцию исключающее "ИЛИ" (XOR) над двумя числами `a` и `b`. Тип результата совпадает с типом входных значений.

Функции битовых карт

Битовые карты можно создавать следующими способами:

- с помощью агрегатной функции `groupBitmap` с суффиксом `-State`;
- путем построения битовой карты из массива.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции битовых карт

Функция	Описание
`bitmapAnd(a, b)`	Возвращает битовую карту, содержащую биты, установленные в обеих битовых картах `a` и `b`.
`bitmapAndCardinality(a, b)`	Возвращает количество битов в пересечении битовых карт `a` и `b`.
`bitmapAndnot(a, b)`	Возвращает битовую карту, содержащую биты, присутствующие в битовой карте `a`, но отсутствующие в битовой карте `b`.
`bitmapAndnotCardinality(a, b)`	Возвращает количество битов в разности битовых карт `a` и `b`.
`bitmapBuild(array)`	Возвращает битовую карту, построенную из массива `array`.
`bitmapCardinality(a)`	Возвращает количество установленных битов в битовой карте `a`.
`bitmapContains(a, b)`	Возвращает `1`, если битовая карта `a` содержит значение `b`, и `0` в противном случае.

Функция	Описание
`bitmapHasAll(a, b)`	Возвращает `1`, если все установленные биты битовой карты `b` присутствуют в битовой карте `a`, и `0` в противном случае.
`bitmapHasAny(a, b)`	Возвращает `1`, если хотя бы один бит битовой карты `b` присутствует в битовой карте `a`, и `0` в противном случае.
`bitmapMax(a)`	Возвращает позицию наибольшего установленного бита в битовой карте или `0`, если битовая карта пуста.
`bitmapMin(a)`	Возвращает позицию наименьшего установленного бита в битовой карте. Если все биты сброшены, функция возвращает: • `UINT32_MAX`, если битовая карта содержит до 2^{64} бит; • `UINT64_MAX`, если битовая карта содержит 2^{64} бит или больше.
`bitmapOr(a, b)`	Возвращает битовую карту, содержащую биты, установленные хотя бы в одной из битовых карт `a` и `b`.
`bitmapOrCardinality(a, b)`	Возвращает количество установленных единичных битов в объединении двух объектов битовых карт `a` и `b`.
`bitmapSubsetInRange(a, start, end)`	Возвращает битовую карту, содержащую биты битовой карты `a` в диапазоне от `start` (включительно) до `end` (исключительно). Индексация ведется с единицы.
`bitmapSubsetLimit(a, range_start, cardinality_limit)`	Возвращает битовую карту, содержащую биты битовой карты `a` начиная с позиции `range_start`, с количеством установленных битов не более, чем `cardinality_limit`. Индексация ведется с единицы.
`bitmapToArray(a)`	Возвращает массив беззнаковых целых чисел из битовой карты `a`.
`bitmapTransform(a, from_array, to_array)`	Возвращает битовую карту, преобразованную из битовой карты `a` путем замены значений битов из массива `from_array` соответствующими значениями из массива `to_array`.
`bitmapXor(a, b)`	Возвращает битовую карту, которая содержит биты, присутствующие только в одной из входных битовых карт `a` и `b`.
`bitmapXorCardinality(a, b)`	Возвращает количество установленных битов в симметрической разности битовых карт `a` и `b`.
`subBitmap(a, offset, limit)`	Возвращает битовую карту, содержащую биты битовой карты `a` начиная с позиции `offset`, с количеством установленных битов не более чем `limit`.

Функции округления

Функции округления в RQL предоставляют различные способы обработки числовых значений для достижения определенной степени точности или адаптации к специфическим требованиям аналитики.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

> Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.
>
> Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции округления

Функция	Описание
`ceil(x[, N])`, `ceiling(x[, N])`	Возвращает наименьшее круглое число, больше или равно `x`. `N` определяет точность округления.
`floor(x[, N])`	Возвращает наибольшее круглое число, меньше или равно `x`. `N` определяет точность округления.
`round(x[, N])`	Округляет число `x` до `N` знаков после запятой. При равном удалении от соседних чисел использует банковское округление для `Float32/64` и математическое для `Decimal`.
`roundAge(num)`	Округляет число `num` до ближайшего значения из набора, представляющего стандартные возрастные группы. Этот метод позволяет классифицировать числовые значения возраста в следующие группы: `17` (для значений от 1 до 17), `18` (для значений от 18 до 24), `25` (для значений от 25 до 34), `35` (для значений от 35 до 44), `45` (для значений от 45 до 54) и `55` (для значений старше 55).
`roundBankers(x[, N])`	Округляет число `x` до `N` знаков после запятой, используя банковское округление для всех типов чисел.
`roundDown(num, arr)`	Округляет число `num` вниз до ближайшего числа из предоставленного массива `arr`.
`roundDuration(num)`	Округляет число `num` до ближайшего значения из предопределенного набора чисел, обычно используемого для категоризации интервалов времени. Этот набор включает в себя следующие значения: `1`, `10`, `30`, `60`, `120`, `180`, `240`, `300`, `600`, `1200`, `1800`, `3600`, `7200`, `18000`, `36000`.
`roundToExp2(num)`	Округляет число `num` вниз до ближайшей степени двух.
`trunc(x[, N])`, `truncate(x[, N])`	Обрезает число `x` до `N` знаков после запятой без округления.

Функции для работы с датой и временем

Большинство функций в этом разделе принимают необязательный аргумент временной зоны, например, `Europe/Amsterdam`. В этом случае используется указанная временная зона, а не локальная (по умолчанию).

Пример

```
SELECT
    toDateTime('2016-06-15 23:00:00') AS time,
    toDate(time) AS date_local,
```

```
    toDate(time, 'Asia/Yekaterinburg') AS date_yekat,
    toString(time, 'US/Samoa') AS time_samoa
```

Визуализация ожидаемого результата запроса

time	date_local	date_yekat	time_samoa
2016-06-15 23:00:00	2016-06-15	2016-06-16	2016-06-15 09:00:00



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции для работы с датой и временем

Функция	Описание
`makeDate`	Создает объект `Date` из года, месяца и дня или из года и дня года.
`makeDateTime`	Создает объект `DateTime` из года, месяца, дня, часа, минуты и секунды.
`serverTimeZone`	Возвращает часовой пояс сервера.
`timestamp`	Преобразует первый аргумент в тип `DateTime64(6)`. Если предоставлен второй аргумент, добавляет указанное время к преобразованному значению.
`timeZone`	Возвращает часовой пояс текущего сеанса.
`timeZoneOf`	Возвращает имя часового пояса для типов данных `DateTime` или `DateTime64`.
`timeZoneOffset`	Возвращает смещение часового пояса в секундах от UTC.
`toDayOfMonth`	Возвращает номер дня в месяце (1—31) даты или даты со временем.
`toDayOfWeek`	Возвращает номер дня недели даты или даты со временем.
`toDayOfYear`	Возвращает номер дня в году (1—366) даты или даты со временем.
`toHour`	Возвращает часовой компонент (0—24) даты со временем.
`toLastDayOfMonth`	Округляет дату или дату со временем до последнего дня месяца.
`toLastDayOfWeek`	Округляет дату или дату со временем до конца недели, которая заканчивается в субботу или воскресенье.
`toMillisecond`	Возвращает миллисекундный компонент (0—999) даты со временем.
`toMinute`	Возвращает минутный компонент (0—59) даты со временем.
`toMonday`	Округляет дату или дату со временем до ближайшего понедельника.
`toMonth`	Возвращает месячный компонент (1—12) даты или даты со временем.

Функция	Описание
`toQuarter`	Возвращает квартал (1—4) даты или даты со временем.
`toSecond`	Возвращает секундный компонент (0—59) даты со временем. Високосные секунды не учитываются.
`toStartOfDay`	Округляет дату со временем до начала дня.
`toStartOfFifteenMinutes`	Округляет дату со временем до начала пятнадцатиминутного интервала.
`toStartOfFiveMinutes`	Округляет дату со временем до начала пятиминутного интервала.
`toStartOfHour`	Округляет дату со временем до начала часа.
`toStartOfInterval`	Обобщенная функция, аналогичная другим функциям `toStartOf*()`. Например, `toStartOfInterval(t, INTERVAL 1 year)` возвращает то же, что и `toStartOfYear(t)`.
`toStartOfISOYear`	Округляет дату или дату со временем до первого дня ISO года, который может отличаться от "обычного" года.
`toStartOfMinute`	Округляет дату со временем до начала минуты.
`toStartOfMonth`	Округляет дату или дату со временем до первого дня месяца.
`toStartOfQuarter`	Округляет дату или дату со временем до первого дня квартала.
`toStartOfSecond`	Округляет дату со временем, удаляя доли секунды.
`toStartOfTenMinutes`	Округляет дату со временем до начала десятиминутного интервала.
`toStartOfWeek`	Округляет дату или дату со временем до начала недели, которая начинается в воскресенье или понедельник.
`toStartOfYear`	Округляет дату или дату со временем до первого дня года.
`toTimeZone`	Конвертирует дату или дату со временем в указанный часовой пояс без изменения внутреннего значения.
`toUnixTimestamp`	Преобразует строку, дату или дату со временем в Unix Timestamp в представлении UInt32.
`toYear`	Возвращает годовой компонент (нашей эры) даты или даты со временем.

Функции для работы со строками

В RQL представлен набор функций, предназначенных для обработки текстовых данных. Функции для поиска в строках, замены в строках, разбиения и слияния строк и массивов рассмотрены отдельно.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse

может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции проверки и приведения длины строк

Функция	Описание
`empty`	Проверяет, является ли входная строка пустой. Возвращает `1` для пустой строки и `0` для непустой строки.
`leftPad`	Дополняет строку слева пробелами или заданной строкой (при необходимости несколько раз), пока результирующая строка не достигнет указанной длины.
`leftPadUTF8`	Дополняет строку слева пробелами или заданной строкой (при необходимости несколько раз) до достижения указанной длины. В отличие от `leftPad`, длина строки измеряется в кодовых точках.
`length`	Возвращает длину строки в байтах.
`lengthUTF8`	Возвращает длину строки в кодовых точках Unicode. Предполагается, что строка содержит допустимый текст в кодировке UTF-8.
`notEmpty`	Проверяет, не является ли входная строка пустой. Возвращает `1` для непустой строки и `0` для пустой строки.
`rightPad`	Дополняет строку справа пробелами или заданной строкой (при необходимости несколько раз) до достижения указанной длины.
`rightPadUTF8`	Дополняет строку справа пробелами или заданной строкой (при необходимости несколько раз) до достижения указанной длины. В отличие от `rightPad`, длина строки измеряется в кодовых точках.
`trim`	Удаляет пробельные символы с начала и конца строки.
`trimBoth`	Удаляет пробельные символы с обеих сторон строки.
`trimLeft`	Удаляет пробельные символы с начала строки.
`trimRight`	Удаляет пробельные символы с конца строки.

Функции преобразования регистра

Функция	Описание
`initcap`	Преобразует первый символ каждого слова в верхний регистр.
`initcapUTF8`	То же, что и `initcap`, но учитывает кодировку UTF-8.
`lower`	Преобразует символы ASCII латиницы в строке в нижний регистр.
`lowerUTF8`	Преобразует строку в нижний регистр, предполагая, что строка содержит допустимый текст в кодировке UTF-8.
`upper`	Преобразует символы ASCII латиницы в строке в верхний регистр.
`upperUTF8`	Преобразует строку в верхний регистр, предполагая, что строка содержит допустимый текст в кодировке UTF-8.

Функции конкатенации и инверсии строк

Функция	Описание
`appendTrailingCharIfAbsent`	Добавляет символ `c` к строке `s`, если `s` непустая и не заканчивается символом `c`.
`concat`	Конкатенирует заданные аргументы.
`concatAssumeInjective`	Аналогично `concat`, но предполагает, что `concat(s1, s2, ⋯) → sn` является инъективной. Может использоваться для оптимизации `GROUP BY`.
`concatWithSeparator`	Конкатенирует заданные строки с использованием заданного разделителя.
`concatWithSeparatorAssumeInjective`	Аналогично `concatWithSeparator`, но предполагает, что `concatWithSeparator(sep, expr1, expr2, expr3⋯) → result` является инъективной. Может использоваться для оптимизации `GROUP BY`.
`repeat`	Конкатенирует строку с самой собой указанное количество раз.
`reverse`	Инвертирует последовательность байтов в строке.
`reverseUTF8`	Инвертирует последовательность кодовых точек Unicode в строке. Предполагается, что строка содержит допустимый текст в кодировке UTF-8.
`space`	Конкатенирует пробел указанное количество раз с самим собой.

Функции поиска и извлечения подстрок

Функция	Описание
`endsWith`	Проверяет, заканчивается ли строка определенным суффиксом.
`endsWithUTF8`	То же, что и `endsWith`, но с учетом кодировки UTF-8.
`firstLine`	Возвращает первую строку из многострочного текста.
`startsWith`	Проверяет, начинается ли строка с определенного префикса.
`startsWithUTF8`	То же, что и `startsWith`, но с учетом кодировки UTF-8.
`substring`	Возвращает подстроку строки `s`, которая начинается с указанного индекса байта `offset`. Счет начинается с 1. Если `offset` равен 0, возвращается пустая строка. Если `offset` отрицательный, подстрока начинается с символов от конца строки.
`substringIndex`	Возвращает подстроку `s` до `count` вхождений разделителя `delim`, как в Spark или MySQL.
`substringIndexUTF8`	Аналогично `substringIndex`, но для кодовых точек Unicode. Предполагается, что строка содержит допустимый текст в кодировке UTF-8.
`substringUTF8`	Аналогично `substring`, но для кодовых точек Unicode. Предполагается, что строка содержит допустимый текст в кодировке UTF-8.

Функции валидации, нормализации и кодирования строк

Функция	Описание
ascii	Возвращает ASCII код первого символа строки.
base58Decode	Декодирует строку из формата Base58.
base58Encode	Кодирует строку в формат Base58.
base64Decode	Декодирует строку из формата Base64.
base64Encode	Кодирует строку в формат Base64.
convertCharset	Возвращает строку s, преобразованную из кодировки from в кодировку to.
decodeHTMLComponent	Декодирует HTML компоненты из строки.
decodeXMLComponent	Декодирует компоненты XML из строки.
encodeXMLComponent	Кодирует компоненты XML в строке.
extractTextFromHTML	Извлекает текст из HTML строки.
idnaDecode	Декодирует строку из формата IDNA.
idnaEncode	Кодирует строку в формат IDNA.
isValidUTF8	Возвращает 1, если набор байтов составляет допустимый текст, закодированный в UTF-8, в противном случае 0.
normalizeUTF8NFC	Нормализует строку в нормализацию формы C (NFC) для Unicode.
normalizeUTF8NFD	Нормализует строку в нормализацию формы D (NFD) для Unicode.
normalizeUTF8NFKC	Нормализует строку в нормализацию формы KC (NFKC) для Unicode.
normalizeUTF8NFKD	Нормализует строку в нормализацию формы KD (NFKD) для Unicode.
punycodeDecode	Декодирует строку из формата Punycode.
punycodeEncode	Кодирует строку в формат Punycode.
toValidUTF8	Заменяет недопустимые символы UTF-8 на символ  (U+FFFD). Все подряд идущие недопустимые символы сжимаются в один символ замены.
tryBase58Decode	Попытка декодировать строку из формата Base58 с возвратом пустой строки в случае ошибки.
tryBase64Decode	Попытка декодировать строку из формата Base64 с возвратом пустой строки в случае ошибки.
tryIdnaEncode	Попытка кодировать строку в формат IDNA с возвратом пустой строки в случае ошибки.
tryPunycodeDecode	Попытка декодировать строку из формата Punycode с возвратом пустой строки в случае ошибки.

Функции хеширования

Функция	Описание
`CRC32`	Вычисляет CRC32 хеш строки.
`CRC32IEEE`	Вычисляет CRC32 используя IEEE полином.
`CRC64`	Вычисляет CRC64 хеш строки.
`normalizedQueryHash`	Вычисляет хеш нормализованного запроса.
`normalizeQuery`	Нормализует строку запроса.

Функции оценки схожести строк

Функция	Описание
`byteHammingDistance`	Вычисляет расстояние Хэмминга между двумя строками.
`damerauLevenshteinDistance`	Вычисляет расстояние Дамерау-Левенштейна между двумя строками.
`editDistance`	Вычисляет расстояние редактирования (Левенштейна) между двумя строками.
`jaroSimilarity`	Вычисляет сходство Джаро между двумя строками.
`jaroWinklerSimilarity`	Вычисляет сходство Джаро-Винклера между двумя строками.
`soundex`	Преобразует строку в ее код Soundex.
`stringJaccardIndex`	Вычисляет индекс Жаккара для двух строк.
`stringJaccardIndexUTF8`	Вычисляет индекс Жаккара для двух строк, учитывая кодировку UTF-8.

Функции разбиения и слияния строк и массивов

Функции разбиения и слияния строк и массивов в RQL предназначены для манипуляции текстовыми данными и коллекциями. Эти функции позволяют разделять строки на подстроки или элементы массива с использованием различных критериев, таких как фиксированные символы, строки, регулярные выражения или типы символов. Также они обеспечивают возможность объединения элементов массива в единую строку с заданным разделителем.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции разбиения и слияния строк и массивов

Функция	Описание
`alphaTokens(s)`	Выделяет из строки `s` подстроки, состоящие из буквенных символов, и возвращает их в виде массива.

Функция	Описание
`arrayStringConcat(arr[, separator])`	Склеивает элементы массива `arr` в строку, используя `separator` в качестве разделителя.
`ngrams(s, n)`	Разбивает строку `s` на n-граммы (последовательности из `n` символов) и возвращает их в виде массива.
`splitByChar(separator, s)`	Разбивает строку `s` на массив подстрок, используя односимвольный разделитель `separator`.
`splitByNonAlpha(s)`	Разбивает строку `s` на массив подстрок, используя неалфавитные символы в качестве разделителей.
`splitByRegexp(regexp, s)`	Разбивает строку `s` на массив подстрок, используя регулярное выражение `regexp` в качестве разделителя.
`splitByString(separator, s)`	Разбивает строку `s` на массив подстрок, используя строку `separator` в качестве разделителя.
`splitByWhitespace(s)`	Разбивает строку `s` на массив подстрок, используя пробельные символы в качестве разделителей.
`tokens(s)`	Разбивает строку `s` на токены, используя не буквенно-цифровые символы ASCII в качестве разделителей.

Функции для поиска в строках

Все функции в этом разделе по умолчанию осуществляют поиск с учетом регистра. Поиск без учета регистра обычно предоставляется отдельными вариантами функций. Обратите внимание, что поиск без учета регистра следует правилам перевода в нижний и верхний регистры английского языка. Например, прописной букве `i` в английском языке соответствует заглавная `I`, тогда как в турецком языке — заглавная `İ`. Результаты для языков, отличных от английского, могут быть неожиданными.

Функции в этом разделе также предполагают, что искомая строка и строка поиска кодируются текстом с однобайтовой кодировкой. В случае отклонения от этого предположения, система не генерирует исключений, и результат поиска остается неопределенным. Для работы со строками, закодированными в UTF-8, обычно предоставляются специализированные версии функций. Аналогично, использование функций, предназначенных для работы с UTF-8, с текстом, не закодированным в UTF-8, не приводит к генерации исключений, и результаты также остаются неопределенными. Обратите внимание, что автоматическая нормализация Unicode не выполняется, для этого можно использовать функции `normalizeUTF8*()`.

Общие функции для работы со строками и функции для замены в строках описаны отдельно.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит

определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции поиска в строках

Функция	Описание
`countMatches(haystack, pattern)`	Возвращает количество совпадений регулярного выражения в строке.
`countSubstrings(haystack, needle)`	Возвращает количество вхождений подстроки в строку.
`extract(haystack, pattern)`	Извлекает фрагмент строки, соответствующий регулярному выражению.
`extractAll(haystack, pattern)`	Извлекает все фрагменты строки, соответствующие регулярному выражению.
`extractAllGroupsHorizontal(haystack, pattern)`	Возвращает массив массивов, где каждый массив содержит фрагменты строки, соответствующие каждой группе в регулярном выражении.
`extractAllGroupsVertical(haystack, pattern)`	Возвращает массив массивов, где каждый внутренний массив содержит фрагменты, соответствующие группам в порядке их появления в строке.
`hasSubsequence(haystack, needle)`	Возвращает `1`, если подстрока является подпоследовательностью строки, иначе `0`.
`ilike(haystack, pattern)`	Возвращает `true`, если строка соответствует шаблону выражения `LIKE` без учета регистра.
`like(haystack, pattern)`	Возвращает `true`, если строка соответствует шаблону выражения `LIKE`.
`match(haystack, pattern)`	Возвращает `true`, если строка соответствует регулярному выражению в синтаксисе регулярных выражений re2.
`multiFuzzyMatchAny(haystack, distance, [pattern1, pattern2, ⋯, patternN])`	Проверяет соответствие строки хотя бы одному из регулярных выражений с учетом допустимого расстояния редактирования.
`multiMatchAllIndices(haystack, [pattern1, pattern2, ⋯, patternN])`	Возвращает массив индексов всех регулярных выражений, которым соответствует строка.
`multiMatchAny(haystack, [pattern1, pattern2, ⋯, patternN])`	Проверяет соответствие строки хотя бы одному из регулярных выражений.
`multiMatchAnyIndex(haystack, [pattern1, pattern2, ⋯, patternN])`	Возвращает индекс регулярного выражения, которому соответствует строка.
`multiSearchAllPositions(haystack, [needle1, needle2, ⋯, needleN])`	Возвращает массив позиций для множества подстрок в строке.
`multiSearchAllPositionsUTF8(haystack, [needle1, needle2, ⋯, needleN])`	Поиск всех позиций множества подстрок в UTF-8 строке.

Функция	Описание
`multiSearchAny(haystack, [needle1, needle2, ⋯, needleN])`	Возвращает 1, если хотя бы одна из указанных подстрок найдена в строке, иначе 0.
`multiSearchFirstIndex(haystack, [needle1, needle2, ⋯, needleN])`	Возвращает индекс первой найденной подстроки из списка в строке.
`multiSearchFirstPosition(haystack, [needle1, needle2, ⋯, needleN])`	Возвращает наименьшую позицию среди всех указанных подстрок в строке.
`ngramDistance(haystack, needle)`	Вычисляет расстояние между строками на основе 4-грамм.
`ngramSearch(haystack, needle)`	Похоже на `ngramDistance`, но вычисляет несимметричную разницу между строками.
`notILike(haystack, pattern)`	Возвращает `true`, если строка не соответствует шаблону выражения `LIKE` без учета регистра.
`notLike(haystack, pattern)`	Возвращает `true`, если строка не соответствует шаблону выражения `LIKE`.
`position(haystack, needle[, start_pos])`	Возвращает позицию (в байтах, начиная с 1) подстроки `needle` в строке `haystack`.
`positionCaseInsensitive(haystack, needle[, start_pos])`	Поиск позиции подстроки в строке без учета регистра.
`positionCaseInsensitiveUTF8(haystack, needle[, start_pos])`	Поиск позиции подстроки в UTF-8 строке без учета регистра.
`positionUTF8(haystack, needle[, start_pos])`	Поиск позиции подстроки в UTF-8 строке.
`regexpExtract(haystack, pattern[, index])`	Извлекает подстроку, соответствующую индексу группы в регулярном выражении.



Шаблон может включать обычные символы и следующие метасимволы:

- `%` — обозначает любое количество произвольных символов или отсутствие символов.
- `_` — обозначает один произвольный символ.
- `\` — используется для экранирования символов `%`, `_` и `\`.

Функции для замены строк

Функции для замены строк в RQL обеспечивают возможности поиска и замены подстрок в текстовых данных. Они позволяют изменять содержимое строк путем замены одних подстрок на другие, как на основе простого текстового сопоставления, так и с использованием регулярных выражений.

Функции для замены строк в RQL предоставляют возможности для модификации текстовых данных

путем замены определенных подстрок или символов на другие. Эти функции обеспечивают простые и сложные замены, включая работу с регулярными выражениями и прямое отображение символов для точной настройки результатов замены.

Общие функции для работы со строками и функции для замены в строках описаны отдельно.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции для замены строк

Функция	Описание
`replaceAll(source, pattern, replacement)`	Заменяет все вхождения подстроки `pattern` в строке `source` на строку `replacement`. Псевдоним: `replace`.
`replaceOne(source, pattern, replacement)`	Заменяет первое вхождение подстроки `pattern` в строке `source` на строку `replacement`.
`replaceRegexpAll(source, pattern, replacement)`	Аналогично `replaceRegexpOne`, но заменяет все вхождения подстрок, соответствующих регулярному выражению `pattern`. Псевдоним: `REGEXP_REPLACE`.
`replaceRegexpOne(source, pattern, replacement)`	Заменяет первое вхождение подстроки, соответствующей регулярному выражению `pattern` (в синтаксисе re2), в строке `source` на строку `replacement`. В `replacement` могут использоваться подстановки `\0-\9`, где `\1-\9` соответствуют 1-й по 9-й захваченной группе, а `\0` соответствует всему совпадению.
`translate(s, from, to)`	Заменяет символы в строке `s`, используя прямое отображение символов, определенное строками `from` и `to`. Строки `from` и `to` должны быть константными ASCII строками одинакового размера. Не-ASCII символы в исходной строке не модифицируются.
`translateUTF8(string, from, to)`	Аналогично `translate`, но предполагает, что строки `s`, `from` и `to` закодированы в UTF-8.

Функции для работы с массивами

Функции для работы с массивами в RQL предоставляют набор операций для манипуляции массивами данных. Эти функции позволяют выполнять проверку на пустоту или заполненность массива, определять его длину, создавать пустые массивы различных типов, генерировать числовые диапазоны, объединять массивы, извлекать элементы, а также проводить более сложные операции, такие как вычисление уникальности элементов, работу с подмассивами и манипуляцию порядком элементов.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции для работы с массивами

Функция	**Описание**
`array(x1[, x2, ⋯, xN])`	Создает массив из аргументов функции. Аргументы должны быть константами и иметь типы, для которых есть наименьший общий тип. Должен быть передан хотя бы один аргумент для определения типа массива. Возвращает результат типа `Array(T)`, где `T` — наименьший общий тип от переданных аргументов.
`arrayAUC(arr_scores, arr_labels)`	Вычисляет площадь под кривой (AUC) для заданных массивов оценок `arr_scores` и меток `arr_labels`.
`arrayCompact(array)`	Удаляет последовательно повторяющиеся элементы из массива `array`, сохраняя только уникальные последовательности. Возвращает новый массив без последовательных дубликатов.
`arrayConcat(arrays)`	Объединяет массивы, переданные в качестве аргументов, в один массив. Пример: `SELECT arrayConcat([1, 2], [3, 4], [5, 6])` возвращает `[1,2,3,4,5,6]`.
`arrayDifference(array)`	Вычисляет разность между соседними элементами массива `array`. Возвращает массив, где первый элемент равен нулю, а каждый следующий элемент равен разности между текущим и предыдущим элементами исходного массива.
`arrayDistinct(array)`	Возвращает массив, содержащий только уникальные элементы из исходного массива `array`.
`arrayElement(arr, n)`	Извлекает элемент с индексом `n` из массива `arr`. Индексы начинаются с единицы. Поддерживаются отрицательные индексы для доступа к элементам с конца массива. Если индекс выходит за границы массива, то возвращается значение по умолчанию для типа элементов массива.
`arrayEnumerate(arr)`	Возвращает массив индексов для каждого элемента входного массива `arr`. Это позволяет идентифицировать позицию каждого элемента в исходном массиве.
`arrayEnumerateDense(arr)`	Возвращает массив, каждый элемент которого указывает на позицию первого вхождения соответствующего элемента в массиве `arr` с учетом его встречаемости.

Функция	Описание
`arrayEnumerateUniq(arr, ...)`	Возвращает массив, где каждый элемент указывает на его первое вхождение в массиве `arr`. Если функция принимает несколько массивов, уникальность определяется по комбинации элементов на соответствующих позициях во всех массивах.
`arrayFlatten(array_of_arrays)`	Преобразует массив массивов `array_of_arrays` в один плоский массив, содержащий все элементы исходных массивов. Элементы вложенных массивов добавляются в результирующий массив последовательно, начиная с первого вложенного массива до последнего.
`arrayIntersect(arr)`	Возвращает массив, содержащий элементы, которые присутствуют во всех переданных массивах `arr`.
`arrayJoin(arr)`	Специальная функция, которая используется для разделения одного массива `arr` на строки, каждая из которых содержит один элемент массива.
`arrayPopBack(array)`	Удаляет последний элемент из массива `array` и возвращает новый массив без этого элемента.
`arrayPopFront(array)`	Удаляет первый элемент из массива `array` и возвращает новый массив без этого элемента.
`arrayProduct(arr)`	Возвращает произведение элементов массива `arr`.
`arrayPushBack(array, single_value)`	Добавляет элемент `single_value` в конец массива `array` и возвращает новый массив с добавленным элементом.
`arrayPushFront(array, single_value)`	Добавляет элемент `single_value` в начало массива `array` и возвращает новый массив с добавленным элементом.
`arrayResize(array, size[, extender])`	Изменяет размер массива `array` до нового размера `size`, используя `extender` для заполнения новых элементов, если `size` больше текущего размера массива.
`arrayReverse(arr)`	Возвращает массив с элементами исходного массива `arr` в обратном порядке.
`arrayRotateLeft(arr, n)`	Циклически сдвигает элементы массива `arr` влево на `n` позиций.
`arrayRotateRight(arr, n)`	Циклически сдвигает элементы массива `arr` вправо на `n` позиций.
`arrayShiftLeft(arr, n)`	Сдвигает элементы массива `arr` влево на `n` позиций, заполняя освободившиеся места нулями.
`arrayShiftRight(arr, n)`	Сдвигает элементы массива `arr` вправо на `n` позиций, заполняя освободившиеся места нулями.
`arraySlice(array, offset[, length])`	Возвращает часть массива `array`, начиная с позиции `offset` и длиной `length`. Если `length` не указано, возвращает все элементы начиная с `offset` до конца массива.
`arrayUniq(arr, ...)`	Возвращает массив уникальных элементов из массива `arr`. Если передано несколько массивов, возвращает массив уникальных комбинаций элементов на соответствующих позициях в этих массивах.

Функция	Описание
`arrayZip(arr1, arr2, ⋯)`	Объединяет несколько массивов `arr1`, `arr2`, ..., создавая массив кортежей, где каждый кортеж содержит элементы из исходных массивов, расположенные на одинаковых позициях. Длина результирующего массива соответствует длине самого короткого из исходных массивов.
`countEqual(arr, x)`	Возвращает количество элементов массива, равных `x`. Эквивалентно `arrayCount(elem → elem = x, arr)`. `NULL` обрабатывается как значение. Пример: `SELECT countEqual([1, 2, NULL, NULL], NULL)` возвращает `2`.
`empty(arr)`	Проверяет, является ли входной массив `arr` пустым. Массив считается пустым, если он не содержит ни одного элемента. Функция также поддерживает работу с типами `String` и `UUID`.
`emptyArrayDate()`	Возвращает пустой массив с элементами типа `Date`.
`emptyArrayDateTime()`	Возвращает пустой массив с элементами типа `DateTime`.
`emptyArrayFloat32()`	Возвращает пустой массив с элементами типа `Float32`.
`emptyArrayFloat64()`	Возвращает пустой массив с элементами типа `Float64`.
`emptyArrayInt16()`	Возвращает пустой массив с элементами типа `Int16`.
`emptyArrayInt32()`	Возвращает пустой массив с элементами типа `Int32`.
`emptyArrayInt64()`	Возвращает пустой массив с элементами типа `Int64`.
`emptyArrayInt8()`	Возвращает пустой массив с элементами типа `Int8`.
`emptyArrayString()`	Возвращает пустой массив строк.
`emptyArrayToSingle(arr)`	Принимает пустой массив `arr` и возвращает массив из одного элемента, равного значению по умолчанию для типа элементов `arr`.
`emptyArrayUInt16()`	Возвращает пустой массив с элементами типа `UInt16`.
`emptyArrayUInt32()`	Возвращает пустой массив с элементами типа `UInt32`.
`emptyArrayUInt64()`	Возвращает пустой массив с элементами типа `UInt64`.
`emptyArrayUInt8()`	Возвращает пустой массив с элементами типа `UInt8`.
`has(arr, elem)`	Проверяет наличие элемента `elem` в массиве `arr`. Возвращает `0`, если элемента в массиве нет, или `1`, если есть. `NULL` обрабатывается как значение. Пример: `SELECT has([1, 2, NULL], NULL)` возвращает `1`.
`indexOf(arr, x)`	Возвращает индекс первого вхождения элемента `x` в массив `arr`, начиная с `1`. Если элемент отсутствует в массиве, возвращает `0`. Элементы, равные `NULL`, обрабатываются как обычные значения.
`length(arr)`	Возвращает количество элементов в массиве `arr`. Тип результата — `UInt64`. Функция также работает для строк. Можно оптимизировать работу функции с использованием настройки `optimize_functions_to_subcolumns`.
`notEmpty(arr)`	Проверяет, является ли входной массив `arr` непустым. Массив считается непустым, если он содержит хотя бы один элемент. Функция также поддерживает работу с типами `String` и `UUID`.

Функция	Описание
`range([start, ] end [, step])`	Возвращает массив чисел от `start` до `end - 1` с шагом `step`. Аргументы `start` и `step` необязательны. По умолчанию `start = 0`, `step = 1`. Аргумент `end` обязателен. Если аргументы `start`, `end`, `step` имеют отрицательные значения или значение аргумента `end` меньше значения аргумента `start`, функция генерирует исключение. Если суммарная длина создаваемых массивов больше значения настройки `function_range_max_elements_in_block`, функция генерирует исключение.
`reverse(arr)`	Синоним функции `arrayReverse`. Возвращает массив с элементами исходного массива `arr` в обратном порядке.

Географические функции

Функции географических координат

Функции географических координат в RQL предназначены для обработки и анализа геопространственных данных. Они позволяют выполнять расчеты расстояний между точками на земной поверхности, определять принадлежность точек к геометрическим фигурам, таким как эллипсы и многоугольники, и вычислять углы между точками.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции географических координат

Функция	Описание
`geoDistance`	Вычисляет расстояние между двумя точками на земной поверхности.
`greatCircleAngle`	Вычисляет угол между двумя точками на поверхности земли по большому кругу.
`greatCircleDistance`	Расчет расстояния по большому кругу между двумя точками на поверхности земли.
`pointInEllipses`	Определяет, находится ли точка внутри заданных эллипсов.
`pointInPolygon`	Проверяет, находится ли заданная точка внутри многоугольника.

Функции Geohash

Geohash — это система геокодирования, которая делит поверхность Земли на сектора в форме сетки и кодирует каждую ячейку короткой строкой из букв и цифр. Это иерархическая структура данных,

поэтому чем длиннее строка geohash, тем точнее географическое местоположение.

Если вам необходимо вручную конвертировать географические координаты в строки geohash, вы можете использовать geohash.org.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции Geohash

Функция	Описание
`geohashDecode`	Декодирует строку Geohash обратно в географические координаты.
`geohashEncode`	Кодирует географические координаты в строку Geohash.
`geohashesInBox`	Генерирует Geohash-коды, покрывающие прямоугольник, заданный двумя угловыми точками.

Функции индексов H3

H3 — это географическая система индексации, в которой поверхность Земли разделена на сетку равных шестиугольных ячеек. Эта система является иерархической, то есть каждый шестиугольник на верхнем уровне ("родитель") может быть разделен на семь равных, но меньших ("дети").

Уровень иерархии называется разрешением и может принимать значения от 0 до 15, где 0 — это базовый уровень с самыми крупными и грубыми ячейками.

Пара широты и долготы может быть преобразована в 64-битный индекс H3, идентифицирующий ячейку сетки.

Индекс H3 в основном используется для группировки местоположений и других геопространственных манипуляций.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции индексов H3

Функция	Описание
`geoToH3`	Преобразует географические координаты в индекс H3.
`h3CellAreaM2`	Возвращает площадь ячейки H3 в квадратных метрах.
`h3CellAreaRads2`	Возвращает площадь ячейки H3 в квадратны х радианах.
`h3Distance(start,end)`	Возвращает расстояние в ячейках между индексами H3 `start` и `end`.
`h3EdgeAngle`	Вычисляет угол ребра индекса H3.
`h3EdgeLengthKm`	Вычисляет длину ребра индекса H3 в километрах.
`h3EdgeLengthM`	Вычисляет длину ребра индекса H3 в метрах.
`h3ExactEdgeLengthKm`	Возвращает точную длину ребра индекса H3 в километрах.
`h3ExactEdgeLengthM`	Возвращает точную длину ребра индекса H3 в метрах.
`h3ExactEdgeLengthRads`	Возвращает точную длину ребра индекса H3 в радианах.
`h3GetBaseCell`	Возвращает базовую ячейку для данного индекса H3.
`h3GetDestinationIndexFromUnidirectionalEdge`	Возвращает конечный индекс для однонаправленного ребра.
`h3GetFaces`	Возвращает список граней, которые пересекает ячейка, представленная индексом H3.
`h3GetIndexesFromUnidirectionalEdge`	Возвращает оба индекса для однонаправленного ребра.
`h3GetOriginIndexFromUnidirectionalEdge`	Возвращает исходный индекс для однонаправленного ребра.
`h3GetResolution`	Возвращает разрешение (уровень детализации) индекса H3.
`h3GetUnidirectionalEdge`	Возвращает однонаправленное ребро между двумя индексами H3.
`h3GetUnidirectionalEdgeBoundary`	Возвращает границы однонаправленного ребра.
`h3GetUnidirectionalEdgesFromHexagon`	Возвращает все однонаправленные ребра для заданной ячейки.
`h3HexAreaKm2`	Возвращает площадь ячейки, представленной индексом H3, в квадратных километрах.
`h3HexAreaM2`	Возвращает площадь ячейки, представленной индексом H3, в квадратных метрах.
`h3HexRing(index, k)`	Возвращает индексы кольца ячеек на расстоянии `k` от индекса H3 `index`.
`h3IndexesAreNeighbors`	Проверяет, являются ли два индекса H3 соседями.
`h3IsPentagon`	Проверяет, является ли индекс H3 пятиугольной ячейкой.
`h3IsResClassIII`	Определяет, принадлежит ли индекс H3 к третьему классу разрешения.
`h3IsValid`	Проверяет, является ли заданный индекс H3 действительным.
`h3kRing`	Возвращает набор индексов H3, составляющих кольцо вокруг заданной точки.

Функция	Описание
`h3Line`	Возвращает линию (последовательность индексов H3) между двумя индексами H3.
`h3NumHexagons`	Возвращает количество уникальных индексов H3 в указанном разрешении.
`h3ToCenterChild`	Возвращает центральный дочерний индекс для заданного индекса H3.
`h3ToChildren`	Возвращает дочерние индексы для заданного индекса H3 и указанного разрешения.
`h3ToGeo`	Преобразует индекс H3 обратно в географические координаты.
`h3ToGeoBoundary`	Возвращает границы ячейки, представленной индексом H3.
`h3ToParent`	Возвращает родительский индекс для заданного индекса H3 и указанного разрешения.
`h3ToString`	Преобразует индекс H3 в строковое представление.
`h3UnidirectionalEdgeIsValid`	Проверяет, является ли однонаправленное ребро действительным.
`stringToH3`	Преобразует строковое представление индекса H3 обратно в индекс.

Функции индексов S2

S2 — это система географической индексации, в которой все географические данные представлены на трехмерной сфере, аналогично глобусу.

В библиотеке S2 точки представлены как S2 Index — определенное число, которое внутренне кодирует точку на поверхности единичной сферы, в отличие от традиционных пар широты и долготы. Для получения индекса точки S2 для данной точки, указанной в формате "широта, долгота", используйте функцию `geoToS2`. Также вы можете использовать функцию `s2ToGeo` для получения географических координат, соответствующих указанному индексу точки S2.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции индексов S2

Функция	Описание
`geoToS2(lon, lat)`	Преобразует географические координаты с широтой `lon` и долготой `lat` в индекс S2.

Функция	Описание
`s2CapContains`	Проверяет, содержит ли сферический сегмент заданную точку или ячейку.
`s2CapUnion`	Объединяет два сферических сегмента.
`s2CellsIntersect`	Проверяет пересечение двух ячеек S2.
`s2GetNeighbors`	Возвращает индексы S2 соседних ячеек.
`s2RectAdd`	Добавляет точку к прямоугольнику S2, потенциально расширяя его.
`s2RectContains`	Проверяет, содержит ли прямоугольник S2 заданную точку.
`s2RectIntersection`	Находит пересечение двух прямоугольников S2.
`s2RectUnion`	Объединяет два прямоугольника S2.
`s2ToGeo`	Преобразует индекс S2 обратно в географические координаты.

Функции временных рядов

В RQL доступны специализированные функции для анализа и обработки данных временных рядов. Эти функции обеспечивают возможности для обнаружения выбросов, определения периодичности и декомпозиции временных рядов на составляющие, такие, как сезонность и тренд.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции временных рядов

Функция	Описание
`seriesDecomposeSTL(series, period)`	Декомпозиция временного ряда на сезонную, трендовую и остаточную компоненты с использованием STL (Seasonal and Trend decomposition using Loess). Принимает массив числовых значений `series` и положительное целое число `period`, указывающее период временного ряда. Количество точек данных в `series` должно быть как минимум в два раза больше значения `period`. Возвращает массив из четырех массивов, где первый массив включает сезонные компоненты, второй — тренд, третий — остаточную компоненту, и четвертый — базовую компоненту (сезонная + тренд).

Функция	Описание
`seriesOutliersDetectTukey(series, [min_percentile, max_percentile, K])`	Обнаруживает выбросы в данных временных рядов, используя ограды Тьюки. Принимает массив числовых значений `series`, минимальный и максимальный процентили `min_percentile` и `max_percentile` для расчета межквартильного размаха (IQR) и константу `K` для обнаружения умеренных или более сильных выбросов. По умолчанию `min_percentile` = 0.25, `max_percentile` = 0.75, `K` = 1.5. Требуется минимум четыре точки данных в `series` для обнаружения выбросов. Возвращает массив той же длины, что и входной, где каждое значение представляет собой оценку возможной аномалии соответствующего элемента в ряду.
`seriesPeriodDetectFFT(series)`	Находит период данных временного ряда с использованием быстрого преобразования Фурье (FFT). Принимает массив числовых значений `series`. Возвращает действительное значение, равное периоду данных временного ряда, или NAN, если количество точек данных меньше четырех.

Функции для работы с IP-адресами

Функции для работы с IP-адресами в RQL предназначены для выполнения различных операций преобразования и анализа с IPv4 и IPv6 адресами.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции для работы с IP-адресами

Функция	Описание
`cutIPv6(x, bytesToCutForIPv6, bytesToCutForIPv4)`	Принимает значение `FixedString(16)`, содержащее IPv6-адрес в бинарном формате, и возвращает строку с адресом с убранным указанным количеством байт.
`IPv4CIDRToRange(ipv4, Cidr)`	Принимает IPv4 и значение `UInt8`, содержащее CIDR, и возвращает кортеж из двух IPv4, содержащих нижний и верхний диапазон сети.
`IPv4NumToString(num)`	Принимает число `UInt32`, интерпретирует его как IPv4-адрес в `big-endian` и возвращает строку с соответствующим IPv4-адресом в формате `A.B.C.D`.
`IPv4NumToStringClassC(num)`	Похожа на `IPv4NumToString`, но заменяет последний октет на xxx.
`IPv4StringToNum(s)`	Обратная функция к `IPv4NumToString`. В случае неверного формата IPv4-адреса генерирует исключение.

Функция	Описание
`IPv4StringToNumOrDefault(s)`	Аналогично `IPv4StringToNum`, но возвращает 0 при неверном формате IPv4-адреса.
`IPv4StringToNumOrNull(s)`	Аналогично `IPv4StringToNum`, но возвращает `null` при неверном формате IPv4-адреса.
`IPv4ToIPv6(x)`	Принимает число `UInt32`, интерпретирует его как IPv4-адрес в `big-endian` и возвращает значение `FixedString(16)`, содержащее IPv6-адрес в бинарном формате.
`IPv6CIDRToRange(ipv6, Cidr)`	Принимает IPv6 и значение `UInt8`, содержащее `CIDR`, и возвращает кортеж из двух IPv6, содержащих нижний и верхний диапазон сети.
`IPv6NumToString(x)`	Принимает значение `FixedString(16)`, содержащее IPv6-адрес в бинарном формате, и возвращает строку с этим адресом в текстовом формате.
`IPv6StringToNum(s)`	Обратная функция к `IPv6NumToString`. В случае неверного формата IPv6-адреса генерирует исключение.
`IPv6StringToNumOrDefault(s)`	Аналогично `IPv6StringToNum`, но возвращает 0 при неверном формате IPv6-адреса.
`IPv6StringToNumOrNull(s)`	Аналогично `IPv6StringToNum`, но возвращает `null` при неверном формате IPv6-адреса.
`isIPAddressInRange`	Определяет, содержится ли IP-адрес в сети, представленной в нотации CIDR. Возвращает 1, если это так, и 0 в противном случае.
`isIPv4String`	Определяет, является ли входная строка IPv4-адресом. Возвращает 1, если строка является IPv4-адресом, и 0 в противном случае.
`isIPv6String`	Определяет, является ли входная строка IPv6-адресом. Возвращает 1, если строка является IPv6-адресом, и 0 в противном случае.
`toIPv4(string)`	Является псевдонимом `IPv4StringToNum()`, принимает строковую форму IPv4-адреса и возвращает значение типа IPv4, бинарно равное значению, возвращаемому `IPv4StringToNum()`.
`toIPv4OrDefault(string)`	Аналогично `toIPv4`, но возвращает 0.0.0.0 (IPv4 типа 0) при неверном формате IPv4-адреса.
`toIPv4OrNull(string)`	Аналогично `toIPv4`, но возвращает `null` при неверном формате IPv4-адреса.
`toIPv6(string)`	Конвертирует строковую форму IPv6-адреса в тип IPv6. В случае неверного формата IPv6-адреса возвращает пустое значение.
`toIPv6OrDefault(string)`	Аналогично `toIPv6`, но возвращает `::` (IPv6 типа 0) при неверном формате IPv6-адреса.
`toIPv6OrNull(string)`	Аналогично `toIPv6`, но возвращает `null` при неверном формате IPv6-адреса.

Функции для работы с URL

Функции для работы с URL предназначены для анализа и манипуляции URL-адресами. Они

позволяют извлекать компоненты URL, такие, как протокол, домен, порт, путь, параметры запроса, а также проводить различные операции по преобразованию и очистке URL-адресов.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции для работы с URL

Функция	Описание
`cutFragment(URL)`	Удаляет fragment из URL.
`cutQueryString(URL)`	Удаляет query string из URL.
`cutQueryStringAndFragment(URL)`	Удаляет query string и fragment из URL.
`cutToFirstSignificantSubdomain(URL)`	Возвращает часть домена до первого существенного поддомена.
`cutToFirstSignificantSubdomainCustom(URL, TLD)`	Возвращает часть домена до первого существенного поддомена с использованием пользовательского списка доменов верхнего уровня.
`cutToFirstSignificantSubdomainCustomWithWWW(URL, TLD)`	То же самое, но не удаляет `www`.
`cutURLParameter(URL, name)`	Удаляет параметр с именем `name` из URL.
`cutWWW(URL)`	Удаляет `www.` из начала домена в URL.
`decodeURLComponent(URL)`	Возвращает декодированный URL.
`domain(URL)`	Извлекает имя хоста из URL.
`domainWithoutWWW(URL)`	Возвращает домен, удалив префикс `www.`, если он присутствовал.
`extractURLParameter(URL, name)`	Извлекает значение параметра `name` из URL.
`extractURLParameterNames(URL)`	Возвращает массив имен параметров из URL.
`extractURLParameters(URL)`	Возвращает массив параметров из URL.
`firstSignificantSubdomain(URL)`	Возвращает первый существенный поддомен из URL.
`firstSignificantSubdomainCustom(URL, TLD)`	Возвращает первый существенный поддомен с использованием пользовательского списка доменов верхнего уровня.
`fragment(URL)`	Возвращает fragment из URL.
`netloc(URL)`	Извлекает сетевую локальность (`username:password@host:port`) из URL.

Функция	Описание
`path(URL)`	Возвращает путь из URL.
`pathFull(URL)`	Возвращает путь, включая query string и fragment из URL.
`port(URL[, default_port = 0])`	Возвращает порт из URL или значение `default_port`, если порт не указан.
`protocol(URL)`	Возвращает протокол из URL.
`queryString(URL)`	Возвращает query string из URL.
`queryStringAndFragment(URL)`	Возвращает query string и fragment из URL.
`topLevelDomain(URL)`	Извлекает домен верхнего уровня из URL.
`URLHierarchy(URL)`	Возвращает массив с URL, обрезанный по `/`, `?`.
`URLPathHierarchy(URL)`	Возвращает массив путей из URL, обрезанный по `/`.

Функции для работы с UUID

Функции для работы с UUID обеспечивают создание, преобразование и проверку уникальных идентификаторов, применяемых для обеспечения уникальности объектов в базах данных, системах и приложениях. Эти функции позволяют генерировать UUID четвертой версии, проверять UUID на пустоту или наличие значений, а также осуществлять преобразования между строковыми и двоичными представлениями UUID. Кроме того, доступна функция для получения уникального идентификатора сервера, что может быть использовано для идентификации источника данных или обеспечения трассировки в распределенных системах.



Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL.

Ознакомиться с актуальным списком функций и их поведением можно в официальной документации ClickHouse.

Обратите внимание, что для функций в официальной документации ClickHouse может быть указана версия, в которой эти функции были введены. Это позволит определить, доступна ли рассматриваемая функция в вашей версии ClickHouse.

Функции для работы с UUID

Функция	Описание
`empty(UUID)`	Проверяет, является ли входной UUID пустым, то есть состоит только из нулей. Работает также с массивами и строками. Возвращает `1` для пустого UUID и `0` для непустого.
`generateUUIDv4([x])`	Генерирует UUID четвертой версии. Необязательный параметр `x` используется для обхода удаления общих подвыражений, если функция вызывается несколько раз в одном запросе. Возвращает значение типа UUID.

Функция	Описание
`notEmpty(UUID)`	Проверяет, не является ли входной UUID пустым. Возвращает `1` для непустого UUID и `0` для пустого.
`serverUUID()`	Возвращает случайный и уникальный UUID сервера, который генерируется при первом запуске и сохраняется навсегда. Возвращает UUID сервера.
`UUIDNumToString(binary[, variant = 1])`	Принимает двоичное представление UUID и возвращает строку из 36 символов в текстовом формате. Вариант формата может быть указан (по умолчанию Big-endian).
`UUIDStringToNum(uuid[, variant = 1])`	Преобразует строковое представление UUID в двоичное, с указанием варианта формата (по умолчанию big-endian). Возвращает значение типа `FixedString(16)`.

Справочник по языку преобразования данных VRL

В данном разделе представлен справочник по языку преобразования данных VRL (Vector Remap Language), который применяется в схемах RObject для трансформации, фильтрации и обогащения данных событий в системе.

Содержание раздела:

- О языке VRL
- Выражения
  - Пустое значение
  - Логическое значение
  - Целое число
  - Число с плавающей точкой
  - Метка времени
  - Строка
  - Регулярное выражение
  - Массив
  - Объект
  - Переменная
  - Путь
  - Индекс
  - Присваивание
  - Арифметическое выражение
  - Сравнение
  - Логическое выражение
  - Вызов функции
  - Объединение
  - Условие
  - Блок
  - Прерывание
- Функции
  - Функции разбора
  - Функции кодирования и декодирования
  - Функции событий и активных списков
  - Функции коллекций

    - Функции строк
    - Функции типов и форматирования
    - Хэш-функции и криптографические функции
    - Функции генерации случайных значений и ID
    - Системные функции
    - Функции отладки
- Справочник ошибок
- Советы по написанию кода на языке VRL

О языке VRL

Vector Remap Language (VRL) — язык на основе выражений, созданный для работы с данными наблюдения. В него встроены различные функции, предназначенные для обработки журналов и метрик.

Применение VRL для преобразований данных

VRL работает на базе инструмента обработки данных Vector. Изначально в Vector преобразование данных осуществлялось на основе встроенного языка конфигурации, и появление VRL позволило существенно сократить описание трансформаций.

Рассмотрим распространенный случай использования — преобразование общего журнала Apache, поступающего из Docker.

```
{
  "time": "2021-02-03T21:13:54.713161211Z",
  "stream": "stdout",
  "log": "198.51.100.12 - user [03/Feb/2021:21:13:55 -0200] \"GET /embrace/supply-
chains/dynamic/vertical HTTP/1.0\" 201 20574"
}
```

Анализ этого журнала без VRL в Vector выглядит следующим образом:

```
# ... источники ...
# Анализ внутреннего журнала Syslog.
[transforms.parse_syslog]
type = "regex_parser"
inputs = ["parse_docker"]
patterns = ['^(?P<host>\S+) (?P<client>\S+) (?P<user>\S+) \[(?<timestamp>[\w:/]+\s[+\-
]\d{4})\] "(?<method>\S+) (?<resource>.+?) (?<protocol>\S+)" (?<status>\d{3})
(?<bytes_out>\S+)$']
field = "log"

# Удаление дублирующихся полей времени и журнала Docker.
```

```
[transform.remove_log]
type = "remove_fields"
inputs = ["parse_syslog"]
fields = ["time", "log"]

# Принудительная обработка полей.
[transforms.coerce_fields]
type = "coercer"
inputs = ["remove_log"]
types.timestamp = "timestamp"
types.status = "int"
types.bytes_out = "int"

# Приемники данных.
...
```

Результат преобразования:

```
{
  "bytes_out": 20574,
  "host": "198.51.100.12",
  "method": "GET",
  "resource": "/embrace/supply-chains/dynamic/vertical",
  "protocol": "HTTP/1.0",
  "status": 201,
  "timestamp": "2021-02-03T23:13:55Z",
  "client": "-",
  "user": "user"
}
```

С помощью языка VRL можно проанализировать вышеуказанный фрагмент так:

```
. = parse_common_log!(.log)
.total_bytes = del(.size)
.internal_request = ip_cidr_contains("198.51.100.0/16", .host) ?? false
```

Результат преобразования:

```
{
  "host": "198.51.100.12",
  "internal_request": true,
  "user": "user",
  "timestamp": "2021-02-03T23:13:55Z",
  "message": "GET /embrace/supply-chains/dynamic/vertical HTTP/1.0",
  "method": "GET",
  "path": "/embrace/supply-chains/dynamic/vertical",
  "protocol": "HTTP/1.0",
  "total_bytes": 20574,
```

```
  "status": 201
}
```

Конфигурация преобразования на VRL проще в написании, чтении и управлении, чем аналогичная на Vector. Добавлено дополнительное поле `internal_request`, чтобы продемонстрировать возможности встроенных функций VRL.

Принципы и характеристики языка VRL

VRL разработан с использованием принципов безопасности и производительности, оставаясь при этом гибким, поэтому он подходит для задач высокопроизводительной обработки событий ИБ.

Типобезопасность и надежность

VRL реализует принципы типобезопасности и надежности путем явного контроля над обработкой возможных ошибок. Это можно проиллюстрировать на примере фрагмента, который упоминался выше.

```
{
  "time":"2021-02-03T21:13:54.713161211Z",
  "stream": "stdout",
  "log": "198.51.100.12 - user [03/Feb/2021:21:13:55 -0200] \"GET /embrace/supply-
chains/dynamic/vertical HTTP/1.0\" 201 20574"
}
```

Его необходимо проанализировать и привести к результату:

```
{
  "host": "198.51.100.12",
  "user": "user",
  "timestamp": "2021-02-03T23:13:55Z",
  "message": "GET /embrace/supply-chains/dynamic/vertical HTTP/1.0",
  "method": "GET",
  "path": "/embrace/supply-chains/dynamic/vertical",
  "protocol": "HTTP/1.0",
  "total_bytes": 20574,
  "status": 201
}
```

Пользователь, только начавший работать с VRL, может написать следующую программу:

```
. = parse_common_log(.log)
.total_bytes = del(.size)
```

При компиляции он получит следующее сообщение об ошибке (Vector):

```
error[E103]: unhandled fallible assignment
 ┌── :1:5
 │
1 │ . = parse_common_log(.log)
 │ ___ ^^^^^^^^^^^^^^^^^^^^^^
 │ |   |
 │ |   this expression is fallible
 │ |   update the expression to be infallible
 │ or change this to an infallible assignment:
 │ ., err = parse_common_log(.log)
 │
= see documentation about error handling at https://errors.vrl.dev/#handling
= learn more about error code 103 at https://errors.vrl.dev/103
= see language documentation at https://vrl.dev
```

Как видно, VRL требует от пользователя обработки любого выражения, которое может привести к ошибке во время выполнения. В нашем случае `.log` может не быть строкой, поэтому нужно либо указать тип поля `.log`, либо обработать ошибку в случае, если `.log` не является строкой. Чтобы устранить эту ошибку, пользователь должен выполнить одно из следующих действий:

- Обработка ошибки:

  ```
  ., err = parse_common_log(.log)
  if err != null {
    .malformed = true
    log("Failed to parse common-log: " + err, level: "error")
  } else {
    .total_bytes = del(.size)
  }
  ```

  Если событие имеет неправильную форму (`.log` не является отформатированной строкой common-log), мы не выводим ошибку в лог и добавляем поле `.malformed`. Это сохраняет исходные данные и позволяет легко направить искаженные данные на проверку. Часто обработкой этой ошибки пренебрегают, что приводит к потере данных и простою системы.

- Выявление ошибки и прерывание программы:

  ```
  . = parse_common_log!(.log)
  .total_bytes = del(.size)
  ```

  Иногда наличие искаженных данных бывает неприемлемым, и программа должна быть прервана. В этом случае VRL предлагает варианты ошибочных функций. Здесь указывается, что добавление к вызову функции постфикса `!` вызовет сбой и прервет выполнение программы. При этом Vector перейдет к обработке следующего события, но одновременно запишет в журнал сообщение об ошибке. Таким образом, пользователям предлагается выбрать метод обработки ошибок.

- Указание типов:

```
.log = to_string!(.log)

., err = parse_common_log(.log)
if err != null {
  # Ошибка возникает только при невалидных строках в сообщении.
  log("Failed to parse common-log: " + err, level: "error")
} else {
  .total_bytes = del(.size)
}
```

В приведенных примерах можно заметить, что ошибку типа трудно отличить от ошибки синтаксического анализа. Возможно, вы предпочтете прерывать выполнение программы при возникновении ошибок типа и обрабатывать ошибки синтаксического анализа. Это достигается с помощью **прогрессивной безопасности типов**. В ходе оценки программа на языке VRL накапливает информацию о типах ваших полей. Когда тип распознается, ошибки типа исключаются при последующем использовании этого поля. Таким образом, в приведенном выше примере пользователь знает, что ошибка может возникнуть только в случае сбоя синтаксического анализа.

Эргономическая безопасность

VRL обеспечивает эргономическую безопасность, то есть делает сложным создание медленных или нестабильных программ. Проверки, выполняемые во время компиляции, предотвращают ошибки во время выполнения. Однако они не могут выявить некоторые более тонкие проблемы с производительностью и поддерживаемостью, которые возникают из-за сложности программы. Эти проблемы могут привести к нестабильности системы сбора и анализа метрик, а также непредвиденным затратам ресурсов.

Для защиты от эргономических трудностей, VRL был строго ограничен, предлагая только те функции, которые необходимы для преобразования данных наблюдения. Любые функции, которые не соответствуют данной задаче или могут ухудшить эргономику, были исключены из языка.

Безопасность доступа к памяти

VRL наследует от Rust гарантии безопасности доступа к памяти. Это позволяет избежать распространенных программных ошибок и уязвимостей, которые возникают из-за неправильного обращения к памяти. Благодаря этому VRL подходит для инфраструктурных приложений, таких как конвейеры обработки событий, где надежность и безопасность являются первостепенными целями.

Выражения

VRL — это язык, основанный на выражениях. Программа на VRL состоит исключительно из выражений, которые возвращают значения. Программы создаются по описанным далее синтаксическим правилам.

Литералы и динамические выражения



Используйте VRL-песочницу, чтобы проверить работу программы.

Программы на VRL состоят из литералов и динамических выражений. Выражения могут быть разделены символом новой строки или точкой с запятой в любых сочетаниях.

Примеры разделения выражений

Выражения, разделенные символом новой строки

```
del(.user_info)
.timestamp = now()
.message = "hello world"
```

Выражения, разделенные точкой с запятой

```
del(.user_info); .timestamp = now()
.message = "hello world"
```

Литералы

В VRL, как и в других языках, **литералы** — это значения, записанные в том виде, в каком они должны быть интерпретированы. К литералам относятся:

- Пустое значение
- Логическое значение
- Числа:
  - Целое число
  - Число с плавающей точкой
- Метка времени
- Строка
- Регулярное выражение
- Коллекции:
  - Массив
  - Объект

Динамические выражения

Динамические выражения возвращают значения, вычисленные на основе входных данных во время выполнения программы. Динамические выражения включают:

- Переменная
- Путь
- Индекс
- Присваивание

- Арифметическое выражение
- Сравнение
- Логическое выражение
- Вызов функции
- Объединение
- Условие
- Блок
- Прерывание

Ключевые слова

Ключевые слова — это зарезервированные слова, используемые для базовой функциональности языка, например, для работы с условиями. Они не могут использоваться для наименования переменных или других пользовательских директив. Следующие слова зарезервированы:

- `abort`
- `as`
- `break`
- `continue`
- `else`
- `false`
- `for`
- `if`
- `impl`
- `in`
- `let`
- `loop`
- `null`
- `return`
- `self`
- `std`
- `then`
- `this`
- `true`
- `type`
- `until`
- `use`
- `while`

Пробелы

Пробел — это любая непустая строка, определенная свойством Unicode `White_Space`.

VRL является языком свободной формы. Это означает, что все пробелы и отступы используются только для разделения токенов в грамматике и не влияют на семантику программы.

Комментарии

Комментарий используется для целей документирования программного кода и обозначается символом `#`. Каждая строка комментария должна начинаться с символа `#`.

В настоящее время VRL не поддерживает многострочные комментарии. Чтобы создать комментарий из нескольких строк, можно использовать ряд однострочных комментариев.

Примеры комментариев

Комментарий из одной строки

```
# Однострочный комментарий.
```

Комментарий из нескольких строк

```
# Первая строка комментария.
# Вторая строка комментария.
```

Пустое значение

Литерал **пустого значения** (Null) представляет отсутствие определенного значения.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры с пустыми значениями

Запись пустого значения

```
null
```

Сравнение пустых значений

Исходный код

```
null == null
```

Результат

```
true
```

Проверка типа

Для проверки типа пустого значения используйте функцию `is_null`.

Логическое значение

Литерал типа **логическое значение** (Boolean) представляет собой двоичное значение, которое может быть только истинным (`true`) или ложным (`false`).



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры логических значений

Истина

```
true
```

Ложь

```
false
```

Валидация и приведение типа

Для проверки типа и валидации логического значения используйте функции `bool` и `is_boolean`.

Для приведения к логическому значению используйте функцию `to_bool`.

Целое число

Литерал типа **целое число** (Integer) — это десятичное представление 64-битного целого числа со знаком.



Используйте VRL-песочницу, чтобы проверить работу программы.

Ограничения

Целые числа в VRL могут находиться в диапазоне от $-2^{63} + 1$ до $2^{63} - 1$. Числа, вышедшие за пределы диапазона переносятся на другой конец диапазона.

Пример выхода за пределы диапазона

Исходный код

```
.int_big = to_string(9_223_372_036_854_775_807)
.int_too_big = to_string(9_223_372_036_854_775_807 + 1)
```

Результат

```
{
```

```
  "int_big": "9223372036854775807",
  "int_too_big": "-9223372036854775808"
}
```

Подчеркивание

В целых числах может использоваться символ `_` для улучшения читаемости, например, для разделения разрядов.

Пример записи числа с подчеркиванием

Исходный код

```
1_000_000
```

Результат

```
1000000
```

Операции над целыми числами

К операциям над целыми числами относятся:

- Арифметические выражения
- Функции преобразования чисел

Для проверки типа и валидации целого числа используйте функции `int` и `is_integer`.

Для форматирования и приведения типов используйте функции представлений чисел.

Число с плавающей точкой

Литерал типа **число с плавающей точкой** (Float) представляет собой десятичное представление 64-битного числа с плавающей точкой, в частности, тип `binary64`, определенный в IEEE 754-2008.

Литерал числа с плавающей точкой записывается десятичными цифрами и может включать знак числа. Он состоит из целой части, десятичной точки и дробной части.



Если дробная часть числа с плавающей точкой равна нулю, в событии число отображается без дробной части.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры отображения чисел с плавающей точкой

Пример записи числа с плавающей точкой

```
1.001
```

Пример числа с плавающей точкой, равного целому числу

Исходный код

```
.number = 1.0
.number_is_float = is_float(.number)
```

Результат

```
{
  "number" = 1
  "number_is_float" = true
}
```

Ограничения

Числа с плавающей точкой в VRL могут находиться в диапазоне от `-1.7976931348623157E+308f64` до `1.7976931348623157E+308f64`. Числа за пределами этого диапазона превращаются в `null`.

Пример выхода за пределы диапазона

Исходный код

```
some_float = 100_000_000.0 * 100_000_000.0
some_float = some_float * some_float
some_float = some_float * some_float
some_float = some_float * some_float
.float_big = some_float * some_float
.float_too_big = .float_big * .float_big
```

Результат

```
{
  "float_big": 1.0000000000000005e+256,
  "float_too_big": null
}
```

Подчеркивания

В числах с плавающей точкой может использоваться символ `_` для улучшения читаемости, например, для разделения разрядов.

Пример записи числа с подчеркиваниями

Исходный код

```
1_000_000.01
```

Результат

```
1000000.01
```

Операции над числами с плавающей точкой

К операциям над числами с плавающей точкой относятся:

- Арифметические выражения
- Функции преобразования чисел

Для проверки типа и валидации числа с плавающей точкой используйте функции `float` и `is_float`.

Для форматирования и приведения типов используйте функции представлений чисел.

Метка времени

Литерал **метки времени** (Timestamp) — это нативная метка времени, выраженная в формате RFC 3339 с точностью до наносекунды.

Метки времени предваряются символом `t` и заключаются в одиночные кавычки: `t'2021-02-11T10:32:50.553955473Z'`.



Используйте VRL-песочницу, чтобы проверить работу программы.

Часовые пояса

Как определено в формате RFC 3339, литералы меток времени поддерживают UTC и локальные смещения времени.

Примеры меток времени

Метка времени с точностью до наносекунд

```
t'2021-02-11T10:32:50.553955473Z'
```

Метка времени с точностью до миллисекунд

```
t'2021-02-11T10:32:50.553Z'
```

Метка времени с указанным смещением

```
t'2021-02-11T10:32:50.553-04:00'
```

Валидация и приведение типа

Для проверки типа метки времени используйте функции `is_timestamp`.

Для форматирования и приведения типов используйте функции форматирования меток времени.

Чтобы получить текущую метку времени, используйте функцию `now`.

Строка

Литерал **строки** (String) представляет собой строку в кодировке UTF-8.

Строка с символами UTF-8

```
"Hello, world! 🌎"
```

Существуют сырые и интерпретированные строковые литералы:

- **Сырые (raw) строковые литералы** состоят из символов, закодированных напрямую в UTF-8, без интерпретации экранированных последовательностей. Сырые литералы предваряются символом `s` и заключаются в одиночные кавычки: `s'Hello, world!'`.

  Обратная косая черта не имеет специального значения в сырых литералах. Сырой литерал может содержать символы новой строки.

- **Интерпретированные строковые литералы** представляют собой последовательности символов, заключенных в двойные кавычки: `"Hello, world!"`.

  Текст в кавычках интерпретируется с учетом экранированных последовательностей и шаблонов. Поэтому обратная косая черта, двойные кавычки и двойные фигурные скобки должны быть экранированы.



Используйте VRL-песочницу, чтобы проверить работу программы.

Отображение экранированных символов

События отображаются в интерфейсе как JSON, поэтому двойные кавычки, обратные косые черты и символы новой строки в них экранируются автоматически.

Примеры отображения экранированных символов

Примеры строк с кавычками

Сырая строка, содержащая кавычки

```
s'{ "foo": "bar" }'
```

Интерпретированная строка, содержащая кавычки

```
"{ \"foo\": \"bar\" }"
```

Результат

```
"{ \"foo\": \"bar\" }"
```

Примеры строк с обратными косыми чертами

Исходный код

```
# Интерпретированный литерал, обратные косые черты экранируются.
.a = "\\currentcontrolset\\control"
# Сырой литерал, экранировать обратные косые черты не нужно.
.b = s'\currentcontrolset\control'
# Две обратные косые черты в сыром литерале интерпретируются так же, как введены.
.c = s'\\currentcontrolset\\control'
```

В интерфейсе, например, в карточке события, обратная косая черта отображается экранированной, то есть как две обратные косые черты.

Результат

```
{
   "a" = "\\currentcontrolset\\control",
   "b" = "\\currentcontrolset\\control",
   "c" = "\\\\currentcontrolset\\\\control"
}
```

Экранированные последовательности интерпретируются как один символ:

Исходный код

```
# Сырая строка.
.a = length(s'\\')
# Интерпретированная строка.
.b = length("\\")
```

Результат

```
{
   "a" = 2,
   "b" = 1
}
```

Экранированные последовательности

Специальные символы в интерпретированных литералах, такие как символы новой строки, могут быть выражены с помощью экранированных последовательностей.

Представление	Описание
`\u{7FFF}`	Символ Unicode из 24 бит (до 6 цифр)
`\n`	Перевод строки
`\r`	Возврат каретки

Представление	Описание
`\t`	Табуляция
`\\`	Обратная косая черта
`\0`	Пустой символ
`\"`	Двойная кавычка
`\'`	Одиночная кавычка
`\{`	Фигурная скобка

Примеры экранированных последовательностей

Пример с пустым символом

Исходный код

```
"Null symbol: \0"
```

Результат

```
"Null symbol: \u0000"
```

Пример с экранированной одиночной кавычкой

Исходный код

```
"Year \'25"
```

Результат

```
"Year '25"
```

Шаблоны

Интерпретированную строку можно шаблонизировать. Для этого имя переменной заключается в двойные фигурные скобки: `{{a}}`. Значение переменной вставляется в строку на этой позиции во время выполнения. Поддерживаются только переменные. Переменная должна быть строкой.



Чтобы вставить путь из события, сначала присвойте его переменной.

Чтобы вставить двойные фигурные скобки в строку, их можно экранировать с помощью обратной косой черты: `\{{...}}`.

Пример шаблонизации

Шаблонизация строки

```
planet = "Earth"
```

```
# Шаблонизированная строка.
.a = "Hello, {{ planet }}!"
# Строка без шаблона, двойная фигурная скобка экранирована.
.b = "Hello, \{{ planet }}!"
```

Результат

```
{
    "a" = "Hello, Earth!",
    "b" = "Hello, {{ planet }}!"
}
```

Многострочные литералы

Интерпретированные строковые литералы можно записать в несколько строк, добавив обратную косую черту перед символом новой строки. Символ новой строки и любые пробелы в начале следующей строки не будут включены в литерал.

Пример многострочного литерала

Исходный код

```
.a = "Hello,
    world!"
.b = "Hello, \
    world!"
```

Результат

```
{
    "a": "Hello, \n    world!",
    "b": "Hello, world!"
}
```

Недопустимые символы

Недопустимые последовательности UTF-8 заменяются символом .

Операции над строками

К операциям над строками относятся:

- Конкатенация (с помощью оператора +)
- Повторение строки заданное число раз (с помощью оператора *)
- Функции поиска и сопоставления
- Функции анализа строк
- Функции разбиения и слияния

- Функции редактирования
- Функции изменения регистра
- Функции разбора по регулярному выражению

Для проверки типа и валидации строки используйте функции `string`, `is_string` и `rv_is_utf8`.

Для приведения к строке используйте функцию `to_string`.

Регулярное выражение

Литерал **регулярного выражения** представляет собой регулярное выражение, используемое для операций сопоставления и разбора над строками.

Регулярные выражения предваряются символом `r` и заключаются в одиночные кавычки: `r'...'`. Используется синтаксис регулярных выражений Rust.



Для изучения регулярных выражений в VRL рекомендуется использовать редактор и тестер выражений Rustexp.

Обычные и специальные символы

Представление	Описание	Эквиваленты

`a`	Большинство символов представляют сами себя.	`\x61`
		`\u0061`
		`\U00000061`

`\\`	Специальные символы `[ ] \ / ^ $ . | ? * + ( ) { }` для представления самих себя должны быть экранированы.	`\x5C`
`\x61`	Шестнадцатеричный код символа ASCII из двух цифр.	`a`
`\u0062`	Шестнадцатеричный код символа Unicode из четырех цифр.	`b`
`\U00000062`	Шестнадцатеричный код символа Unicode из восьми цифр.	`c`

`\x{00003A9}`
`\u{000003A9}`
`\U{3A9}`
	Любой шестнадцатеричный код, которому соответствует символ Unicode.	`Ω`

`\t`	Символ горизонтальной табуляции.	`\x09`
`\n`	Символ перевода строки	`\x0A`
`\r`	Символ возврата каретки.	`\x0D`
`\v`	Символ вертикальной табуляции.	`\x0B`
`\a`	Символ звукового сигнала.	`\x07`
`\f`	Символ смены страницы.	`\x0C`

Метасимволы и специальные символьные классы

Представление	Описание и эквиваленты
`.`	Метасимвол, обозначающий любой символ, кроме символа перевода строки `\n`.
`\p{Property}`	Любой символ со свойством Unicode, указанным в фигурных скобках. Список поддерживаемых свойств представлен в документации репозитория regex.
`\P{Property}`	Любой символ, не имеющий свойства Unicode, указанного в фигурных скобках.
`\pX`	Любой символ Unicode, относящийся к классу, обозначенному одной буквой: • `L` — буквенные символы: графемы всех письменностей; • `M` — диакритические символы; • `N` — цифры; • `P` — знаки препинания; • `S` — знаки валют, математические и иные обозначения; • `Z` — пробельные символы, разделители строк и абзацев; • `C` — управляющие символы, символы из областей для частного использования и нераспределенных областей. Эквивалент: `\p{Property}`
`\PX`	Любой символ Unicode, не относящийся к классу, обозначенному одной буквой. Эквивалент: `\P{Property}`
`[[:ascii:]]`	Любой символ ASCII. Эквивалент: `[\x00-\x7F]`
`[[:print:]]`	Любой символ ASCII, доступный для печати: любой символ, кроме управляющих. Эквивалент: `[ -~]`
`[[:graph:]]`	Любой отображаемый символ ASCII: любой символ, кроме управляющих и пробельных. Эквивалент: `[!-~]`

Представление	Описание и эквиваленты
`\w`	Любой символ Unicode, входящий в состав слов: • буква или идеограмма любой письменности; • диакритический символ; • цифра; • знак пунктуации, соединяющий слова (например, _); • управляющий символ для лигатур. Эквивалент: `[\p{Alphabetic}\p{M}\d\p{Pc}\p{Join_Control}]`
`\W`	Любой символ Unicode, не входящий в состав слов. Эквивалент: `[^\w]`

`[[:word:]]`	Любая буква базового латинского алфавита, цифра ASCII или подчеркивание.
	Эквиваленты:
	`[0-9A-Za-z_]`
	`(?-u:\w)`
	`[\w&&\p{ascii}]`

`[[:alnum:]]`	Любая буква базового латинского алфавита или цифра ASCII. Эквивалент: `[0-9A-Za-z]`
`[[:alpha:]]`	Любая буква базового латинского алфавита. Эквивалент: `[A-Za-z]`
`[[:upper:]]`	Любая прописная буква базового латинского алфавита. Эквивалент: `[A-Z]`
`[[:lower:]]`	Любая строчная буква базового латинского алфавита. Эквивалент: `[a-z]`
`\d`	Любая десятичная цифра Unicode. Эквивалент: `\p{Nd}`
`\D`	Любой символ, кроме десятичных цифр Unicode. Эквивалент: `[^\d]`

Представление	Описание и эквиваленты

`[[:digit:]]`	Любая десятичная цифра ASCII.
	
	Эквиваленты:
	
	`[0-9]`
	
	`[\d&&\p{ascii}]`
	
	`(?-u:\d)`

`[[:xdigit:]]`	Любая шестнадцатеричная цифра. Эквивалент: `[0-9A-Fa-f]`
`[[:punct:]]`	Любой знак препинания ASCII. Эквивалент: `[!-/:@\\\[\]\{\}\(\)~‘]`
`\s`	Любой пробельный символ, то есть символ со свойством Unicode `Whitespace`. Эквивалент: `\p{White_Space}`
`\S`	Любой символ, кроме пробельных. Эквивалент: `\P{White_Space}`
`[[:space:]]`	Любой пробельный символ ASCII. Эквивалент: `[\t\n\v\f\r ]`
`[[:blank:]]`	Пробел или горизонтальная табуляция. Эквивалент: `[\t ]`
`[[:cntrl:]]`	Любой управляющий символ ASCII. Эквивалент: `[\x00-\x1F\x7F]`

Символьные классы и перечисление

Представление	Описание	Найденные совпадения
`xy`	Последовательность `x` и `y`.	`xy`
`x|y`	Перечисление: одна из последовательностей `x` или `y`, причем предпочтение отдается `x`.	`x`
`[xyz]`	Объединение: символы из указанных в квадратных скобках.	`x`, `y`, `z`
`[^xyz]`	Отрицание: все символы, кроме указанных в квадратных скобках.	Все символы, кроме `x`, `y` и `z`.
`[a-z]`	Диапазон: все символы между указанными.	Символы от `a` до `z` включительно.

Представление	Описание	Найденные совпадения
`[x[^xyz]]`	Вложенный класс: все символы, указанные хотя бы в одном из классов.	Все символы, кроме `y` и `z`.
`[a-y&&xyz]`	Пересечение: символы, которые входят в оба класса.	`x`, `y`
`[0-9--4]`	Разность: символы первого класса, которые не входят во второй класс. (Эквивалентно записи `[0-9&&[^4]]`).	Все десятичные цифры ASCII, кроме `4`.
`[a-g~~b-h]`	Симметрическая разность: символы, которые входят лишь в один класс из указанных.	`a`, `h`

Приоритет в перечислениях

В перечислениях проверяется каждый указанный вариант слева направо, поиск прекращается при первом совпадении.

Пример применения приоритета



Используйте VRL-песочницу, чтобы проверить работу программы.

Исходный код

```
.res = parse_regex_all!("authentication", r'authentication|auth', numeric_groups:
true)
```

Результат

```
[
  {
    "0": "authentication"
  }
]
```

Здесь функция с шаблоном `r'authentication|auth'` сначала пытается найти последовательность `"authentication"`, и поскольку она полностью соответствует строке, именно это совпадение и будет возвращено. Однако, если изменить порядок вариантов, изменится и результат.

Исходный код

```
.res = parse_regex_all!("authentication", r'auth|authentication', numeric_groups:
true)
```

Результат

```
[
```

```
  {
    "0": "auth"
  }
]
```

При таком порядке вариантов сначала будет найдено `"auth"`. На этом поиск остановится и вернет `"auth"`, несмотря на то, что более длинный вариант `"authentication"` также соответствует строке.

Порядок вычисления символьных классов

1. Диапазоны: выражение `r'[а-яё]'` равнозначно выражению `r'[[а-я]ё]'` и выбирает все строчные буквы русского алфавита.
2. Объединение: выражение `r'[ab&&bc]'` равнозначно выражению `r'[[ab]&&[bc]]'` и выбирает символы, которые входят в оба класса.
3. Пересечение, разность, симметрическая разность: имеют одинаковый приоритет и вычисляются по порядку слева направо. Например, следующее выражение:

   ```
   r'[\pL--\p\{Greek}&&\p\{Uppercase}]'
   ```

   равнозначно такому выражению:

   ```
   r'[[\pL--\p\{Greek}]&&\p\{Uppercase}]'
   ```

   Эти выражения выбирают все буквенные символы в верхнем регистре, кроме греческих.
4. Отрицание: выражение `r'[^a-z&&[:xdigit:]]'` равнозначно выражению `r'[^[a-z&&[:xdigit:]]]'`.

   Эти выражения выбирают все символы, кроме тех, которые одновременно являются шестнадцатеричными цифрами и строчными латинскими буквами.

Группировка

Группы в круглых скобках определяют области действия флагов, квантификаторов, и перечислений.

Регулярные выражения поддерживают захват групп, что позволяет извлекать части шаблона как строки. Именованные группы захвата позволяют связать результаты извлечения с ключами.

Представление	Описание
`(exp)`	Нумерованная группа захвата.
`(?P<name>exp)`	Именованная группа захвата. Возможно обращение как по номеру, так и по ключу, указанному в угловых скобках. (Эквивалентно записи `(?<name>exp)`.) Имя может включать буквенные символы и цифры Unicode, а также символы `.,_[]` и начинаться с буквенного символа либо подчеркивания.
`(?:exp)`	Незахватываемая группа.

Представление	Описание
`(?x)`	Установка флага `x` внутри текущей группы.
`(?-x)`	Выключение флага `x` внутри текущей группы.
`(?x-y:exp)`	Установка флага `x` и выключение флага `y` внутри незахватываемой группы.

Примеры группировки

Пример извлечения даты

Рассмотрим следующую функцию:

Исходный код

```
.res = parse_regex!("2012-12-12", r'(?P<y>\d{4})-(?P<m>\d{2})-(?P<d>\d{2})')
```

Результат

```
{
  "res": {
    "d": "12",
    "m": "12",
    "y": "2012"
  }
}
```

Данная функция извлекает из строки с датой вида `YYYY-MM-DD` группы, содержащие год, месяц и число с ключами `y`, `m` и `d` соответственно.

Пример изменения формата даты

В следующем примере используется захват именованных групп, чтобы изменить формат даты.

Исходный код

```
replace_with("2022-01-05, 2023-08-25 и 2024-10-18", r'(?x)
  (?P<y>\d{4}) # Год из 4 цифр Unicode.
  -
  (?P<m>\d{2}) # Месяц из 2 цифр Unicode.
  -
  (?P<d>\d{2}) # День из 2 цифр Unicode.
') -> |match| {
    to_string(match.d) + "/" + to_string(match.m) + "/" + to_string(match.y)
}
```

Результат

```
"05/01/2022, 25/08/2023 и 18/10/2024"
```

Квантификаторы

Квантификатор после символа, символьного класса или группы определяет, сколько раз может встречаться предшествующее выражение.

При жадной квантификации возвращается наиболее длинная строка из возможных соответствий, при ленивой — наиболее короткая.

Представление	Описание	Поведение
`x?`	Ноль или одно повторение `x`.	Жадная квантификация
`x*`	Ноль или больше повторений `x`.	Жадная квантификация
`x+`	Одно или больше повторений `x`.	Жадная квантификация
`x??`	Ноль или одно повторение `x`.	Ленивая квантификация
`x*?`	Ноль или больше повторений `x`.	Ленивая квантификация
`x+?`	Одно или больше повторений `x`.	Ленивая квантификация
`x{n}`	В точности `n` повторений `x`. Аналогично записи `x{n}?`.	Поведение не зависит от режима квантификации
`x{n,}`	Как минимум `n` повторений `x`.	Жадная квантификация
`x{n,m}`	Между `n` и `m` повторений `x`	Жадная квантификация
`x{n,}?`	Как минимум `n` повторений `x`.	Ленивая квантификация
`x{n,m}?`	Между `n` и `m` повторений `x`	Ленивая квантификация

Примеры квантификаторов

Следующая функция ищет последовательности символов, заключенные в квадратные скобки.

Исходный код

```
.res = parse_regex_all!("[ERROR] [2024-10-05 10:01:00] [code: 500]", r'\[.*\]',
numeric_groups: true)
```

Результат

```
[
  {
    "0": "[ERROR] [2024-10-05 10:01:00] [code: 500]"
  }
]
```

Если заменить квантификатор на ленивый: `r'\[.*?\]'`, функция вернет отдельные подстроки в квадратных скобках.

Исходный код

```
.res = parse_regex_all!("[ERROR] [2024-10-05 10:01:00] [code: 500]", r'\[.*?\]',
```

```
numeric_groups: true)
```

Результат

```
[
  {
    "0": "[ERROR]"
  },
  {
    "0": "[2024-10-05 10:01:00]"
  },
  {
    "0": "[code: 500]"
  }
]
```

Позиция в тексте

Следующие символы позволяют указать положение шаблона относительно начала и конца строки, а также границ слов.

Представление	Описание
`^`	Начало текста (в многострочном режиме — начало строки).
`$`	Конец текста (в многострочном режиме — конец строки).
`\A`	Только в начале текста, в том числе в многострочном режиме.
`\z`	Только в конце текста, в том числе в многострочном режиме.
`\b`	Граница слова Unicode: символ, соответствующий классу `\w` с одной стороны, и символ, соответствующий классу `\W`, либо граница текста, с другой стороны.
`\B`	Не на границе слова Unicode.
`\b{start}`, `\<`	Начало слова Unicode: символ класса `\W` или начало текста слева, символ класса `\w` справа.
`\b{end}`, `\>`	Конец слова Unicode: символ класса `\w` слева, символ класса `\w` или конец текста справа.
`\b{start-half}`	Проверка начала слова Unicode только слева: символ класса `\W` или начало текста.
`\b{end-half}`	Проверка конца слова Unicode только справа: символ класса `\W` или конец текста.

Примеры использования позиций

Следующая функция возвращает последовательности алфавитно-цифровых и соединительных символов на границе слова.

Исходный код

```
.res = parse_regex_all!("@guest, @superuser, @root", r'(?P<user>\b\w+)')
```

Результат

```
[
  {
    "user": "guest"
  },
  {
    "user": "superuser"
  },
  {
    "user": "root"
  }
]
```

Чтобы получить последовательности с символом `@`, изменим проверку начала слова.

Исходный код

```
.res = parse_regex_all!("@guest, @superuser, @root", r'(?P<user>\b{start-half}@\w+)')
```

Результат

```
[
  {
    "user": "@guest"
  },
  {
    "user": "@superuser"
  },
  {
    "user": "@root"
  }
]
```

Флаги

Регулярные выражения позволяют использовать флаги — модификаторы, которые изменяют поведение поиска для всего выражения или группы. Флаги могут быть объединены: `r'(?ixm)pattern'`, `r'(?im)pattern'`. Флаги можно включать и выключать внутри выражения.

Опция	Описание	Состояние по умолчанию
`i`	Режим без учета регистра: буквы в выражении соответствуют и прописным, и строчным буквам в строке.	Выключен

Опция	Описание	Состояние по умолчанию
`m`	Многострочный режим: • `^` соответствует началу новой строки. • `$` соответствует концу строки.	Выключен
`x`	Режим с поддержкой комментариев. Строчные комментарии должны начинаться с символа `#`. Пробелы при включенном режиме игнорируются. Чтобы добавить в шаблон пробел, используйте экранирование: `\ ` или шестнадцатеричный код: `\x20`.	Выключен
`s`	Разрешает метасимволу точки `.` соответствовать символу новой строки `\n`.	Выключен
`U`	Меняет местами значение квантификаторов `x*` и `x*?` (ленивый режим).	Выключен
`u`	Поддержка Unicode.	Включен

Пример флага

Рассмотрим следующую функцию:

Исходный код

```
.res = parse_regex!("AaAaAbbBBBb", r'(?i)a+(?-i)b+', numeric_groups: true)
```

Результат

```
{
  "0": "AaAaAbb"
}
```

В начале регулярного выражения включается регистронезависимый режим, поэтому `a+` соответствует буквам `a` и `A`. Однако `b+`, после выключения регистронезависимого режима, соответствует только строчной `b`.

Валидация и приведение типа

Для проверки типа регулярного выражения используйте функцию `is_regex`.

Для приведения типа к регулярному выражению используйте функцию `regex`.

Массив

Литерал **массива** представляет собой набор выражений, разделенных запятыми, который создает непрерывный массив. Массив может быть динамически расширен.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры массивов

Пустой массив

```
[]
```

Массив из строк

```
["первый", "второй", "третий"]
```

Массив из литералов разного типа, включая объект

```
["смешанный", 1, 1.0, true, false, {"foo": "bar"}]
```

Вложенные массивы

```
[
  "первый-уровень",
  [
    "второй-уровень",
    [
      "третий-уровень"
    ]
  ]
]
```

Массив из динамических выражений

```
[.поле1, .поле2, to_int!("2"), variable_1]
```

Массив из динамических выражений

```
[
  "выражения",
  1 + 2,
  2 == 5,
  true || false
]
```

Операции над массивами

Для работы с массивами используются следующие функции:

- Функции извлечения данных
- Функции фильтрации

- Функции итерации по элементам коллекции
- Функции изменения структуры коллекции
- Функции действий с отдельными элементами
- Функции разбиения и слияния массивов

Чтобы обратиться к элементу массива, используйте индекс этого элемента.

Для проверки типа и валидации массива используйте функции `array` и `is_array`.

Объект

Литерал **объекта** представляет собой расширяемую структуру "ключ — значение", которая синтаксически эквивалентна JSON-объекту.

Верно сформированный JSON-документ является допустимым объектом VRL.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры объектов

Пример объекта с вложенными массивом и объектом

```
{
  "поле1": "значение1",
  "поле2": [
      "значение2",
      "значение3",
      "значение4"
   ],
  "поле3": {
    "поле4": "значение5"
  }
}
```

Пример массива со значениями, представленными путями и переменными

```
{
  "поле1": ."некий_путь",
  "поле2": some_variable,
  "поле3": {
    "вложенное_поле": "некое значение"
  }
}
```

Упорядочивание

Поля объекта упорядочены по ключу в алфавитном порядке по возрастанию. Таким образом,

операции, такие как кодирование в JSON, производят строку с ключами, упорядоченными в алфавитном порядке.

Операции над объектами

К функциям операций над объектами относятся:

- Функции извлечения данных
- Функции фильтрации
- Функции итерации по элементам коллекции
- Функции изменения структуры коллекции
- Функции действий с отдельными элементами

Для обновления полей в объекте используйте присваивание со слиянием.

Для проверки типа и валидации объекта используйте функции `object` и `is_object`.

Переменная

Выражение **переменной** именует переменные. Название переменной представляет собой последовательность из одной или нескольких букв, цифр и подчеркиваний. Первый символ в названии переменной должен быть буквой или цифрой.



Рекомендуется использовать осмысленные названия переменных.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
first ~ (trailing)*
```

Элемент	Описание
`first`	Первый символ может быть только латинские буквы в любом регистре и цифры.

Элемент	Описание
`trailing`	Последующие символы должны содержать только латинские буквы в любом регистре, цифры и подчеркивания.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры переменных

Пример простой переменной

Исходный код

```
my_variable = 1
```

Результат

```
1
```

Пример переменной с путем

Исходный код

```
my_object = { "one": 1 }
my_object.one
```

Результат

```
1
```

Путь

Выражение **пути** представляет собой последовательность сегментов, разделенных точками, которые определяют местоположение значения внутри объекта. Ведущая точка означает, что путь указывает на **событие**. Ведущий знак процента означает, что путь указывает на **метаданные** события.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;

- `+` — элемент обязателен и может повторяться в выражении несколько раз;
- `*` — элемент может отсутствовать или повторяться несколько раз.

```
root_path ~ path_segments
```

Элемент	Описание
`root_path`	Корень пути, представленный одним из следующих символов: • Символ `.` представляет корень события. • Символ `%` представляет корень метаданных события.
`path_segments`	Сегменты вложенного пути. Каждый сегмент должен быть отделен точкой в начале и может содержать латинские буквы в любом регистре, цифры и подчеркивания. Сегменты, содержащие символы вне этого диапазона, должны быть заключены в кавычки.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры путей

Пример корневого пути события

Событие Vector

```
{
  "message": "Hello, World!"
}
```

Исходный код

```
.
```

Результат

```
{
  "message": "Hello, World!"
}
```

Пример корневого пути метаданных

Событие Vector

```
{
  "message": "Hello, World!"
}
```

Исходный код

```
%
```

Результат

```
{}
```

Пример пути верхнего уровня

Событие Vector

```
{
  "message": "Hello, World!"
}
```

Исходный код

```
.message
```

Результат

```
Hello, World!
```

Пример вложенного пути

Событие Vector

```
{
  "parent": {
    "child": "Hello, World!"
  }
}
```

Исходный код

```
.parent.child
```

Результат

```
Hello, World!
```

Пример объединения вложенных путей

Событие Vector

```
{
```

```
  "grand_parent": {
    "parent2": {
      "child": "Hello, World!"
    }
  }
}
```

Исходный код

```
.grand_parent.parent1.child || .grand_parent.parent2.child
```

Результат

```
Hello, World!
```

Пример пути в кавычках

Событие Vector

```
{
  "parent.key.with.special characters": {
    "child": "Hello, World!"
  }
}
```

Исходный код

```
."parent.key.with.special characters".child
```

Результат

```
Hello, World!
```

Пример пути к первому элементу массива

Событие Vector

```
{
  "массив": [
    "первый",
    "второй"
  ]
}
```

Исходный код

```
."массив"[0]
```

Результат

```
"первый"
```

Пример пути ко второму элементу массива

Событие Vector

```
{
  "массив": [
    "первый",
    "второй"
  ]
}
```

Исходный код

```
."массив"[1]
```

Результат

```
"второй"
```

Индекс

Выражение **индекса** обозначает элемент массива. Индексы массивов в VRL начинаются с нуля.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
"[" ~ index ~ "]"
```

Элемент	Описание
`index`	Индекс представляет собой позицию элемента, начиная с нуля.



Используйте VRL-песочницу, чтобы проверить работу программы.

Пример индекса элемента массива

Событие Vector

```
{
  "массив": [
    "первый",
    "второй"
  ]
}
```

Исходный код

```
."массив"[0]
```

Результат

```
"первый"
```

Присваивание

Выражение **присваивания** присваивает результат выражения справа от знака равенства цели слева. Целью может быть путь, индекс массива или переменная.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
target ~ ("," ~ error)? ~ operator ~ expression
```

Элемент	Описание
`target`	Цель. Должна быть переменной, индексом или путем. Возможно присваивание сразу нескольким целям, разделенным оператором `=`.
`error`	Ошибка. Позволяет необязательное присваивание, когда выражение справа может возвращать ошибки. Обычно используется при вызове функций, которые могут возвращать ошибки.
`operator`	Оператор. Разделяет `target` и `expression` и определяет условия присваивания: • `=` — обычное присваивание; • `|=` — присваивание со слиянием объектов.
`expression`	Выражение, зависящее от типа цели `target`: • Если `target` — это переменная, `expression` может быть любым выражением. • Если `target` — это путь, `expression` может быть любым выражением, которое возвращает поддерживаемый тип значения объекта, то есть любым выражением, кроме регулярного.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры присваивания

Пример присваивания пути

Исходный код

```
.message = "Hello, World!"
```

Результат

```
{
  "message": "Hello, World!",
}
```

Пример присваивания по вложенному пути

Исходный код

```
.parent.child = "Hello, World!"
```

Результат

```
{
  "parent": {
    "child": "Hello, World!"
  }
}
```

Пример присваивания элементу массива

Исходный код

```
.array[1] = "Hello, World!"
```

Результат

```
{
  "array": [
    null,
    "Hello, World!"
  ]
}
```

Пример присваивания переменной

Исходный код

```
my_variable = "Hello, World!"
```

Результат

```
"Hello, World!"
```

Пример присваивания с возможностью ошибки (успех)

Исходный код

```
.parsed, .err = parse_json("{\"Hello\": \"World!\"}")
```

Результат

```
{
  "parsed": {
    "Hello": "World!"
  }
}
```

Пример присваивания с возможностью ошибки (ошибка)

Исходный код

```
.parsed, .err = parse_json("malformed")
```

Результат

```
{
```

```
  "err": "malformed"
}
```

Множественное присваивание

Оператор `=` может использоваться для присваивания нескольким целям сразу. Множественное присваивание неприменимо, если используется присваивание ошибок.

Пример множественного присваивания

Программа VRL

```
.first = .second = "Hello, World!"
```

Результат

```
{
  "first": "Hello, World!",
  "second": "Hello, World!"
}
```

Присваивание со слиянием

Оператор `|=` обновляет поля в целевом объекте на основе содержимого исходного объекта. Здесь целевой объект — это объект в левой части выражения, а исходный — объект в правой части. При этом:

- Поля, которые есть только в целевом объекте, остаются неизменными.
- Поля, которые есть только в исходном объекте, добавляются в целевой.
- Если поле существует в обоих объектах, значение поля из исходного объекта заменяет значение из целевого объекта. Вложенные структуры в таком случае не учитываются.



Чтобы при слиянии объединить вложенные поля, используйте функцию `merge` с параметром `deep=true`.

Пример присваивания со слиянием

Исходный код

```
my_variable = {
  "parent1": {
    "child1": 1,
    "child2": 2
  },
  "parent2": {
    "child3": 3
  }
}
my_variable |= {
```

```
  "parent1": {
    "child2": 4,
    "child5": 5
  }
}
```

Результат

```
{
  "parent1": {
    "child2": 4,
    "child5": 5
  },
  "parent2": {
    "child3": 3
  }
}
```

Арифметическое выражение

Арифметическое выражение выполняет операцию над двумя выражениями (операндами) согласно оператору.

Хотя арифметическое выражение обычно применяется к числам, его можно использовать и с другими типами, такими как строки.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
expression ~ (operator ~ expression)+
```

Элемент	Описание
`expression`	Выражение может быть любым выражением, которое возвращает тип, допустимый для оператора.

Элемент	Описание
`operator`	Оператор определяет операцию, выполняемую над операндами слева и справа. Существующие операторы: • `+` — сложение чисел или конкатенация строк; • `-` — вычитание чисел; • `*` — перемножение чисел или повторение строки указанное число раз; • `/` — деление чисел (не целочисленное).

Для нахождения остатка используйте функцию `mod`.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры арифметических операций

Примеры сложения

Тип операндов	Исходный код	Результат
Целые числа	`1 + 1`	`2`
Числа с плавающей точкой	`0.1 + 0.2`	`0.30000000000000004`
Строки	`"Hello" + ", " + "World!"`	`"Hello, World!"`



Некоторые десятичные дроби не могут быть точно представлены в двоичном виде, что приводит к небольшим неточностям в вычислениях (например, `0.1 + 0.2 != 0.3`).

Примеры вычитания

Тип операндов	Исходный код	Результат
Целые числа	`2 - 1`	`1`
Числа с плавающей точкой	`2.0 - 1.1`	`0.9`

Примеры умножения

Тип операндов	Исходный код	Результат
Целые числа	`2 * 2`	`4`
Числа с плавающей точкой	`2.5 * 0.5`	`1.25`
Строка и число	`"строка" * 2`	`"строкастрока"`

Примеры деления

Тип операндов	Исходный код	Результат
Целые числа	`4 / 2`	`2`
Числа с плавающей точкой	`4.0 / 2.0`	`2`

Операции над числами разных типов

Арифметические операции над числами разных типов возвращают числа с плавающей точкой.

Примеры операций над числами разных типов

Операции над числами разных типов

Операция	Исходный код	Результат
Сложение	`1 + 1.1`	`2.1`
Вычитание	`2.0 - 1`	`1`
Умножение	`2.5 * 2`	`5`
Деление	`2.0 / 1`	`2`



Если дробная часть числа с плавающей точкой равна нулю, в событии число отображается без дробной части.

Приоритет операций

Действует стандартный приоритет операций: умножение и деление выполняются до сложения и вычитания. Для изменения приоритета используются скобки.

Примеры применения приоритета операций

Стандартный приоритет операций

Программа VRL

```
5 + 6 * 9 - 7
```

Результат

```
52
```

Задание приоритета с помощью скобок

Программа VRL

```
(5 + 6) * (9 - 7)
```

Результат

```
22
```

Сравнение

Выражение **сравнения** сравнивает два выражения (операнда) и возвращает логическое значение в соответствии с оператором.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
expression ~ operator ~ expression
```

Элемент	Описание
`expression`	Операнд может быть любым выражением, которое возвращает тип, допустимый для оператора.
`operator`	Оператор определяет операцию, выполняемую над операндами слева и справа: • `==` — равно; • `!=` — неравно; • `>` — больше; • `>=` — больше или равно; • `<` — меньше; • `<=` — меньше или равно.



Используйте VRL-песочницу, чтобы проверить работу программы.

Равенство и неравенство

Операторы `==` и `!=` предназначены для сравнения между собой любых значений. Числовые значения разных типов могут быть равными, другие значения считаются равными, только если относятся к одному типу.

Примеры равенства и неравенства

Пример равенства чисел

Исходный код

```
1 == 1.0
```

Результат

```
true
```

Пример неравенства чисел

Исходный код

```
2 * 2 != 5
```

Результат

```
true
```

Пример сравнения объектов

Исходный код

```
{"key1": "value1", "key2": "value2"} == {"key2": "value2", "key1": "value1"}
```

Результат

```
true
```

Пример сравнения пустых значений

Исходный код

```
null == null
```

Результат

```
true
```

Пример сравнения значений разных типов

Исходный код

```
2 == "2"
```

Результат

```
false
```

Сравнение порядка

Операторы сравнения >, <, >= и <= применимы только к числам, строкам и меткам времени.

При сравнении строк используется лексикографический порядок, где символы сравниваются последовательно, на основе их номеров в таблице Unicode.

При сравнении меток времени большими считаются метки более поздней даты.

Примеры сравнения порядка

Примеры сравнения чисел

Оператор	Исходный код	Результат
Больше или равно	`2 >= 2.0`	`true`
Больше	`2 > 1`	`true`
Меньше или равно	`2.0 <= 2`	`true`
Меньше	`1 < 2`	`true`

Пример сравнения временных меток

Исходный код

```
t'2024-04-04T22:22:22.234142+01:00' > t'2024-04-04T22:22:22.234142+04:00'
```

Результат

```
true
```

Пример сравнения строк

Исходный код

```
"ё" > "я"
# Буква "ё" находится в таблице Unicode после "я".
```

Результат

```
true
```

Логическое выражение

Выражение логического типа сравнивает два выражения (операнда), прекращая выполнение операций на последнем выражении, вычисленном в соответствии с оператором.



При использовании нескольких логических операторов необходимо явно задавать приоритет операций с помощью скобок, чтобы избежать двусмысленных условий.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

Унарные операции

```
operator ~ expression
```

Бинарные операции

```
expression ~ operator ~ expression
```

Элемент	Описание
`expression`	Операнд — любое выражение, которое возвращает допустимый тип, определенный оператором.
`operator`	Оператор, определяющий операцию, выполняемую над операндами. Существующие операторы: • `&&` — конъюнкция (AND); • `||` — дизъюнкция (OR); • `!` — отрицание (NOT).



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры логических выражений

Пример выражения с оператором `AND`

Исходный код

```
true && true
```

Результат

```
true
```

Пример выражения с оператором `OR` и логическим значением

Исходный код

```
false || "foo"
```

Результат

```
foo
```

Пример выражения с оператором `OR` и пустым значением

Исходный код

```
null || "foo"
```

Результат

```
foo
```

Пример выражения с оператором `NOT`

Исходный код

```
!false
```

Результат

```
true
```

Вызов функции

Выражение **вызова функции** вызывает встроенные функции VRL.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в

кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.

- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
function ~ abort? ~ "(" ~ arguments? ~ ")" ~ closure?
```

Элемент	Описание
`function`	Имя встроенной функции.
`abort`	Прерывание. Подробности приведены в разделе Прерывание.
`arguments`	Выражения, разделенные запятыми, которые могут дополнительно предваряться документированным именем. Подробности приведены в разделе Примеры способов передачи аргументов в функции.
`closure`	Замыкание. Подробности приведены в разделе Замыкание.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры способов передачи аргументов в функции

Пример позиционного вызова функции

Исходный код

```
split("hello, world!", ", ")
```

Результат

```
[
  "hello",
  "world!"
]
```

Пример вызова функции с упорядоченными именованными аргументами

Исходный код

```
split("hello, world!", pattern: ", ")
```

Результат

```
[
  "hello",
  "world!"
```

```
]
```

Пример вызова функции с неупорядоченными именованными аргументами

Исходный код

```
split(pattern: ", ", value: "hello, world!")
```

Результат

```
[
  "hello",
  "world!"
]
```

Замыкание

Замыкание — необязательный фрагмент кода, выполняемый при вызове функции. Замыкание применяется преимущественно в функциях, выполняющих итерации по коллекциям.

Пример синтаксиса:

```
for_each([]) -> |index, value| {
  ...
}
```

К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока замыкания, недоступны за его пределами.

Прерывание

Прерывание — литерал `!`, который можно использовать с функциями, возвращающими ошибки, для прерывания программы, если функция возвращает ошибку:

```
result = f!()
```

В противном случае ошибки должны обрабатываться:

```
result, err = f()
```

Отсутствие обработки ошибок от функций, которые могут возвращать ошибку, приводит к ошибкам на этапе компиляции. Обратитесь к разделу Справочник ошибок для дополнительной информации.

Примеры вызова функций с прерыванием и без прерываний

Пример вызова функции, не возвращающей ошибок

Исходный код

```
split("apples and pears and bananas", " and ")
```

Результат

```
["apples","pears","bananas"]
```

Примеры вызова функции, возвращающей ошибки

Исходное событие

```
{
  "log": {
    "message": "apples and pears and bananas"
  }
}
```

Исходный код

```
# Компилятор не может определить тип аргумента, поэтому нужно выполнить обработку
ошибки.
split!(.message, " and ")
```

Результат

```
["apples","pears","bananas"]
```

Объединение

Выражение **объединения** состоит из нескольких выражений (операндов), разделенных оператором объединения ??. Логика объединения позволяет выбирать из списка выражений первое значение, которое соответствует заданному условию.

Оператор ?? полезен при обработке ошибок, так как позволяет задавать альтернативные способы обработки данных.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.

- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
expression ~ (operator ~ expression)+
```

Элемент	Описание
expression	Любое выражение.
operator	Оператор. Разделяет два или более выражения.



Используйте VRL-песочницу, чтобы проверить работу программы.

Пример объединения при ошибке

Исходный код

```
parse_syslog("не Syslog") ?? parse_common_log("не Common Log Format") ?? "malformed"
```

Результат

```
malformed
```

Условие

Выражение **условия** определяет условное выполнение двух ветвей в зависимости от значения логического выражения. Если логическое выражение оценивается как истинное (`true`), выполняется ветвь `if`. В противном случае выполняется ветвь `else`, если она присутствует.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;

- `*` — элемент может отсутствовать или повторяться несколько раз.

```
"if" ~ predicate ~ block ~ ("else if" ~ predicate ~ block)* ~ ("else" ~ block)?
```

Элемент	Описание
`predicate`	Предикат, представленный выражением, которое приводится к значению логического типа. Если логическое значение не возвращается, возникает ошибка на этапе компиляции. Предикат может содержать несколько выражений. Несколько предикатов выражений должны быть заключены в скобки. Выражения должны быть разделены точкой с запятой (`;`) или новой строкой.
`block`	Выражение блока.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры условий

Пример условия `if true`

Исходный код

```
if true {
  "Hello, World!"
}
```

Результат

```
Hello, World!
```

Пример условия `if false`

Исходный код

```
if false {
  # Не вычисляется.
  null
}
```

Результат

```
null
```

Пример условия `if ... else`

Исходный код

```
if false {
  # Не вычисляется.
  null
} else {
  "Hello, World!"
}
```

Результат

```
Hello, World!
```

Пример условия `if ⋯ else if ⋯ else`

Исходный код

```
if false {
  # Не вычисляется.
  null
} else if false {
  # Не вычисляется.
  null
} else {
  "Hello, World!"
}
```

Результат

```
Hello, World!
```

Пример условия с несколькими выражениями в предикатах

Исходный код

```
x = 3
# Выражения в предикате разделены точкой с запятой.
if (x = x + 1; x == 5) {
  # Вычисления не производятся.
  null
} else if (
  # Выражения в предикате начинаются с новой строки.
  x = x + 1
  x == 5
) {
  "Hello, World!"
}
```

Результат

```
Hello, World!
```

Блок

Выражение **блока** представляет собой последовательность одного или нескольких выражений, заключенных в фигурные скобки.

Блоки не могут быть пустыми. Сочетание фигурных скобок без содержимого (`{}`) рассматривается как пустой объект.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
"{" ~ "NEWLINE"* ~ expressions ~ "NEWLINE"* ~ "}"
```

Элемент	Описание
`expressions`	Одно или несколько выражений.



Используйте VRL-песочницу, чтобы проверить работу программы.

Примеры блоков

Пример простого блока

Исходный код

```
{
  message = "{\"Hello\": \"World!\"}"
  parse_json!(message)
}
```

Результат

```
{
  "Hello": "World!"
}
```

Пример присваивания результата, полученного в блоке

Исходный код

```
.structured = {
  message = "{\"Hello\": \"World!\"}"
  parse_json!(message)
}
```

Прерывание

Выражение **прерывания** `abort` приводит к завершению программы VRL, прерывая любые изменения, вносимые в событие.

Синтаксис

▼ *Список принятых обозначений*

В данном разделе при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

```
"abort" ~ message?
```

Элемент	Описание
`message`	Необязательное отладочное сообщение, которое может быть использовано в диагностических целях и включается в метаданные события, отброшенного при преобразовании `remap`.



Используйте VRL-песочницу, чтобы проверить работу программы.

Пример прерывания обработки события, не соответствующего условию

Событие Vector

```
{
  "message": "hello world"
}
```

Исходный код

```
if contains(string!(.message), "hello") {
  abort
}
.message = "not hello world"
```

Результат

```
{
  "message": "hello world"
}
```

Функции



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.



Если нужно неоднократно обращаться к блоку кода в правилах нормализации и декларативных правилах корреляции, можно создать глобальную функцию.

Синтаксис

Синтаксис вызова функций VRL подробно описан в разделе Вызов функции. Рассмотрим следующую спецификацию функции:

```
redact(value: <строка | объект | массив>, filters: <массив>, [redactor: <строка |
объект> ]))
:: <строка | объект | массив>
```

Описанную функцию можно вызвать так:

Исходный код

```
redact("Userid 123456-7890", filters: [r'\d+\-\d+'], redactor: "sha2")
```

Возможность функции возвращать ошибки

Некоторые функции VRL могут быть помечены как возвращающие ошибки или не возвращающие ошибки. Когда функция определена как возвращающая ошибки, она может вернуть ошибку во время выполнения, требуя обработки ошибки перед компиляцией программы.

Если функция определена как не возвращающая ошибки, это означает, что **при правильных аргументах функции** функция никогда не может завершиться с ошибкой во время выполнения, и поэтому обработка ошибки не требуется.

Важно помнить, что даже если функция не должна возвращать ошибку, но при этом один из ее аргументов возвращает ошибку во время выполнения, вся функция будет считаться ошибочной. В такой ситуации необходимо обработать ошибку.

Следует отметить, что даже если функция определена как **не возвращающая ошибки**, но при этом любой из ее аргументов возвращает ошибку во время выполнения, функция рассматривается как **возвращающая ошибки**, и, следовательно, в этом случае необходимо будет обработать ошибку.

Компилятор VRL обеспечивает обработку всех потенциальных ошибок в программе, что исключает сбои во время выполнения.



Используйте VRL-песочницу, чтобы проверить работу программы.

Функции разбора



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции разбора по регулярному выражению

parse_regex

Разбирает строку `value` с использованием предоставленного регулярного выражения `pattern`. Эта функция отличается от функции `parse_regex_all` тем, что ищет только первое совпадение.



VRL предоставляет специализированные функции для анализа распространенных форматов журналов. Поскольку регулярные выражения требуют больших ресурсов, рекомендуется использовать функцию `parse_regex`, только если нет специальной функции VRL для нужного формата.

Функция возвращает объект с полями, соответствующими именованным группам. При включенной опции `numeric_groups` объект также включает поля, ключи которых являются последовательными номерами:

- поле `"0"` содержит совпадение полностью;
- последующие поля содержат захваченные группы по порядку.



Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Спецификация функции

```
parse_regex(value: <строка>, pattern: <регулярное выражение>, [numeric_groups:
<логическое значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска.		да
`pattern`	регулярное выражение	Регулярное выражение для поиска.		да
`numeric_groups`	логическое значение	Если `true`, будет возвращаться каждая захваченная группа в регулярном выражении. Индекс `0` будет содержать совпадение полностью.	`false`	нет



Если в `pattern` нет именованных групп и значение `numeric_groups` равно `false`, функция возвращает пустой объект.

Ошибки

Функция `parse_regex` **возвращает ошибку**, для которой требуется обработка, если строку `value` не удалось разобрать с использованием предоставленного шаблона `pattern`.

Примеры

Разбор с использованием регулярного выражения с захватом именованной группы

Исходный код

```
parse_regex!("Connection attempt from 192.0.2.0, from 192.0.2.1.",
r'from (?P<ip>\d+\.\d+\.\d+\.\d+)')
```

Результат

```
{
  "ip": "192.0.2.0"
```

```
}
```

Разбор с использованием регулярного выражения без захвата именованных групп

Исходный код

```
parse_regex!("Connection attempt from 192.0.2.0, from 192.0.2.1.",
r'from (\d+\.\d+\.\d+\.\d+)',
numeric_groups: true)
```

Результат

```
{
  "0": "from 192.0.2.0",
  "1": "192.0.2.0"
}
```

parse_regex_all

Разбирает `value` с использованием предоставленного регулярного выражения `pattern`. Эта функция отличается от функции `parse_regex` тем, что ищет **все** совпадения, а не только первое.



VRL предоставляет функции специального назначения для разбора общих форматов журналов. Прежде чем использовать функцию `parse_regex_all`, проверьте, существует ли уже функция VRL, которая разбирает нужный формат.

Функция возвращает массив объектов с полями, соответствующими именованным группам. При включенной опции `numeric_groups` объекты также включают поля, ключи которых являются последовательными номерами:

- поле `"0"` содержит совпадение полностью;
- последующие поля содержат захваченные группы по порядку.



Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Спецификация функции

```
parse_regex_all(value: <строка>, pattern: <регулярное выражение>, [numeric_groups:
<логическое значение>])
:: <массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`pattern`	регулярное выражение	Регулярное выражение для поиска.		да
`numeric_groups`	логическое значение	Если `true`, будет возвращаться каждая захваченная группы в регулярном выражении. Индекс `0` будет содержать совпадение полностью.	`false`	нет



Если в `pattern` нет именованных групп и значение `numeric_groups` равно `false`, функция возвращает массив пустых объектов.

Ошибки

Функция `parse_regex_all` **возвращает ошибку**, для которой требуется обработка, если `value` не удалось разобрать с использованием предоставленного шаблона `pattern`.

Примеры

Разбор с использованием регулярного выражения (все совпадения)

Исходный код

```
parse_regex_all!("Connection attempt from 192.0.2.0, from 192.0.2.1.",
r'from (?P<ip>\d+\.\d+\.\d+\.\d+)',
numeric_groups: true)
```

Результат

```
[
  {
    "0": "from 192.0.2.0",
    "1": "192.0.2.0",
    "ip": "192.0.2.0"
  },
  {
    "0": "from 192.0.2.1",
    "1": "192.0.2.1",
    "ip": "192.0.2.1"
  }
]
```

Функции разбора форматов логов

parse_apache_log

Разбирает строки журнала доступа и ошибок Apache. Строки могут иметь формат `common`, `combined` или `error`.



Отсутствие информации в сообщении журнала может быть обозначено `-`. Эти поля опускаются в результате, возвращаемом функцией.

Спецификация функции

```
parse_apache_log(value: <строка>, format: <строка>, [timestamp_format: <строка>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`format`	строка	Формат, используемый для разбора журнала.		да
`timestamp_format`	строка	Формат даты и времени, используемый для кодирования метки времени. Время разбирается в локальном часовом поясе, если метка времени не указывает часовой пояс.	`%d/%b/%Y:%T %z`	нет

Ошибки

Функция `parse_apache_log` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует указанному формату;
- `timestamp_format` не является допустимой строкой формата;
- метку времени в `value` не удалось разобрать с использованием предоставленного формата `timestamp_format`.

Примеры

Разбор строки в формате журнала Apache (common)

Исходный код

```
parse_apache_log!("192.0.2.0 bob frank [10/Oct/2000:13:55:36 -0700] \"GET
/apache_pb.gif HTTP/1.0\" 200 2326",
format: "common")
```

Результат

```
{
  "host": "192.0.2.0",
  "identity": "bob",
  "message": "GET /apache_pb.gif HTTP/1.0",
  "method": "GET",
  "path": "/apache_pb.gif",
  "protocol": "HTTP/1.0",
```

```
  "size": 2326,
  "status": 200,
  "timestamp": "2000-10-10T20:55:36Z",
  "user": "frank"
}
```

Разбор строки в формате журнала Apache (combined)

Исходный код

```
parse_apache_log!(
s'192.0.2.0 bob frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200
2326 "http://www.example.com/vertical/channels/front-end/bandwidth" "Mozilla/5.0 (X11;
Linux i686; rv:5.0) Gecko/1945-10-12 Firefox/37.0"',
"combined")
```

Результат

```
{
  "agent": "Mozilla/5.0 (X11; Linux i686; rv:5.0) Gecko/1945-10-12 Firefox/37.0",
  "host": "192.0.2.0",
  "identity": "bob",
  "message": "GET /apache_pb.gif HTTP/1.0",
  "method": "GET",
  "path": "/apache_pb.gif",
  "protocol": "HTTP/1.0",
  "referrer": "http://www.example.com/vertical/channels/front-end/bandwidth",
  "size": 2326,
  "status": 200,
  "timestamp": "2000-10-10T20:55:36Z",
  "user": "frank"
}
```

Разбор строки в формате журнала Apache (ошибка)

Исходный код

```
parse_apache_log!(
s'[01/Mar/2021:12:00:19 +0000] [ab:alert] [pid 4803:tid 3814] [client
147.159.108.175:24259] I will bypass the haptic COM bandwidth, that should matrix the
CSS driver!',
"error")
```

Результат

```
{
  "client": "147.159.108.175",
  "message": "I will bypass the haptic COM bandwidth, that should matrix the CSS
driver!",
```

```
  "module": "ab",
  "pid": 4803,
  "port": 24259,
  "severity": "alert",
  "thread": "3814",
  "timestamp": "2021-03-01T12:00:19Z"
}
```

rv_parse_avro

Разбирает `value` как файл формата Apache Avro, который содержит метаданные со схемой.



Если сообщение не содержит схемы, используйте функцию `rv_parse_avro_raw`.



Тип `Decimal` в сообщениях Avro не поддерживается.

Спецификация функции

```
rv_parse_avro(value: <строка>, [reader_schema: <строка>])
:: <массив объектов>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения Avro		да
`reader_schema`	JSON	Схема для считывания в формате JSON. Строка должна быть представлена литералом, без использования переменных.		нет



Использование схемы считывания, заданной в параметре `reader_schema`, может замедлить выполнение функции.

Ошибки

Функция `rv_parse_avro` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректным сообщением Avro;
- `value` закодировано не в UTF-8;
- `reader_schema` не является строкой;
- невозможно разобрать предоставленную `reader_schema`;
- значения типов `TimeMicros`, `TimeMillis`, `Float` и `Double` в разобранном сообщении не являются числовыми.

Примеры

Базовый разбор

Исходный код

```
# .message -- путь к сообщению в формате Avro.
rv_parse_avro!(.message)
```

Результат

```
# Пример массива объектов.
[
  {
    "entity": "001",
    "time": "1433269388"
  },
  {
    "entity": "002",
    "time": "1433270488"
  },
]
```

С указанием схемы

Исходный код

```
# .message -- путь к сообщению в формате Avro.
rv_parse_avro!(.message,
reader_schema: s'{"type": "record", "fields": [{"name": "element", "type": "string"},
{"name": "time", "type": "string"}]}')
```

Результат

```
# Пример массива объектов.
[
  {
    "element": "001",
    "time": "1433269388"
  },
  {
    "element": "002",
    "time": "1433270488"
  },
]
```

rv_parse_avro_raw

Разбирает `value` как отдельную запись Apache Avro, сериализованную без заголовков или встроенной схемы. Функция требует предоставлять схему расшифровки данных.



Тип `Decimal` в сообщениях Avro не поддерживается.

Спецификация функции

```
rv_parse_avro_raw(value: <строка>, decoder_schema: <avro_schema>, [reader_schema:
<строка>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения Avro		да
`decoder_schema`	строка	Схема для расшифровки данных. Строка должна быть представлена литералом, без использования переменных.		да
`reader_schema`	JSON	Схема в формате JSON для валидации расшифрованных данных. Строка должна быть представлена литералом, без использования переменных.		нет



Использование схемы считывания, заданной в параметре `reader_schema`, может замедлить выполнение функции.

Ошибки

Функция `rv_parse_avro_raw` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректным сообщением Avro;
- `decoder_schema` не является строкой;
- `reader_schema` не является строкой;
- невозможно разобрать предоставленную `decoder_schema`;
- невозможно разобрать предоставленную `reader_schema`;
- значения типов `TimeMicros`, `TimeMillis`, `Float` и `Double` в разобранном сообщении не являются числовыми.

Примеры

Базовый разбор

Исходный код

```
# .message -- путь к сообщению в формате Avro.
rv_parse_avro_raw!(.message,
```

```
decoder_schema: s'{"type": "record", "fields": [{"name": "element", "type": "string"},
{"name": "time", "type": "string"}]}')
```

Результат

```
# Пример объекта.
{
  "element": "value",
  "time": "1433270488"
}
```

С указанием схемы считывания

Исходный код

```
# .message -- путь к сообщению в формате Avro
rv_parse_avro_raw!(.message,
decoder_schema: s'{"type": "record", "fields": [{"name": "element", "type": "string"},
{"name": "time", "type": "string"}]}',
reader_schema: s'{"type": "record", "fields": [{"name": "element", "type":
"string"}]}')
```

Результат

```
# Пример объекта.
{
  "element": "value"
}
```

parse_aws_alb_log

Разбирает `value` в формате журнала доступа Application Load Balancer в AWS (Amazon Web Services).

Спецификация функции

```
parse_aws_alb_log(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Журнал доступа Application Load Balancer.		да

Ошибки

Функция `parse_aws_alb_log` **возвращает ошибку**, для которой требуется обработка, если `value` не соответствует указанному формату лога AWS ALB.

Пример

Разбор лога AWS ALB

Исходный код

```
parse_aws_alb_log!(
"http 2018-11-30T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 - 0.000 0.001 0.000 200 200 34 366 \"GET
http://www.example.com:80/ HTTP/1.1\" \"curl/7.46.0\" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067 \"Root=1-58337364-23a8c76965a2ef7629b185e3\" \"-\" \"-\" 0
2018-11-30T22:22:48.364000Z \"forward\" \"-\" \"-\" \"-\" \"-\" \"-\" \"-\""
)
```

Результат

```
{
  "actions_executed": "forward",
  "chosen_cert_arn": null,
  "classification": null,
  "classification_reason": null,
  "client_host": "192.168.131.39:2817",
  "domain_name": null,
  "elb": "app/my-loadbalancer/50dc6c495c0c9188",
  "elb_status_code": "200",
  "error_reason": null,
  "matched_rule_priority": "0",
  "received_bytes": 34,
  "redirect_url": null,
  "request_creation_time": "2018-11-30T22:22:48.364000Z",
  "request_method": "GET",
  "request_processing_time": 0,
  "request_protocol": "HTTP/1.1",
  "request_url": "http://www.example.com:80/",
  "response_processing_time": 0,
  "sent_bytes": 366,
  "ssl_cipher": null,
  "ssl_protocol": null,
  "target_group_arn": "arn:aws:elasticloadbalancing:us-east-
2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
  "target_host": null,
  "target_port_list": [],
  "target_processing_time": 0.001,
  "target_status_code": "200",
  "target_status_code_list": [],
  "timestamp": "2018-11-30T22:23:00.186641Z",
  "trace_id": "Root=1-58337364-23a8c76965a2ef7629b185e3",
  "traceability_id": null,
  "type": "http",
  "user_agent": "curl/7.46.0"
```

```
}
```

parse_aws_cloudwatch_log_subscription_message

Разбирает события журналов AWS CloudWatch, настроенных через подписки Cloudwatch, из источника `aws_kinesis_firehose`.

Спецификация функции

```
parse_aws_cloudwatch_log_subscription_message(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения для разбора.		да

Ошибки

Функция `parse_aws_cloudwatch_log_subscription_message` **возвращает ошибку**, для которой требуется обработка, если `value` не является правильно отформатированным сообщением подписки AWS Cloudwatch Log.

Пример

Разбор сообщения подписки AWS Cloudwatch Log

Исходный код

```
# .message -- пример пути к сообщению журнала AWS CloudWatch.
parse_aws_cloudwatch_log_subscription_message!(.message)
```

Результат

```
{
  "log_events": [
    {
      "id": "35683658089614582423604394983260738922885519999578275840",
      "message": "{\"bytes\":26780,\"datetime\":\"14/Sep/2020:11:45:41
-0400\",\"host\":\"192.0.2.1\",\"method\":\"PUT\",\"protocol\":\"HTTP/1.0\",\"referer\
":\"https://www.principalcross-
platform.io/markets/ubiquitous\",\"request\":\"/expedite/convergence\",\"source_type\"
:\"stdin\",\"status\":301,\"user-identifier\":\"-\"}",
      "timestamp": "2020-09-14T19:09:29.039Z"
    }
  ],
  "log_group": "test",
  "log_stream": "test",
  "message_type": "DATA_MESSAGE",
```

```
  "owner": "111111111111",
  "subscription_filters": [
    "Destination"
  ]
}
```

parse_aws_vpc_flow_log

Разбирает `value` в формате журналов потокового контроля AWS Virtual Private Cloud (VPC Flow Logs).

Спецификация функции

```
parse_aws_vpc_flow_log(value: <строка>, [format: <строка>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Журнал потоков VPC.		да
`format`	строка	Формат журнала потоков VPC.		нет

Ошибки

Функция `parse_aws_vpc_flow_log` **возвращает ошибку**, для которой требуется обработка, если `value` имеет неправильный формат журнала потоков VPC AWS.

Примеры

Разбор журнала потоков VPC AWS в формате по умолчанию

Исходный код

```
parse_aws_vpc_flow_log!("2 123456789010 eni-1235b8ca123456789 - - - - - - - 1431280876
1431280934 - NODATA")
```

Результат

```
{
  "account_id": "123456789010",
  "action": null,
  "bytes": null,
  "dstaddr": null,
  "dstport": null,
  "end": 1431280934,
  "interface_id": "eni-1235b8ca123456789",
  "log_status": "NODATA",
  "packets": null,
  "protocol": null,
  "srcaddr": null,
```

```
  "srcport": null,
  "start": 1431280876,
  "version": 2
}
```

Разбор журнала потоков VPC AWS в пользовательском формате

Исходный код

```
parse_aws_vpc_flow_log!(
"- eni-1235b8ca123456789 10.0.1.5 10.0.0.220 10.0.1.5 203.0.113.5",
"instance_id interface_id srcaddr dstaddr pkt_srcaddr pkt_dstaddr"
)
```

Результат

```
{
  "dstaddr": "10.0.0.220",
  "instance_id": null,
  "interface_id": "eni-1235b8ca123456789",
  "pkt_dstaddr": "203.0.113.5",
  "pkt_srcaddr": "10.0.1.5",
  "srcaddr": "10.0.1.5"
}
```

parse_cef

Разбирает `value` в формате CEF (Common Event Format). Игнорирует все данные до заголовка CEF.



Рекомендуется использовать функцию `rv_parse_cef`, которая поддерживает заголовки syslog.



- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.
- Обрамляющие кавычки удаляются из значений.
- Пустые значения возвращаются как пустые строки.

Спецификация функции

```
parse_cef(value: <строка>, [translate_custom_fields: <логическое значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`translate_custom_fields`	логическое значение	Преобразовывать пользовательские пары полей в формат `ключ: значение`	`false`	нет

Ошибки

Функция `parse_cef` **возвращает ошибку**, для которой требуется обработка, если `value` имеет неправильный формат строки CEF.

Примеры

Разбор сообщения, сгенерированного Privileged Threat Analytics (PTA)

Исходный код

```
parse_cef!(
"CEF:0|CyberArk|PTA|12.6|1|Suspected credentials theft|8|suser=mike2@prod1.example.com
shost=prod1.example.com src=198.51.100.0 duser=andy@dev1.example.com
dhost=dev1.example.com dst=203.0.113.0 cs1Label=ExtraData cs1=None cs2Label=EventID
cs2=52b06812ec3500ed864c461e deviceCustomDate1Label=detectionDate
deviceCustomDate1=1388577900000 cs3Label=PTAlink
cs3=https://198.51.100.0/incidents/52b06812ec3500ed864c461e cs4Label=ExternalLink
cs4=None"
)
```

Результат

```
{
  "cefVersion": "0",
  "cs1": "None",
  "cs1Label": "ExtraData",
  "cs2": "52b06812ec3500ed864c461e",
  "cs2Label": "EventID",
  "cs3": "https://198.51.100.0/incidents/52b06812ec3500ed864c461e",
  "cs3Label": "PTAlink",
  "cs4": "None",
  "cs4Label": "ExternalLink",
  "deviceCustomDate1": "1388577900000",
  "deviceCustomDate1Label": "detectionDate",
  "deviceEventClassId": "1",
  "deviceProduct": "PTA",
  "deviceVendor": "CyberArk",
  "deviceVersion": "12.6",
  "dhost": "dev1.example.com",
  "dst": "203.0.113.0",
  "duser": "andy@dev1.example.com",
  "name": "Suspected credentials theft",
  "severity": "8",
  "shost": "prod1.example.com",
```

```
  "src": "198.51.100.0",
  "suser": "mike2@prod1.example.com"
}
```

Сообщение с заголовком syslog, которое игнорируется

Исходный код

```
parse_cef!(
"Sep 29 08:26:10 host CEF:1|Security|threatmanager|1.0|100|worm successfully
stopped|10|src=198.51.100.0 dst=203.0.113.0 spt=1232"
)
```

Результат

```
{
  "cefVersion": "1",
  "deviceVendor": "Security",
  "deviceProduct": "threatmanager",
  "deviceVersion": "1.0",
  "deviceEventClassId": "100",
  "name": "worm successfully stopped",
  "severity": "10",
  "src": "198.51.100.0",
  "dst": "203.0.113.0",
  "spt": "1232"
}
```

Преобразование пользовательских полей

Исходный код

```
parse_cef!(
"CEF:0|Dev|firewall|2.2|1|Connection
denied|5|c6a1=2001:0DB8:2CA1:0000:0000:0567:5673:23b5 c6a1Label=Device IPv6 Address",
translate_custom_fields: true
)
```

Результат

```
{
  "cefVersion": "0",
  "deviceVendor": "Dev",
  "deviceProduct": "firewall",
  "deviceVersion": "2.2",
  "deviceEventClassId": "1",
  "name": "Connection denied",
  "severity": "5",
  "Device IPv6 Address": "2001:0DB8:2CA1:0000:0000:0567:5673:23b5"
```

```
}
```

rv_parse_cef

Разбирает `value` в формате Common Event Format (CEF). Поддерживает и обрабатывает необязательный заголовок syslog.

Спецификация функции

```
rv_parse_cef(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление записи CEF		да

Ошибки

Функция `rv_parse_cef` **может возвращать ошибки**, для которых требуется обработка:

- строка `value` не начинается с обязательного префикса `CEF:`;
- строка `value` не содержит обязательных полей и не является корректной записью CEF.

Примеры

Базовый разбор

Исходный код

```
rv_parse_cef!(s'CEF:0|Microsoft|Microsoft Windows||Microsoft-Windows-Security-
Auditing:4624|An account was successfully logged on.|Low| eventId=1013540004
externalId=4624')
```

Результат

```
{
  "deviceVendor": "Microsoft",
  "deviceProduct": "Microsoft Windows",
  "deviceVersion": "",
  "deviceEventClassId": "Microsoft-Windows-Security-Auditing:4624",
  "name": "An account was successfully logged on.",
  "severity": "Low",
  "eventId": "1013540004",
  "externalId": "4624"
}
```

Сообщение с заголовком syslog

Исходный код

```
value = r#"<134>1 2022-02-14T03:17:30-08:00 TEST
CEF:0|Vendor|Product|20.0.560|600|User Signed In|3|src=192.0.2.255 "#
rv_parse_cef!(value)
```

Результат

```
{
  ahost: "TEST",
  at: "2022-02-14T03:17:30-08:00",
  deviceProduct: "Product",
  deviceVendor: "Vendor",
  deviceVersion: "20.0.560",
  name: "User Signed In",
  severity: "3",
  signatureId: "600",
  src: "192.0.2.255",
  syslog_facility: "16",
  syslog_priority: "134",
  syslog_severity: "6",
})
```

parse_common_log

Разбирает `value` с использованием формата общего журнала Apache (Common Log Format, CLF).



Отсутствие информации в сообщении журнала может быть обозначено `-`. Эти поля опускаются в результате, возвращаемом функцией.

Спецификация функции

```
parse_common_log(value: <строка>, [timestamp_format: <строка>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`timestamp_format`	строка	Формат даты и времени для кодирования метки времени.	`%d/%b/%Y:%T %z`	нет

Ошибки

Функция `parse_common_log` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует формату общего журнала.
- `timestamp_format` не является допустимой строкой формата.

- Метку времени в `value` не удается разобрать с использованием предоставленного формата `timestamp_format`.

Примеры

Разбор сообщения в формате общего журнала (с форматом метки времени по умолчанию)

Исходный код

```
parse_common_log!("192.0.2.2 bob frank [10/Oct/2000:13:55:36 -0700] \"GET
/apache_pb.gif HTTP/1.0\" 200 2326")
```

Результат

```
{
  "host": "192.0.2.2",
  "identity": "bob",
  "user": "frank",
  "timestamp": "2000-10-10T20:55:36Z",
  "message": "GET /apache_pb.gif HTTP/1.0",
  "method": "GET",
  "path": "/apache_pb.gif",
  "protocol": "HTTP/1.0",
  "status": 200,
  "size": 2326
}
```

Разбор сообщения в формате общего журнала с пользовательским форматом метки времени

Исходный код

```
parse_common_log!(
"192.0.2.25 bob frank [2000-10-10T20:55:36Z] \"GET /apache_pb.gif HTTP/1.0\" 200
2326",
"%+"
)
```

Результат

```
{
  "host": "192.0.2.25",
  "identity": "bob",
  "user": "frank",
  "timestamp": "2000-10-10T20:55:36Z",
  "message": "GET /apache_pb.gif HTTP/1.0",
  "method": "GET",
  "path": "/apache_pb.gif",
  "protocol": "HTTP/1.0",
  "status": 200,
  "size": 2326
```

```
}
```

rv_extract_from_dit

Извлекает домен и имя пользователя из записи DIT (Data Information Tree), используемой в протоколе LDAP.



- Результат возвращается в виде JSON-объекта с полями `"username"` и `"domain"`.
- Имя пользователя и домен возвращаются в нижнем регистре.

Спецификация функции

```
rv_extract_from_dit(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление записи DIT		да

Ошибки

Функция `rv_extract_from_dit` **может возвращать ошибки**, для которых требуется обработка:

- `value` закодировано не в UTF-8.
- `value` не является корректной записью DIT.

Пример

Исходный код

```
rv_extract_from_dit!("CN=Dev-India,OU=Distribution
Groups,DC=gp,DC=gl,DC=example,DC=com")
```

Результат

```
{
  "domain": "gp.gl.example.com",
  "username": "dev-india"
}
```

parse_etld

Разбирает eTLD (effective top-level domain) из строки `value`, представляющей собой доменное имя.

Спецификация функции

```
parse_etld(value: <строка>, [plus_parts: <целое число>, psl: <строка>])
```

```
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка домена.		да
`plus_parts`	целое число	Число поддоменов, возвращаемых функцией.		нет
`psl`	строка	Путь к файлу или URL списка публичных суффиксов. По умолчанию используется список publicsuffix.org/list/public_suffix_list.dat.		нет

`parse_etld` возвращает структуру вида:

```
{
  "etld": "ac.uk",
  "etld_plus": "example.ac.uk",
  "known_suffix": true
}
```

- `etld` — имя домена;
- `etld_plus` — имя домена с поддоменами. Количество поддоменов определяется аргументом `plus_parts`;
- `known_suffix` — `true`, если суффикс входит в список, переданный в аргументе `psl`, `false` — иначе.

Ошибки

Функция `parse_etld` **возвращает ошибку**, для которой требуется обработка, если невозможно определить eTLD для строки `value`.

Примеры

Разбор eTLD

Исходный код

```
parse_etld!("sub.example.com")
```

Результат

```
{
  "etld": "com",
  "etld_plus": "com",
  "known_suffix": true
}
```

Разбор eTLD+1

Исходный код

```
parse_etld!("example.com", plus_parts: 1)
```

Результат

```
{
  "etld": "com",
  "etld_plus": "example.com",
  "known_suffix": true
}
```

Разбор eTLD с неизвестным суффиксом

Исходный код

```
parse_etld!("example.acmecorp")
```

Результат

```
{
  "etld": "acmecorp",
  "etld_plus": "acmecorp",
  "known_suffix": false
}
```

Разбор eTLD с другим PSL

Исходный код

```
parse_etld!("example.acmecorp",
# Пример пути к списку доменов.
psl: "resources/public_suffix_list.dat")
```

Результат

```
{
  "etld": "acmecorp",
  "etld_plus": "acmecorp",
  "known_suffix": false
}
```

dns_lookup

Выполняет DNS Lookup для доменного имени `value`.



Не рекомендуется применять `dns_lookup` в высоконагруженных сценариях. Функция выполняет сетевые запросы, используя однопоточный процесс, который блокирует поток выполнения при каждом запросе, пока не получит ответ, из-за чего производительность в высоконагруженных приложениях может деградировать.

Избегайте использования этой функции в контекстах, чувствительных к задержке, из-за возможных сетевых задержек или сбоев.

Спецификация функции

```
dns_lookup(value: <строка>, [qtype: <строка>, qclass: <строка>, options: <объект>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Доменное имя.		да
`qtype`	строка	Тип искомой записи DNS.		нет
`qclass`	строка	Класс сети, для которой выполняется поиск DNS-записей.	`IN`	нет
`options`	строка	Опции поиска.		нет

Ошибки

Функция `dns_lookup` **может возвращать ошибки**, для которых требуется обработка:

- Невозможно разобрать имя хоста `value`.
- Невозможно разобрать тип запроса `qtype`.
- Невозможно разобрать класс сети `qclass`.
- Не удалось выполнить запрос DNS Lookup.

Примеры

Базовый поиск

Исходный код

```
res = dns_lookup!("dns.google")
# Сбросить нестатический ttl, так что результат статический.
res.answers = map_values(res.answers) -> |value| {
  value.ttl = 600
  value
}
# Удалить дополнительные ответы.
res.answers = filter(res.answers) -> |_, value| {
  value.rData == "8.8.8.8"
}
# Удалить класс, поскольку он динамический.
```

```
res.additional = map_values(res.additional) -> |value| {
  del(value.class)
  value
}
res
```

Результат


```
{
  "additional": [
    {
      "domainName": "",
      "rData": "OPT ...",
      "recordType": "OPT",
      "recordTypeId": 41,
      "ttl": 0
    }
  ],
  "answers": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "rData": "8.8.8.8",
      "recordType": "A",
      "recordTypeId": 1,
      "ttl": 600
    }
  ],
  "authority": [],
  "fullRcode": 0,
  "header": {
    "aa": false,
    "ad": false,
    "anCount": 2,
    "arCount": 1,
    "cd": false,
    "nsCount": 0,
    "opcode": 0,
    "qdCount": 1,
    "qr": true,
    "ra": true,
    "rcode": 0,
    "rd": true,
    "tc": false
  },
  "question": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "questionType": "A",
```

```
      "questionTypeId": 1
    }
  ],
  "rcodeName": "NOERROR"
}
```

Заданные `qclass` и `qtype`

Исходный код

```
res = dns_lookup!("dns.google", class: "IN", qtype: "A")
# Сбросить нестатический ttl, так что результат статический.
res.answers = map_values(res.answers) -> |value| {
  value.ttl = 600
  value
}
# Удалить дополнительные ответы.
res.answers = filter(res.answers) -> |_, value| {
  value.rData == "8.8.8.8"
}
# Удалить класс, поскольку он динамический.
res.additional = map_values(res.additional) -> |value| {
  del(value.class)
  value
}
res
```

Результат

```
{
  "additional": [
    {
      "domainName": "",
      "rData": "OPT ...",
      "recordType": "OPT",
      "recordTypeId": 41,
      "ttl": 0
    }
  ],
  "answers": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "rData": "8.8.8.8",
      "recordType": "A",
      "recordTypeId": 1,
      "ttl": 600
    }
  ],
  "authority": [],
```

```
  "fullRcode": 0,
  "header": {
    "aa": false,
    "ad": false,
    "anCount": 2,
    "arCount": 1,
    "cd": false,
    "nsCount": 0,
    "opcode": 0,
    "qdCount": 1,
    "qr": true,
    "ra": true,
    "rcode": 0,
    "rd": true,
    "tc": false
  },
  "question": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "questionType": "A",
      "questionTypeId": 1
    }
  ],
  "rcodeName": "NOERROR"
}
```

Заданные опции

Исходный код

```
res = dns_lookup!("dns.google", options: {"timeout": 30, "attempts": 5})
# Сбросить нестатический ttl, так что результат статический.
res.answers = map_values(res.answers) -> |value| {
  value.ttl = 600
  value
}
# Удалить дополнительные ответы.
res.answers = filter(res.answers) -> |_, value| {
  value.rData == "8.8.8.8"
}
# Удалить класс, поскольку он динамический.
res.additional = map_values(res.additional) -> |value| {
  del(value.class)
value
}
res
```

Результат


```
{
  "additional": [
    {
      "domainName": "",
      "rData": "OPT ...",
      "recordType": "OPT",
      "recordTypeId": 41,
      "ttl": 0
    }
  ],
  "answers": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "rData": "8.8.8.8",
      "recordType": "A",
      "recordTypeId": 1,
      "ttl": 600
    }
  ],
  "authority": [],
  "fullRcode": 0,
  "header": {
    "aa": false,
    "ad": false,
    "anCount": 2,
    "arCount": 1,
    "cd": false,
    "nsCount": 0,
    "opcode": 0,
    "qdCount": 1,
    "qr": true,
    "ra": true,
    "rcode": 0,
    "rd": true,
    "tc": false
  },
  "question": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "questionType": "A",
      "questionTypeId": 1
    }
  ],
  "rcodeName": "NOERROR"
}
```

Заданный DNS-сервер

Исходный код

```
res = dns_lookup!("dns.google", options: {"servers": ["dns.quad9.net"]})
# Сбросить нестатический ttl, так что результат статический.
res.answers = map_values(res.answers) -> |value| {
  value.ttl = 600
  value
}
# Удалить дополнительные ответы.
res.answers = filter(res.answers) -> |_, value| {
  value.rData == "8.8.8.8"
}
# Удалить класс, поскольку он динамический.
res.additional = map_values(res.additional) -> |value| {
  del(value.class)
  value
}
res
```

Результат


```
{
  "additional": [
  {
    "domainName": "",
    "rData": "OPT ...",
    "recordType": "OPT",
    "recordTypeId": 41,
    "ttl": 0
  }
  ],
  "answers": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "rData": "8.8.8.8",
      "recordType": "A",
      "recordTypeId": 1,
      "ttl": 600
    }
  ],
  "authority": [],
  "fullRcode": 0,
  "header": {
  "aa": false,
  "ad": false,
  "anCount": 2,
  "arCount": 1,
  "cd": false,
```

```
  "nsCount": 0,
  "opcode": 0,
  "qdCount": 1,
  "qr": true,
  "ra": true,
  "rcode": 0,
  "rd": true,
  "tc": false
  },
  "question": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "questionType": "A",
      "questionTypeId": 1
    }
  ],
  "rcodeName": "NOERROR"
}
```

parse_glog

Разбирает `value` с использованием формата glog (Google Logging Library).

Спецификация функции

```
parse_glog(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Ошибки

Функция `parse_glog` **возвращает ошибку**, для которой требуется обработка, если `value` не соответствует формату `glog`.

Пример

Разбор через формат `glog`

Исходный код

```
parse_glog!("I20210131 14:48:54.411655 15520 main.c++:9] Hello world!")
```

Результат

```
{
```

```
  "file": "main.c++",
  "id": 15520,
  "level": "info",
  "line": 9,
  "message": "Hello world!",
  "timestamp": "2021-01-31T14:48:54.411655Z"
}
```

parse_grok

Разбирает `value`, используя формат `grok`. Поддерживаются все шаблоны grok.



Рекомендуется использовать шаблоны Grok, которые поддерживаются сообществом. Такие шаблоны, скорее всего, будут проверены и улучшены в будущем, в отличие от пользовательских шаблонов.

Спецификация функции

```
parse_grok(value: <строка>, pattern: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`pattern`	строка	Шаблон Grok.		да

Ошибки

Функция `parse_grok` **возвращает ошибку**, для которой требуется обработка, если `value` не удается разобрать с использованием предоставленного шаблона `pattern`.

Пример

Разбор с использованием Grok

Исходный код

```
parse_grok!(
"2020-10-02T23:22:12.223222Z info Hello world",
"%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:level} %{GREEDYDATA:message}"
)
```

Результат

```
{
  "level": "info",
  "message": "Hello world",
  "timestamp": "2020-10-02T23:22:12.223222Z"
```

```
}
```

parse_groks

Разбирает `value`, используя несколько шаблонов `grok`. Поддерживаются все шаблоны grok.



Рекомендуется использовать шаблоны Grok, которые поддерживаются сообществом. Такие шаблоны, скорее всего, будут проверены и улучшены в будущем, в отличие от пользовательских шаблонов.

Спецификация функции

```
parse_groks(value: <строка>, patterns: <массив>, [aliases: <объект>, alias_sources:
<массив>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`patterns`	массив	Шаблоны Grok, которые применяются последовательно до первого совпадения.		да
`aliases`	объект	Общий набор псевдонимов Grok, на которые можно ссылаться в шаблонах для их упрощения.		нет
`alias_sources`	массив строк	Пути к файлам, содержащим псевдонимы в формате JSON.		нет

Ошибки

Функция `parse_groks` **может возвращать ошибки**, для которых требуется обработка:

- `value` не удается разобрать с использованием предоставленных шаблонов;
- `patterns` не является массивом;
- `aliases` не является объектом;
- элемент массива `alias_sources` не является строкой или не указывает на корректный файл.

Пример

Разбор с использованием нескольких шаблонов Grok

Исходный код

```
parse_groks!(
"2020-10-02T23:22:12.223222Z info Hello world",
patterns: [
"%{common_prefix} %{_status} %{_message}",
```

```
"%{common_prefix} %{_message}",
],
aliases: {
"common_prefix": "%{_timestamp} %{_loglevel}",
"_timestamp": "%{TIMESTAMP_ISO8601:timestamp}",
"_loglevel": "%{LOGLEVEL:level}",
"_status": "%{POSINT:status}",
"_message": "%{GREEDYDATA:message}"
})
```

Результат

```
{
  "level": "info",
  "message": "Hello world",
  "timestamp": "2020-10-02T23:22:12.223222Z"
}
```

parse_influxdb

Разбирает `value` как строку протокола строк InfluxDB, создавая список метрик, совместимых с Vector.



- Эта функция вернет событие журнала с размерностью Vector-совместимой метрики, но не само событие метрики. Вероятно, вам потребуется обработать вывод этой функции преобразованием `log_to_metric` с опцией `all_metrics`, установленной в значение `true`, чтобы конвертировать события логов с размерностью метрик в события метрик и создать реальные метрики.
- Единственный создаваемый тип метрики — `gauge`. К имени каждой метрики добавляется префикс, состоящий из поля `measurement`, подчеркивания и поля `field key`.
- Строка — это единственный тип, который не поддерживается как значение поля из-за ограничений модели метрик Vector.

Спецификация функции

```
parse_influxdb(value: <строка> )
:: <массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление протокола строк InfluxDB для разбора.		да

Ошибки

Функция `parse_influxdb` **может возвращать ошибки**, для которых требуется обработка:

- `value` — не валидная строка протокола строк InfluxDB.
- Список полей содержит поле типа `строка`.
- Список полей содержит значение поля `NaN`.

Пример

Разбор протокола строк InfluxDB

Исходный код

```
parse_influxdb!("cpu,host=A,region=us-west
usage_system=64i,usage_user=10u,temperature=50.5,on=true,sleep=false
1590488773254420000")
```

Результат

```
[
  {
    "gauge": {
      "value": 64
    },
    "kind": "absolute",
    "name": "cpu_usage_system",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  },
  {
    "gauge": {
      "value": 10
    },
    "kind": "absolute",
    "name": "cpu_usage_user",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  },
  {
    "gauge": {
      "value": 50.5
    },
    "kind": "absolute",
    "name": "cpu_temperature",
    "tags": {
      "host": "A",
      "region": "us-west"
```

```
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  },
  {
    "gauge": {
      "value": 1
    },
    "kind": "absolute",
    "name": "cpu_on",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  },
  {
    "gauge": {
      "value": 0
    },
    "kind": "absolute",
    "name": "cpu_sleep",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  }
]
```

parse_klog

Разбирает `value` в формате klog, используемом компонентами Kubernetes.

Спецификация функции

```
parse_klog(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Ошибки

Функция `parse_klog` **возвращает ошибку**, для которой требуется обработка, если `value` не соответствует формату `klog`.

Пример

Разбор по формату klog

Исходный код

```
parse_klog!("I0505 17:59:40.692994 28133 klog.go:70] hello from klog")
```

Результат

```
{
  "file": "klog.go",
  "id": 28133,
  "level": "info",
  "line": 70,
  "message": "hello from klog",
  "timestamp": "2024-05-05T17:59:40.692994Z"
}
```

rv_parse_leef

Разбирает `value` в формате Log Event Extended Format (LEEF).

Спецификация функции

```
rv_parse_leef(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения LEEF		да

Ошибки

Функция `rv_parse_leef` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректным сообщением LEEF.

Пример

Сообщение LEEF с заголовком syslog и атрибутами события, разделенными символом ^

Исходный код

```
rv_parse_leef!("<13>1 2024-07-18T11:07:53.520Z 192.168.1.1
LEEF:2.0|Microsoft|MSExchange|2007|7732|^|dvchost=www.example.com^msg=io\t|p")
```

Результат

```
{
```

```
  "syslog_facility": "1",
  "syslog_priority": "13",
  "syslog_severity": "5",
  "at": "2024-07-18T11:07:53.520Z",
  "ahost": "192.168.1.1",
  "dvchost": "www.example.com",
  "msg": "io\t|p",
  "deviceVendor": "Microsoft",
  "eventName": "7732",
  "productName": "MSExchange",
  "productVersion": "2007",
}
```

parse_linux_authorization

Разбирает журналы авторизации Linux, которые обычно находятся в каталогах `/var/log/auth.log` (в системах на основе Debian) или `/var/log/secure` (в системах на основе RedHat), в соответствии с форматом syslog.



Если в сообщении не указан год, он определяется автоматически:

- Если текущий месяц — январь, а сообщение относится к декабрю, то используется предыдущий год.
- В противном случае используется текущий год.

Спецификация функции

```
parse_linux_authorization(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст, содержащий сообщение для разбора.		да

Ошибки

Функция `parse_linux_authorization` **возвращает ошибку**, для которой требуется обработка, если `value` не является правильно отформатированным сообщением syslog.

Пример

Разбор события авторизации Linux

Исходный код

```
parse_linux_authorization!(
s'Mar 23 01:49:58 localhost sshd[1111]: Accepted publickey for eng from
```

```
198.51.100.1 port 8888 ssh2: RSA SHA256:foobar'
)
```

Результат

```
{
  "appname": "sshd",
  "hostname": "localhost",
  "message": "Accepted publickey for eng from 198.51.100.1 port 8888 ssh2: RSA
SHA256:foobar",
  "procid": 1111,
  "timestamp": "2023-03-23T01:49:58Z"
}
```

parse_logfmt

Разбирает `value` в формате logfmt.

- Ключи и значения могут быть заключены в двойные кавычки `"`.
- Символ `"` может быть экранирован символом `\`.
- Согласно спецификации logfmt, ключами без значений присваивается логическое значение `true`.
- Функция обрабатывает три управляющие последовательности при разборе: `\n`, `\"` и `\\`. Это значит, что, например, `\n` во входной строке будет заменен на действительный символ новой строки в разобранных ключах или значениях.

Спецификация функции

```
parse_logfmt(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Ошибки

Функция `parse_logfmt` **возвращает ошибку**, для которой требуется обработка, если `value` не является правильно отформатированной строкой "ключ — значение" (logfmt).

Пример

Разбор лога в формате logfmt

Исходный код

```
parse_logfmt!(
"@timestamp=\"Sun Jan 10 16:47:39 EST 2021\" level=info msg=\"Stopping all fetchers\"
tag#production=stopping_fetchers id=ConsumerFetcherManager-1382721708341
```

```
module=kafka.consumer.ConsumerFetcherManager"
)
```

Результат

```
{
  "@timestamp": "Sun Jan 10 16:47:39 EST 2021",
  "id": "ConsumerFetcherManager-1382721708341",
  "level": "info",
  "module": "kafka.consumer.ConsumerFetcherManager",
  "msg": "Stopping all fetchers",
  "tag#production": "stopping_fetchers"
}
```

rv_parse_msgpack

Разбирает `value` в формате MessagePack.



- Возвращаются только типы MessagePack. Дополнительные типы `extension` не поддерживаются. Если нужно преобразовать строку в метку времени, используйте функцию `parse_timestamp`.
- В типе `map` поддерживаются только строковые ключи.

Спецификация функции

```
rv_parse_msgpack(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление MessagePack		да

Ошибки

Функция `rv_parse_msgpack` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является массивом байтов.

Примеры

Преобразование массива байтов

Исходный код

```
# .message -- массив байтов [135, 163, 105, 110, 116, 1, 165, 102, 108, 111, 97, 116,
203, 63, 224, 0, 0, 0, 0, 0, 0, 167, 98, 111, 111, 108, 101, 97, 110, 195, 164, 110,
117, 108, 108, 192, 166, 115, 116, 114, 105, 110, 103, 167, 102, 111, 111, 32, 98, 97,
114, 165, 97, 114, 114, 97, 121, 146, 163, 102, 111, 111, 163, 98, 97, 114, 166, 111,
```

```
98, 106, 101, 99, 116, 130, 163, 102, 111, 111, 1, 163, 98, 97, 122, 203, 63, 224, 0,
0, 0, 0, 0, 0,].
rv_parse_msgpack!(.message)
```

Результат

```
{
  "int": 1,
  "float": 0.5,
  "boolean": true,
  "null": null,
  "string": "foo bar",
  "array": [
    "foo",
    "bar"
  ],
  "object": {
    "foo": 1,
    "baz": 0.5
}
```

Преобразование после декодирования из Base64

Исходный код

```
rv_parse_msgpack!(decode_base64!("3wAAAAGlZmllbGSldmFsdWU="))
```

Результат

```
{
  "field": "value"
}
```

rv_parse_msgpack_batch

Разбирает `value`, которое состоит из нескольких сообщений в формате MessagePack.



- Возвращаются только типы MessagePack. Дополнительные типы (extension) не поддерживаются. Если нужно преобразовать строку в метку времени, используйте функцию `parse_timestamp`.
- В типе map поддерживаются только строковые ключи.

Спецификация функции

```
rv_parse_msgpack_batch(value: <строка>)
:: <массив объектов>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление пакета объектов MessagePack		да

Ошибки

Функция `rv_parse_msgpack_batch` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является массивом байтов.

Примеры

Преобразование массива байтов

Исходный код

```
# .message -- массив байтов [135, 163, 105, 110, 116, 1, 165, 102, 108, 111, 97, 116,
203, 63, 224, 0, 0, 0, 0, 0, 0, 167, 98, 111, 111, 108, 101, 97, 110, 195, 164, 110,
117, 108, 108, 192, 166, 115, 116, 114, 105, 110, 103, 167, 102, 111, 111, 32, 98, 97,
114, 165, 97, 114, 114, 97, 121, 146, 163, 102, 111, 111, 163, 98, 97, 114, 166, 111,
98, 106, 101, 99, 116, 130, 163, 102, 111, 111, 1, 163, 98, 97, 122, 203, 63, 224, 0,
0, 0, 0, 0, 0, 135, 163, 105, 110, 116, 1, 165, 102, 108, 111, 97, 116, 203, 63, 224,
0, 0, 0, 0, 0, 0, 167, 98, 111, 111, 108, 101, 97, 110, 195, 164, 110, 117, 108, 108,
192, 166, 115, 116, 114, 105, 110, 103, 167, 102, 111, 111, 32, 98, 97, 114, 165, 97,
114, 114, 97, 121, 146, 163, 102, 111, 111, 163, 98, 97, 114, 166, 111, 98, 106, 101,
99, 116, 130, 163, 102, 111, 111, 1, 163, 98, 97, 122, 203, 63, 224, 0, 0, 0, 0, 0,
0,].
rv_parse_msgpack_batch!(.message)
```

Результат

```
[
  {
    "int": 1,
    "float": 0.5,
    "boolean": true,
    "null": null,
    "string": "foo bar",
    "array": [
      "foo",
      "bar"
    ],
    "object": {
      "foo": 1,
      "baz": 0.5
    }
  },
  {
    "int": 1,
```

```
    "float": 0.5,
    "boolean": true,
    "null": null,
    "string": "foo bar",
    "array": [
      "foo",
      "bar"
    ],
    "object": {
      "foo": 1,
      "baz": 0.5
    }
  }
]
```

Преобразование после декодирования из Base64

Исходный код

```
rv_parse_msgpack_batch(decode_base64!("3wAAAAGlZmllbGSldmFsdWXfAAAAAaVmaWVsZKV2YWx1ZQ==
="))
```

Результат

```
[
  {
    "field": "value"
  },
  {
    "field": "value"
  }
]
```

rv_parse_netflow

Разбирает `value` в формате пакетов NetFlow.

Спецификация функции

```
rv_parse_netflow(value: <строка>)
:: <массив объектов>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление пакетов NetFlow.		да

Ошибки

Функция `rv_parse_netflow` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильным NetFlow пакетом

Пример

Разбор пакетов NetFlow

Исходный код

```
rv_parse_netflow!(decode_base64!("AAUCAAMABAAFAAYHCAkAAQIDBAUGBwgJAAECAwQFBgcICQABAgME
BQYHCAkAAQIDBAUGBwgJAAECAwQFBgcICQABAgMEBQYH"))
```

Результат

```
[
  {
    "V5": {
      "body": {
        "d_octets": 66051,
        "d_pkts": 101124105,
        "dst_addr": "4.5.6.7",
        "dst_as": 515,
        "dst_mask": 5,
        "dst_port": 1029,
        "first": {
          "nanos": 87000000,
          "secs": 67438
        },
        "input": 515,
        "last": {
          "nanos": 553000000,
          "secs": 134807
        },
        "next_hop": "8.9.0.1",
        "output": 1029,
        "pad1": 6,
        "pad2": 1543,
        "protocol_number": 8,
        "protocol_type": "EGP",
        "src_addr": "0.1.2.3",
        "src_as": 1,
        "src_mask": 4,
        "src_port": 515,
        "tcp_flags": 7,
        "tos": 9
      },
      "header": {
        "count": 512,
        "engine_id": 7,
        "engine_type": 6,
```

```
        "flow_sequence": 33752069,
        "sampling_interval": 2057,
        "sys_up_time": {
          "nanos": 672000000,
          "secs": 50332
        },
        "unix_nsecs": 134807553,
        "unix_secs": 83887623,
        "version": 5
      }
    }
  }
]
```

parse_nginx_log

Разбирает строки журналов доступа и ошибок Nginx. Строки могут быть в формате `combined`, `ingress_upstreaminfo`, `main` или `error`.



Отсутствие информации в сообщении журнала может быть обозначено `-`. Эти поля опускаются в результате, возвращаемом функцией.

Спецификация функции

```
parse_nginx_log(value: <строка>, format: <строка>, [timestamp_format: <строка>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`format`	строка	Формат, используемый для разбора журнала.		да
`timestamp_format`	строка	Формат даты и времени, используемый для кодирования метки времени. Если в метке времени не указан часовой пояс, время разбирается как локальное. Формат по умолчанию `%d/%b/%Y:%T %z` для комбинированных журналов и `%Y/%m/%d %H:%M:%S` для журналов ошибок.	`%d/%b/%Y:%T %z`	нет

Ошибки

Функция `parse_nginx_log` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует указанному формату.
- `timestamp_format` не является допустимой строкой формата.

- Метка времени в `value` не удается разобрать с использованием предоставленного формата `timestamp_format`.

Примеры

Разбор в формате журнала Nginx (комбинированном)

Исходный код

```
parse_nginx_log!(
s'203.0.113.1 - alice [01/Apr/2021:12:02:31 +0000] "POST /not-found HTTP/1.1" 404 153
"http://localhost/somewhere" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/72.0.3626.119 Safari/537.36" "2.75"',
"combined",
)
```

Результат

```
{
  "client": "203.0.113.1",
  "user": "alice",
  "timestamp": "2021-04-01T12:02:31Z",
  "request": "POST /not-found HTTP/1.1",
  "method": "POST",
  "path": "/not-found",
  "protocol": "HTTP/1.1",
  "status": 404,
  "size": 153,
  "referer": "http://localhost/somewhere",
  "agent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/72.0.3626.119 Safari/537.36",
  "compression": "2.75"
}
```

Разбор в формате журнала ошибок Nginx

Исходный код

```
parse_nginx_log!(
s'2021/04/01 13:02:31 [error] 31#31: *1 open() "/usr/share/nginx/html/not-found"
failed (2: No such file or directory), client: 203.0.113.1, server: localhost,
request: "POST /not-found HTTP/1.1", host: "localhost:8081"',
"error"
)
```

Результат

```
{
  "timestamp": "2021-04-01T13:02:31Z",
  "severity": "error",
```

```
  "pid": 31,
  "tid": 31,
  "cid": 1,
  "message": "open() \"/usr/share/nginx/html/not-found\" failed (2: No such file or
directory)",
  "client": "203.0.113.1",
  "server": "localhost",
  "request": "POST /not-found HTTP/1.1",
  "host": "localhost:8081"
}
```

parse_proto

Разбирает `value` как payload — полезную нагрузку Protocol Buffers.

Спецификация функции

```
parse_proto(value: <строка>, desc_file: <строка>, message_type: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Полезная нагрузка Protocol Buffers для разбора.		да
`desc_file`	строка	Путь к файлу набора дескрипторов Protocol Buffers. Должен быть строковым литералом. Этот файл является результатом работы команды `protoc -o`.		да
`message_type`	строка	Имя типа сообщения, используемого для сериализации. Должно быть строковым литералом.		да

Примечание

Функция разбирает и возвращает только сообщения (message) Protocol Buffers.

Ошибки

Функция `encode_proto` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является валидным payload Protocol Buffers.
- Файл, указанный в аргументе `desc_file`, не существует.
- Тип сообщения, указанный в аргументе `message_type`, отсутствует в файле дескрипторов.

Пример

Исходный код

```
.payload = encode_base64(encode_proto!({"name": "someone", "phones": [{"number":
"123456"}]}, "resources/protobuf_descriptor_set.desc", "test_protobuf.Person"))
```

Результат

```
Cgdzb21lb25lIggKBjEyMzQ1Ng==
```

rv_parse_qradar

Разбирает `value` в формате QRadar, включая необязательный заголовок syslog.

Спецификация функции

```
rv_parse_qradar(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения QRadar		да

Ошибки

Функция `rv_parse_qradar` **может возвращать ошибки**, для которых требуется обработка:

- `value` закодировано не в UTF-8;
- `value` содержит ключ без соответствующего значения.

Примеры

Разбор сообщения без заголовка syslog и со вложенным полем Message

Исходный код

```
rv_parse_msgpack!("AgentDevice=WindowsLog\tAgentLogFile=Security\tPluginVersion=7.2.9.
105\tSource=Microsoft-Windows-Security-
Auditing\tComputer=dc04.lab2012.local\tOriginatingComputer=192.0.2.0\tUser=\tDomain=\t
EventID=4624\tEventIDCode=4624\tEventType=8\tEventCategory=12544\tRecordNumber=1110397
1\tTimeGenerated=1610705793\tTimeWritten=1610705793\tLevel=Log Always\tKeywords=Audit
Success\tTask=SE_ADT_LOGON_LOGON\tOpcode=Info\tMessage=Вход с учетной записью выполнен
успешно.  Субъект:  ИД безопасности:  NULL SID  Имя учетной записи:  -  Домен учетной
записи:  -  Код входа:  0x0  Тип входа:  3  Уровень олицетворения:  Делегирование
Новый вход:  ИД безопасности:  NT AUTHORITY\\СИСТЕМА  Имя учетной записи:  USER  Домен
учетной записи:  LAB2012  Код входа:  0x1E6112B9  GUID входа:  {06457096-03A4-55FB-
28EA-9882B565E681}  Сведения о процессе:  Идентификатор процесса:  0x0  Имя процесса:
-  Сведения о сети:  Имя рабочей станции: -  Сетевой адрес источника:
```

```
fe80::7830:1b00:4e5f:1300  Порт источника:  51755  Данное событие возникает при
создании сеанса входа.\n".as_bytes())
```

Результат


```
{
  "AgentDevice": "WindowsLog",
  "AgentLogFile": "Security",
  "PluginVersion": "7.2.9.105",
  "Source": "Microsoft-Windows-Security-Auditing",
  "Computer": "dc04.lab2012.local",
  "OriginatingComputer": "192.0.2.0"
  "User": ""
  "Domain": ""
  "EventID": "4624"
  "EventIDCode": "4624"
  "EventType": "8"
  "EventCategory": "12544"
  "RecordNumber": "11103971"
  "TimeGenerated": "1610705793"
  "TimeWritten": "1610705793"
  "Level": "Log Always"
  "Keywords": "Audit Success"
  "Task": "SE_ADT_LOGON_LOGON"
  "Opcode": "Info"
  "Message": {
    "Вход с учетной записью выполнен успешно.": "",
    "Субъект": "ИД безопасности: NULL SID",
    "Имя учетной записи": "-",
    "Домен учетной записи": "-",
    "Код входа": "0x0",
    "Тип входа": "3",
    "Уровень олицетворения": "Делегирование",
    "Новый вход": {
      "ИД безопасности": "NT AUTHORITY\\СИСТЕМА",
      "Имя учетной записи": "USER",
      "Домен учетной записи": "LAB2012",
      "Код входа": "0x1E6112B9",
      "GUID входа": "{06457096-03A4-55FB-28EA-9882B565E681}"
    },
    "Сведения о процессе": {
      "Идентификатор процесса": "0x0",
      "Имя процесса": "-"
    },
    "Сведения о сети": {
      "Имя рабочей станции": "-",
      "Сетевой адрес источника": "2001:DB8::7830:1b00:4e5f:1300",
      "Порт источника": "51755"
    }
  }
```

```
}
```

Разбор сообщения с заголовком syslog

Исходный код

```
rv_parse_msgpack!("<13>Jan 15 13:16:35 DC04
AgentDevice=WindowsLog\tAgentLogFile=Security\tPluginVersion=7.2.9.105\tSource=Microso
ft-Windows-Security-
Auditing\tComputer=dc04.lab2012.local\tOriginatingComputer=192.0.2.0\tUser=\tDomain=\t
EventID=4624\tEventIDCode=4624\tEventType=8\tEventCategory=12544\tRecordNumber=1110397
1\tTimeGenerated=1610705793\tTimeWritten=1610705793\tLevel=Log Always\tKeywords=Audit
Success\tTask=SE_ADT_LOGON_LOGON\tOpcode=Info\n".as_bytes())
```

Результат

```
{
  "syslog_facility": "1",
  "syslog_priority": "13",
  "syslog_severity": "5",
  "at": "Jan 15 13:16:35",
  "ahost": "DC04",
  "AgentDevice": "WindowsLog",
  "AgentLogFile": "Security",
  "PluginVersion": "7.2.9.105",
  "Source": "Microsoft-Windows-Security-Auditing",
  "Computer": "dc04.lab2012.local",
  "OriginatingComputer": "192.0.2.0"
  "User": ""
  "Domain": ""
  "EventID": "4624"
  "EventIDCode": "4624"
  "EventType": "8"
  "EventCategory": "12544"
  "RecordNumber": "11103971"
  "TimeGenerated": "1610705793"
  "TimeWritten": "1610705793"
  "Level": "Log Always"
  "Keywords": "Audit Success"
  "Task": "SE_ADT_LOGON_LOGON"
  "Opcode": "Info"
}
```

parse_syslog

Разбирает `value` в формате syslog.



- Функция разбирает различные форматы syslog, такие как RFC 6587, RFC 5424, RFC 3164, стиль syslog в Nginx и другие распространенные варианты.

- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Спецификация функции

```
parse_syslog(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст, содержащий сообщение syslog для разбора.		да

Ошибки

Функция `parse_syslog` **возвращает ошибку**, для которой требуется обработка, если `value` не является правильно отформатированным сообщением syslog.

Пример

Разбор журнала Syslog (5424)

Исходный код

```
parse_syslog!(
s'<13>1 2020-03-13T20:45:38.119Z example.name non 2426 ID931 [exampleSDID@32473
iut="3" eventSource= "Application" eventID="1011"] Try to override the THX port, maybe
it will reboot the neural interface!'
)
```

Результат

```
{
  "severity": "notice",
  "facility": "user",
  "timestamp": "2020-03-13T20:45:38.119Z",
  "hostname": "example.name",
  "appname": "non",
  "procid": 2426,
  "msgid": "ID931",
  "message": "Try to override the THX port, maybe it will reboot the neural
interface!",
  "exampleSDID@32473": {
    "eventID": "1011",
    "eventSource": "Application",
    "iut": "3"
  },
  "version": 1
```

```
}
```

Функции преобразования кодов и параметров Syslog

to_syslog_facility

Преобразует `value` из кода устройства Syslog в соответствующее ему ключевое слово Syslog, например, `0` в `"kern"`, а `1` — в `"user"`.

Спецификация функции

```
to_syslog_facility(value: <целое число>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число	Код устройства.		да

Ошибки

Функция `to_syslog_facility` **возвращает ошибку**, для которой требуется обработка, если `value` не является допустимым кодом устройства Syslog.

Пример

Приведение к устройству Syslog

Исходный код

```
to_syslog_facility!(4)
```

Результат

```
auth
```

to_syslog_facility_code

Преобразует `value` из ключевого слова Syslog в соответствующий ему код устройства Syslog (от `0` до `23`).

Спецификация функции

```
to_syslog_facility_code(value: <строка>)
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Ключевое слово Syslog.		да

Ошибки

Функция `to_syslog_facility_code` **возвращает ошибку**, если `value` не является допустимым ключевым словом Syslog.

Пример

Приведение к ключевому слову Syslog

Исходный код

```
to_syslog_facility_code!("authpriv")
```

Результат

```
10
```

to_syslog_level

Преобразует `value` из кода уровня серьезности Syslog в соответствующее ключевое слово, например, `0` в `"emerg"`, `1` — в `"alert"`.

Спецификация функции

```
to_syslog_level(value: <целое число>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число	Код уровня серьезности.		да

Ошибки

Функция `to_syslog_level` **возвращает ошибку**, для которой требуется обработка, если `value` не является допустимым кодом уровня серьезности Syslog.

Пример

Приведение к уровню Syslog

Исходный код

```
to_syslog_level!(5)
```

Результат

```
notice
```

to_syslog_severity

Преобразует `value`, ключевое слово уровня Syslog, в целое число уровня серьезности Syslog (`0` до `7`).

Спецификация функции

```
to_syslog_severity(value: <строка>)
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Ключевое слово уровня Syslog для преобразования.		да

Ошибки

Функция `to_syslog_severity` **возвращает ошибку**, для которой требуется обработка, если `value` не является допустимым ключевым словом уровня Syslog.

Пример

Приведение к уровню серьезности Syslog

Исходный код

```
to_syslog_severity!("alert")
```

Результат

```
1
```

Функции разбора структурированных данных

parse_cbor

Разбирает `value` в формате CBOR.

Спецификация функции

```
parse_cbor(value: <строка>)
:: <двоичное значение | целое число | число с плавающей точкой | строка | объект |
массив | пустое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Полезная нагрузка CBOR для разбора.		да



Функция возвращает только типы CBOR.

Ошибки

Функция `parse_csv` **возвращает ошибку**, для которой требуется обработка, если `value` не является допустимой полезной нагрузкой CBOR.

Пример

Разбор CBOR

Исходный код

```
parse_cbor!(decode_base64!("oWVmaWVsZGV2YWx1ZQ=="))
```

Результат

```
{
  "field": "value"
}
```

parse_csv

Разбирает отдельную строку в формате CSV. Если входное значение состоит из нескольких строк, будет проанализирована только первая строка.



Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Спецификация функции

```
parse_csv(value: <строка>, [delimiter: <строка>])
:: <массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`delimiter`	строка	Разделитель полей для использования при разборе. Должен быть однобайтовым символом UTF-8.	`,`	нет

Ошибки

Функция `parse_csv` **может возвращать ошибки**, для которых требуется обработка:

- Разделитель не является однобайтовым символом UTF-8.
- `value` не является допустимой строкой CSV.

Примеры

Разбор отдельной строки в формате CSV

Исходный код

```
parse_csv!("foo,bar,\"foo \"\", bar\"")
```

Результат

```
[
  "foo",
  "bar",
  "foo \", bar"
]
```

Разбор отдельной строки в формате CSV с пользовательским разделителем

Исходный код

```
parse_csv!("foo bar", delimiter: " ")
```

Результат

```
[
  "foo",
  "bar"
]
```

rv_parse_html

Разбирает `value` в формате HTML.

Спецификация функции

```
rv_parse_html(value: <строка>, [trim: <логическое значение>, attr_prefix: <строка>,
text_key: <строка>, always_use_text_key: <логическое значение>, parse_bool:
<логическое значение>, parse_null: <логическое значение>, parse_number: <логическое
значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	HTML-документ в виде строки.		да
`trim`	логическое значение	Удалять пробелы вокруг элементов HTML.	`true`	нет
`include_attr`	логическое значение	Включать атрибуты HTML в возвращаемый объект.	`true`	нет
`attr_prefix`	строка	Префикс строки для ключей атрибутов HTML.	`"@"`	нет
`text_key`	строка	Ключ, под которым возвращается содержимое элементов HTML.	`"text"`	нет
`always_use_text_key`	логическое значение	Всегда возвращать содержимое элементов HTML под ключом, указанным в параметре `text_key`. Если `true`, содержимое будет сохраняться под отдельным ключом, даже если у элемента нет атрибутов.	`false`	нет
`parse_bool`	логическое значение	Интерпретировать строки `"true"` и `"false"` как логические значения.	`true`	нет
`parse_null`	логическое значение	Интерпретировать пустые строки и строки `"null"` как пустые значения.	`true`	нет
`parse_number`	логическое значение	Интерпретировать строковые представления чисел (`"1"`) как числовые типы.	`true`	нет

Ошибки

Функция `rv_parse_html` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректным документом HTML.

Примеры

С указанием text_key, без разбора числовых типов

Исходный код

```
value = s'<book category="PROGRAMMING"> <title lang="en">The Rust programming
language</title> <author>Steve Klabnik, Carol Nichols</author>
<year>2019</year></book>'
rv_parse_html!(value,
text_key: "value",
parse_number: false)
```

Результат

```
{
  "book": {
    "@category": "PROGRAMMING",
    "author": "Steve Klabnik, Carol Nichols",
    "title": {
      "@lang": "en",
      "value": "The Rust programming language"
    },
    "year": "2019"
  }
}
```

С использованием text_key во всех случаях

Исходный код

```
value = s'<b>test</b>'
rv_parse_html!(value,
text_key: "value",
text_key: "node",
always_use_text_key: true)
```

Результат

```
{
  "b": {
    "node": "test"
  }
}
```

Без удаления пробелов между элементами

Исходный код

```
value = s'<root>  <a>test</a>  </root>'
rv_parse_html!(value, trim: false)
```

Результат

```
{
  "root": {
    "a": "test",
    "text": [
      "  ",
      "  "
    ]
  }
```

```
}
```

parse_json

Разбирает `value` как JSON.



Возвращаются только типы JSON. Если нужно преобразовать `string` в `timestamp`, используйте функцию `parse_timestamp`.



Чтобы извлечь содержимое из поврежденного JSON, используйте функцию `rv_json_extract_payload`.

Спецификация функции

```
parse_json(value: <строка>, [max_depth: <целое число>, lossy: <логическое значение>])
:: <логическое значение | целое число | число с плавающей точкой | строка | объект |
массив | пустое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка с представлением JSON для разбора.		да
`max_depth`	целое число	Количество уровней для разбора вложенных JSON-документов. Значение должно быть в диапазоне от 1 до 128.		нет
`lossy`	логическое значение	Разбирать ли JSON способом с потерями. Если аргумент установлен в `true`, невалидные UTF-8 символы заменяются на символ Юникода `&#xFFFD;` (U+FFFD), иначе возвращается ошибка, если есть любые невалидные UTF-8 символы.	`true`	нет

Ошибки

Функция `parse_json` **возвращает ошибку**, для которой требуется обработка, если `value` не является допустимым JSON-представлением.

Примеры

Разбор JSON

Исходный код

```
parse_json!("{\"key\": \"val\"}")
```

Результат

```
{
  "key": "val"
}
```

Разбор JSON с ограничением глубины

Исходный код

```
parse_json!("{\"top_level\":{\"key\": \"val\"}}",
max_depth: 1)
```

Результат

```
{
  "top_level": "{\"key\": \"val\"}"
}
```

rv_json_extract_payload

Извлекает значение ключа `payload_key` из поврежденного или невалидного JSON.



Если указанный ключ встречается несколько раз, то будет извлечено значение только первого вхождения.

Спецификация функции

```
rv_json_extract_payload(value: <строка>, payload_key: <строка>)
:: <логическое значение | целое число | число с плавающей точкой | строка | объект |
массив | пустое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка с представлением JSON для извлечения		да
`payload_key`	строка	Ключ, значение которого нужно извлечь		да

Ошибки

Функция `rv_json_extract_payload` **может возвращать ошибку**, для которой требуется обработка, если `value` не содержит ключ `payload_key`.

Примеры

JSON с валидной структурой

Исходный код

```
rv_json_extract_payload(s'{"a": 123, "payload"    : "data"}',
payload_key: "payload")
```

Результат

```
"data"
```

JSON с невалидной структурой

Исходный код

```
rv_json_extract_payload(s'"asd":123,"payload" : "text"',
payload_key: "payload")
```

Результат

```
"text"
```

Несколько ключей `payload_key`:

Исходный код

```
rv_json_extract_payload(s'{ asd":123,"payload" : "first", "payload": "second" }',
payload_key: "payload")
```

Результат

```
"first"
```

parse_key_value

Разбирает `value` в формате "ключ — значение", также известном как logfmt. Ключи и значения могут быть заключены в кавычки `"`.



Символы `"` могут быть экранированы с помощью обратной косой черты `\`.

Все значения возвращаются как строки или как массивы строк для дублирующихся ключей. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Спецификация функции

```
parse_key_value(value: <строка>, [key_value_delimiter: <строка>, field_delimiter:
<строка>, whitespace: <строка>, accept_standalone_key: <логическое значение>])
```

```
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`key_value_delimiter`	строка	Строка, разделяющая ключ и значение.	`=`	нет
`field_delimiter`	строка	Строка, разделяющая каждую пару "ключ — значение".	(пробел)	нет
`whitespace`	строка	Обработка пробелов вокруг разделителя, заданного в параметре `key_value_delimiter`: • `lenient` — игнорировать пробелы; • `strict` — интерпретировать пробелы как обычные символы.	`lenient`	нет
`accept_standalone_key`	логическое значение	Разрешены ли ключи без значений. Если такие ключи разрешены, в возвращаемом объекте им будет присвоено значение `true`.	`true`	нет

Ошибки

Функция `parse_key_value` **возвращает ошибку**, для которой требуется обработка, если `value` не является правильно отформатированной строкой "ключ — значение".

Примеры

Разбор лога в формате logfmt

Исходный код

```
parse_key_value!(
"@timestamp=\"Sun Jan 10 16:47:39 EST 2021\" level=info msg=\"Stopping all fetchers\"
tag#production=stopping_fetchers id=ConsumerFetcherManager-1382721708341
module=kafka.consumer.ConsumerFetcherManager"
)
```

Результат

```
{
  "@timestamp": "Sun Jan 10 16:47:39 EST 2021",
  "level": "info",
  "msg": "Stopping all fetchers",
  "tag#production": "stopping_fetchers",
  "id": "ConsumerFetcherManager-1382721708341",
  "module": "kafka.consumer.ConsumerFetcherManager"
```

```
}
```

Разбор лога с разделителем запятой

Исходный код

```
parse_key_value!(
"path:\"/cart_link\", host:store.app.com, fwd: \"203.0.113.16\", dyno: web.1,
connect:0ms, service:87ms, status:304, bytes:632, protocol:https",
field_delimiter: ",",
key_value_delimiter: ":"
)
```

Результат

```
{
  "path": "/cart_link",
  "host": "store.app.com",
  "fwd": "203.0.113.16",
  "dyno": "web.1",
  "connect": "0ms",
  "service": "87ms",
  "status": "304",
  "bytes": "632",
  "protocol": "https"
}
```

Разбор журнала с ключами без значений

Исходный код

```
parse_key_value!(
"env:prod,service:backend,region:eu-east1,beta",
field_delimiter: ",",
key_value_delimiter: ":",
)
```

Результат

```
{
  "env": "prod",
  "service": "backend",
  "region": "eu-east1",
  "beta": true
}
```

Разбор дублирующихся ключей

Исходный код

```
parse_key_value!(
"at=info,method=GET,path=\"/index\",status=200,tags=dev,tags=dummy",
field_delimiter: ",",
key_value_delimiter: "=",
)
```

Результат

```
{
  "at": "info",
  "method": "GET",
  "path": "/index",
  "status": "200",
  "tags": [
    "dev",
    "dummy"
  ]
}
```

parse_ruby_hash

Разбирает `value` как хэш Ruby.



Возвращаются только типы Ruby. Если нужно преобразовать `строку` в `метку времени`, используйте функцию `parse_timestamp`.

Спецификация функции

```
parse_ruby_hash(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление хэша Ruby для разбора		да

Ошибки

Функция `parse_ruby_hash` **возвращает ошибку**, для которой требуется обработка, если `value` не является допустимым представлением хэша Ruby.

Пример

Разбор хэша Ruby

Исходный код

```
parse_ruby_hash!(s'{ "test" => "value", "testNum" => 0.2, "testObj" => { "testBool" =>
true, "testNull" => nil } }')
```

Результат

```
{
  "test": "value",
  "testNum": 0.2,
  "testObj": {
    "testBool": true,
    "testNull": null
  }
}
```

parse_xml

Разбирает `value` как XML.



Допустимый XML должен содержать ровно один корневой узел. Функция всегда возвращает объект.

Спецификация функции

```
parse_xml(value: <строка>, [include_attr: <логическое значение>, attr_prefix:
<строка>, text_key: <строка>, always_use_text_key: <логическое значение>, parse_bool:
<логическое значение>, parse_null: <логическое значение>, parse_number: <логическое
значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление XML-документа для разбора.		да
`include_attr`	логическое значение	Включать атрибуты XML-тегов в возвращаемый объект.	`true`	нет
`attr_prefix`	строка	Префикс строки для ключей атрибутов XML-тегов.	`@`	нет
`text_key`	строка	Имя ключа для расширенных текстовых узлов.	`text`	нет
`always_use_text_key`	логическое значение	Всегда возвращать узлы текста как `{"<text_key>": "value"}`.	`false`	нет
`parse_bool`	логическое значение	Разбирать "true" и "false" как логические значения.	`true`	нет

Аргумент	Тип	Описание	По умолчанию	Обязателен
`parse_null`	логическое значение	Разбирать "null" как `null`.	`true`	нет
`parse_number`	логическое значение	Разбирать числа как целые или вещественные.	`true`	нет

Ошибки

Функция `parse_xml` **возвращает ошибку**, для которой требуется обработка, если `value` не является правильным XML-документом

Пример

Разбор XML

Исходный код

```
value = s'<book category="CHILDREN"><title lang="en">Harry Potter</title><author>J K.
Rowling</author><year>2005</year></book>'

parse_xml!(value, text_key: "value", parse_number: false)
```

Результат

```
{
  "book": {
    "@category": "CHILDREN",
    "author": "J K. Rowling",
    "title": {
      "@lang": "en",
      "value": "Harry Potter"
    },
    "year": "2005"
  }
}
```

Функции разбора метаданных и параметров

parse_bytes

Разбирает `value` как количество информации в указанных единицах измерения.

Спецификация функции

```
parse_bytes(value: <строка>, unit: <строка>, [base: <строка>])
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Количество информации в двоичных единицах измерения либо единицах СИ.		да
`unit`	строка	Единицы, в которые нужно перевести объем.		да
`base`	строка	Система приставок, используемая при неоднозначных обозначениях: • "2" — двоичная; • "10" — десятичная.	`"2"`	нет

Используемые обозначения представлены в теблице:

Обозначение	Десятичные единицы	Двоичные единицы
`B`	байт	
`kB`	килобайт	
`KiB`		кибибайт
`MB`	мегабайт	мебибайт (`MiB`)
`GB`	гигабайт	гибибайт (`GiB`)
`TB`	терабайт	тебибайт (`TiB`)
`PB`	петабайт	пебибайт (`PiB`)
`EB`	эксабайт	эксбибайт (`EiB`)

Ошибки

Функция `parse_bytes` **возвращает ошибку**, если `value` не является объемом информации в правильном формате.

Примеры

Перевод кибибайтов в число мебибайт

Исходный код

```
parse_bytes!("1024KiB", unit: "MiB")
```

Результат

```
1
```

Перевод терабайтов в число мегабайт

Исходный код

```
parse_bytes!("4TB", unit: "MB", base: "10")
```

```
4000000
```

Перевод ГБ в число байт с указанием используемой системы

Исходный код

```
parse_bytes!("1GB", unit: "B", base: "2")
```

Результат

```
1073741824
```

parse_duration

Разбирает `value` в удобочитаемый формат длительности, используя единицы, указанные в параметре `unit`.

Спецификация функции

```
parse_duration(value: <строка>, unit: <строка>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка с длительностью. Должна состоять из пар чисел и единиц измерения.		да
`unit`	строка	Единица измерения для вывода длительности.		да

Используемые единицы измерения:

- `ns` — наносекунды;
- `us`, `µs` — микросекунды;
- `ms` — миллисекунды;
- `cs` — сантисекунды;
- `ds` — децисекунды;
- `s` — секунды;
- `m` — минуты;

- `h` — часы;
- `d` — сутки.

Ошибки

Функция `parse_duration` **возвращает ошибку**, для которой требуется обработка, если `value` не имеет правильного формата длительности

Примеры

Преобразование миллисекунд в секунды

Исходный код

```
parse_duration!("1005ms", unit: "s")
```

Результат

```
1.005
```

Преобразование секунд и минут в миллисекунды

Исходный код

```
parse_duration!("1s1m", unit: "ms")
```

Результат

```
61000
```

parse_query_string

Разбирает `value` как строку запроса (query string).



- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.
- Пустые ключи и значения разрешены.

Спецификация функции

```
parse_query_string(value: <строка>)
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Примеры

Разбор строки запроса

Исходный код

```
parse_query_string("foo=%2B1&bar=2&bar=3&xyz")
```

Результат

```
{
  "bar": [
    "2",
    "3"
  ],
  "foo": "+1",
  "xyz": ""
}
```

Разбор строки запроса Ruby on Rails

Исходный код

```
parse_query_string("?foo%5b%5d=1&foo%5b%5d=2")
```

Результат

```
{
  "foo[]": [
    "1",
    "2"
  ]
}
```

parse_tokens

Разбирает `value` в формате "token". Токенами считаются такие последовательности символов:

- Слова, окруженные пробелами.
- Тексты, заключенные в двойные кавычки: `".."`. Кавычки можно включить в токен, если они экранированы обратной косой чертой (\).
- Тексты, заключенные в квадратные скобки: `[..]`. Закрывающие квадратные скобки можно включить в токен, если они экранированы обратной косой чертой (\).



Все значения токенов возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Спецификация функции

```
parse_tokens(value: <строка>)
:: <массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для токенизации.		да

Ошибки

Функция `parse_tokens` **возвращает ошибку**, для которой требуется обработка, если `value` не является правильно отформатированной строкой с токенами.

Пример

Разбор токенов

Исходный код

```
parse_tokens(
"A sentence \"with \\\"a\\\" sentence inside\" and [some brackets]"
)
```

Результат

```
[
  "A",
  "sentence",
  "with \\\"a\\\" sentence inside",
  "and",
  "some brackets"
]
```

parse_url

Разбирает `value` в формате URL.



По умолчанию, если во входном URL указан стандартный порт, он не учитывается. Поле `port` в возвращаемом объекте остается пустым.

Спецификация функции

```
parse_url(value: <строка>, [default_known_ports: <логическое значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст URL.		да
`default_known_ports`	логическое значение	Заполнять ли номер порта значением по умолчанию. Если во входном URL не указан номер порта или он совпадает со стандартным, поле будет заполнено значением по умолчанию. Это применимо к протоколам `http`, `https`, `ftp`, `ws` и `wss`.	`false`	нет

Ошибки

Функция `parse_url` **возвращает ошибку**, для которой требуется обработка, если `value` не является правильно отформатированным URL.

Примеры

Разбор URL

Исходный код

```
parse_url!("ftp://foo:bar@example.com:4343/foobar?hello=world#123")
```

Результат

```
{
  "scheme": "ftp",
  "username": "foo",
  "password": "bar",
  "host": "example.com",
  "port": 4343,
  "path": "/foobar",
  "query": {
    "hello": "world"
  },
  "fragment": "123"
}
```

Разбор URL с портом по умолчанию

Исходный код

```
parse_url!("https://example.com", default_known_ports: true)
```

Результат

```
{
  "scheme": "https",
  "username": "",
```

```
  "password": "",
  "host": "example.com",
  "port": 443,
  "path": "/",
  "query": {},
  "fragment": null
}
```

parse_user_agent

Разбирает `value` как строку пользовательского агента. Формат пользовательского агента имеет нечеткое определение, поэтому функция обеспечивает лишь максимально возможные результаты, а не точный разбор.



- Все значения возвращаются как строки или как `null`. Рекомендуется преобразовывать строки в нужные типы, используя соответствующие функции.
- Полям, которые не были разобраны, присваивается значение `null`.
- Разные режимы возвращают разные схемы.

Спецификация функции

```
parse_user_agent(value: <строка>, [mode: <строка>])
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`mode`	строка	Определяет характеристики производительности и надежности: • `fast` — самый быстрый, но менее надежный режим. Используется синтаксический анализатор из проекта Woothee. • `reliable` — обеспечивает большую надежность, чем режим `fast`, и в большинстве случаев не снижает скорость работы. Используется синтаксический анализатор Wood These. Если Wood These не разобрал какие-либо поля, используется анализатор uapproject. • `enriched` — объединяет результаты Wood These и uapproject.	`fast`	нет

Примеры

Быстрый режим

Исходный код

```
parse_user_agent(
"Mozilla Firefox 1.0.1 Mozilla/5.0 (X11; U; Linux i686; de-DE; rv:1.7.6)
Gecko/20050223 Firefox/1.0.1"
)
```

Результат

```
{
  "browser": {
    "family": "Firefox",
    "version": "1.0.1"
  },
  "device": {
    "category": "pc"
  },
  "os": {
    "family": "Linux",
    "version": null
  }
}
```

Надежный режим

Исходный код

```
parse_user_agent(
"Mozilla/4.0 (compatible; MSIE 7.66; Windows NT 5.1; SV1; .NET CLR 1.1.4322)",
mode: "reliable"
)
```

Результат

```
{
  "browser": {
    "family": "Internet Explorer",
    "version": "7.66"
  },
  "device": {
    "category": "pc"
  },
  "os": {
    "family": "Windows XP",
    "version": "NT 5.1"
  }
}
```

Расширенный режим

Исходный код

```
parse_user_agent(
"Opera/9.80 (J2ME/MIDP; Opera Mini/4.3.24214; iPhone; CPU iPhone OS 4_2_1 like Mac OS
X; AppleWebKit/24.783; U; en) Presto/2.5.25 Version/10.54",
mode: "enriched"
)
```

Результат

```
{
  "browser": {
    "family": "Opera Mini",
    "major": "4",
    "minor": "3",
    "patch": "24214",
    "version": "10.54"
  },
  "device": {
    "brand": "Apple",
    "category": "smartphone",
    "family": "iPhone",
    "model": "iPhone"
  },
  "os": {
    "family": "iOS",
    "major": "4",
    "minor": "2",
    "patch": "1",
    "patch_minor": null,
    "version": "4.2.1"
  }
}
```

Функции преобразования IP

ip_aton

Преобразует IPv4-адрес в представлении чисел и точек в байты, представленные в сетевом порядке, как целое число.

Функция повторяет поведение функции inet_aton.

Спецификация функции

```
ip_aton(value: <строка>)
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	IP-адрес для преобразования в бинарный формат.		да

Ошибки

Функция `ip_aton` **возвращает ошибку**, для которой требуется обработка, если `value` не является допустимым IPv4-адресом.

Пример

IPv4 в целое число

Исходный код

```
ip_aton!("233.252.0.4")
```

Результат

```
3925606404
```

ip_cidr_contains

Определяет, содержится ли `ip` в блоке, заданном с помощью `cidr`.

Спецификация функции

```
ip_cidr_contains(cidr: <строка | массив>, ip: <строка>)
:: <логическое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`cidr`	строка массив	Маска CIDR (v4 или v6).		да
`ip`	строка	IP-адрес (v4 или v6).		да

Ошибки

Функция `ip_cidr_contains` **может возвращать ошибки**, для которых требуется обработка:

- `cidr` не является допустимой маской CIDR.
- `ip` не является допустимым IP-адресом.

Примеры

IPv4 содержит CIDR

Исходный код

```
ip_cidr_contains!("203.0.113.0/24", "203.0.113.32")
```

Результат

```
true
```

IPv4 является частным

Исходный код

```
ip_cidr_contains!(["10.0.0.0/8", "172.16.0.0/12", "192.168.0.0/16"], "192.168.10.32")
```

Результат

```
true
```

Исходный код

```
ip_cidr_contains!("2001:db8:4:ba::/64", "2001:db8:4:ba:2e0:81ff:fe22:d1f1")
```

Результат

```
true
```

ip_ntoa

Преобразует числовое представление IPv4-адреса в байтах, представленных в сетевом порядке, в представление чисел и точек.

Функция повторяет поведение функции inet_ntoa.

Спецификация функции

```
ip_ntoa(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Целочисленное представление IPv4-адреса.		да

Ошибки

Функция `ip_ntoa` **возвращает ошибку**, для которой требуется обработка, если `value` не может быть

представлено в u32.

Пример

Целое число в IPv4

Исходный код

```
ip_ntoa!(3925606404)
```

Результат

```
233.252.0.4
```

ip_ntop

Преобразует IPv4 и IPv6-адреса из бинарной формы в текстовую форму.

Функция повторяет поведение функции inet_ntop.



Двоичные данные для этой функции не всегда легко отобразить в печатном виде. Однако результаты функций, таких как `decode_base64` или `decode_percent`, все равно можно использовать корректно.

Спецификация функции

```
ip_ntop(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Двоичные данные для преобразования. Для IPv4-адресов длина должна быть 4 байта (32 бита). Для IPv6-адресов длина должна быть 16 байт (128 бит).		да

Ошибки

Функция `ip_ntop` **возвращает ошибку**, для которой требуется обработка, если `value` должно быть длиной 4 или 16 байт

Примеры

Преобразование IPv4-адреса из байтов после декодирования из Base64

Исходный код

```
ip_ntop!(decode_base64!("wKgAAQ=="))
```

Результат

```
192.168.0.1
```

Преобразование IPv6-адреса из байтов после декодирования из Base64

Исходный код

```
ip_ntop!(decode_base64!("IAENuIWjAAAAAIouA3BzNA=="))
```

Результат

```
2001:db8:85a3::8a2e:370:7334
```

ip_pton

Преобразует IPv4 и IPv6-адреса из текстовой формы в бинарную форму.

- Бинарная форма IPv4-адресов имеет длину 4 байта (32 бита).
- Бинарная форма IPv6-адресов имеет длину 16 байт (128 бит).

Функция повторяет поведение функции inet_pton.



Двоичные данные из этой функции не всегда могут быть удобочитаемыми. Однако результаты функций, таких как `encode_base64` или `encode_percent`, все равно можно корректно обрабатывать.

Спецификация функции

```
ip_pton(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	IP-адрес (v4 или v6) для преобразования в бинарную форму.		да

Ошибки

Функция `ip_pton` **возвращает ошибку**, для которой требуется обработка, если `value` не является допустимым IP-адресом (v4 или v6) в текстовой форме.

Примеры

Преобразование IPv4-адреса в байты и кодирование в Base64

Исходный код

```
encode_base64(ip_pton!("192.168.0.1"))
```

Результат

```
wKgAAQ==
```

Преобразование IPv6-адреса в байты и кодирование в Base64

Исходный код

```
encode_base64(ip_pton!("2001:db8:85a3::8a2e:370:7334"))
```

Результат

```
IAENuIWjAAAAAIouA3BzNA==
```

ip_subnet

Извлекает подсетевой адрес из `ip` с использованием заданной подсети (`subnet`).



Функция работает как с IPv4-, так и с IPv6 адресами. Версия IP для маски должна быть такой же, как у предоставленного адреса.

Спецификация функции

```
ip_subnet(ip: <строка>, subnet: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	Обязателен
`ip`	строка	IP-адрес (v4 или v6).	да
`subnet`	строка	Подсеть, которую необходимо извлечь из IP-адреса в одном из представлений: • длина префикса, например, `/8`; • сетевая маска — IPv4- или IPv6-адрес, например, 255.255.0.0.	да

Ошибки

Функция `ip_subnet` **может возвращать ошибки**, для которых требуется обработка:

- `ip` не является допустимым IP-адресом.
- `subnet` не является допустимой подсетью.

Примеры

Подсеть IPv4

Исходный код

```
ip_subnet!("192.168.10.32", "255.255.255.0")
```

Результат

```
192.168.10.0
```

Подсеть IPv6

Исходный код

```
ip_subnet!("2404:6800:4003:c02::64", "/32")
```

Результат

```
2404:6800::
```

ip_to_ipv6

Преобразует `ip` в адрес IPv6.

Спецификация функции

```
ip_to_ipv6(ip: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`ip`	строка	IP-адрес, который необходимо преобразовать в IPv6.		да

Ошибки

Функция `ip_to_ipv6` **возвращает ошибку**, для которой требуется обработка, если `ip` не является допустимым IP-адресом.

Пример

IPv4 в IPv6

Исходный код

```
ip_to_ipv6!("192.168.10.32")
```

Результат

```
::ffff:192.168.10.32
```

ipv6_to_ipv4

Преобразует `ip` в адрес IPv4. Если `ip` уже является IPv4-адресом, то он возвращается без изменений. Если `ip` является IPv6-адресом, то он должен быть совместим с IPv4, в противном случае возникнет ошибка.

Спецификация функции

```
ipv6_to_ipv4(ip: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`ip`	строка	IPv4-совместимый IPv6-адрес для преобразования.		да

Ошибки

Функция `ipv6_to_ipv4` **может возвращать ошибки**, для которых требуется обработка:

- `ip` не является допустимым IP-адресом.
- `ip` является IPv6-адресом, который не совместим с IPv4.

Пример

IPv6 в IPv4

Исходный код

```
ipv6_to_ipv4!("::ffff:192.168.0.1")
```

Результат

```
192.168.0.1
```

rv_ipv6_to_ipv4

Преобразует `value` из адреса IPv6 в адрес IPv4. Если `value` уже является IPv4-адресом или выполнить преобразование из IPv6 невозможно, возвращается значение без изменений. Если `value` не является

IP-адресом, функция возвращает пустую строку.

Спецификация функции

```
rv_ipv6_to_ipv4(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	Обязателен
`value`	строка	IPv6-адрес для преобразования в IPv4.	да

Примеры

Адрес IPv4, отображенный в IPv6

Исходный код

```
rv_ipv6_to_ipv4!("::ffff:192.168.0.1")
```

Результат

```
"192.168.0.1"
```

Адрес IPv6, совместимый с IPv4

Исходный код

```
rv_ipv6_to_ipv4!("0:0:0:0:0:ffff:c633:6410")
```

Результат

```
"198.51.100.16"
```

Адрес IPv6, несовместимый с IPv4

Исходный код

```
rv_ipv6_to_ipv4!("2001:0db8:85a3::8a2e:0370:7334")
```

Результат

```
"2001:0db8:85a3::8a2e:0370:7334"
```

Адрес IPv4

Исходный код

```
rv_ipv6_to_ipv4!("198.51.100.16")
```

Результат

```
"198.51.100.16"
```

Невалидный адрес IP

Исходный код

```
rv_ipv6_to_ipv4!("i am not an ipaddress")
```

Результат

```
""
```

Функции кодирования и декодирования



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции декодирования

decode_base16

Декодирует `value` (строку в формате Base16) в исходную строку.

Спецификация функции

```
decode_base16(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Base16 для декодирования.		да

Ошибки

Функция `decode_base16` **возвращает ошибку**, для которой требуется обработка, если `value` не является корректной закодированной строкой в формате Base16.

Пример

Декодирование данных в формате Base16

Исходный код

```
decode_base16!("796f7520686176652073756363657373366756c6c79206465636f646564206d65")
```

Результат

```
"you have successfully decoded me"
```

decode_base64

Декодирует `value` (строка в формате Base64) в исходную строку.

Спецификация функции

```
decode_base64(value: <строка>, [charset: <строка>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Base64 для декодирования.		да
`charset`	строка	Набор символов, используемый при декодировании данных: • `standard` — стандартный формат Base64; • `url_safe` — измененный Base64 для URL.	`standard`	нет

Ошибки

Функция `decode_base64` **возвращает ошибку**, для которой требуется обработка, если `value` не является корректной закодированной строкой в формате Base64.

Примеры

Декодирование данных в формате Base64 (по умолчанию)

Исходный код

```
decode_base64!("eW91IGhhdmUgc3VjY2Vzc2Z1bGx5IGRlY29kZWQgbWU=")
```

Результат

```
"you can't make your heart feel something it won't"
```

Декодирование данных в измененном формате Base64 для URL

Исходный код

```
decode_base64!("eW91IGNhbid0IG1ha2UgeW91ciBoZWFydCBmZWVsIHNvbWV0aGluZyBpdCB3b24ndA==",
charset: "url_safe")
```

Результат

```
"you can't make your heart feel something it won't"
```

decode_charset

Преобразует строку, закодированную не в UTF-8, в строку UTF-8, используя заданную кодировку.

Спецификация функции

```
decode_charset(value: <строка>, from_charset: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка не в UTF-8, которую нужно декодировать.		да
`from_charset`	строка	Набор символов, используемый для декодирования. Допустимые значения представлены в спецификации Encoding Standard.		да

Ошибки

Функция `decode_charset` **возвращает ошибку**, для которой требуется обработка, если `from_charset` не является корректным названием кодировки.

Примеры

Декодирование строки в кодировке EUC-KR

Исходный код

```
decode_charset!(decode_base64!("vsiz58fPvLy/5A=="), "euc-kr")
```

Результат

```
"□□□□□"
```

Декодирование строки в кодировке EUC-JP

Исходный код

```
decode_charset!(decode_base64!("pLOk86TLpMGkzw=="), "euc-jp")
```

Результат

```
"こんにちは"
```

Декодирование строки в кодировке GB2312

Исходный код

```
decode_charset!(decode_base64!("xOO6ww=="), "gb2312")
```

Результат

```
"□好"
```

decode_gzip

Декодирует `value` (строка в формате Gzip) в исходную строку.

Спецификация функции

```
decode_gzip(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Gzip для декодирования.		да

Ошибки

Функция `decode_gzip` **возвращает ошибку**, для которой требуется обработка, если `value` не является корректной строкой, закодированной в формате Gzip.

Пример

Декодирование данных в формате Gzip

Исходный код

```
encoded_text =
decode_base64!("H4sIAHEAymMAA6vML1XISCxLVSguTU5OLS5OK83JqVRISU3OT0lNUchNBQD7BGDaIAAAAA
==")
decode_gzip!(encoded_text)
```

Результат

```
you have successfully decoded me
```

rv_decode_lz4

Декодирует `value` как запись, сжатую с помощью алгоритма LZ4.

Спецификация функции

```
rv_decode_lz4(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Запись, сжатая алгоритмом LZ4.		да

Ошибки

Функция `rv_decode_lz4` **возвращает ошибку**, для которой требуется обработка, если не удается декодировать запись как LZ4.

Пример

Исходный код

```
rv_decode_lz4!(decode_base64!("CQAAAJBlbmNvZGVfbWU="))
```

Результат

```
"encode_me"
```

decode_mime_q

Заменяет подстроки `value`, закодированные в кодировке q или base64, на исходные строки.

Спецификация функции

```
decode_mime_q(value: <строка>)
```

```
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Закодированная строка с кодировочным словом для декодирования.		да

Ошибки

Функция `decode_mime_q` **может возвращать ошибки**, для которых требуется обработка:

- `value` содержит некорректную закодированную строку с кодировочными словами.

Примеры

Декодирование одиночной с кодировочными словами

Исходный код

```
decode_mime_q!("=?utf-8?b?SGVsbG8sIFdvcmxkIQ==?=")
```

Результат

```
"Hello, World!"
```

Встроенная

Исходный код

```
decode_mime_q!("From: =?utf-8?b?SGVsbG8sIFdvcmxkIQ==?= <=?utf-
8?q?hello=5Fworld=40example=2ecom?=>")
```

Результат

```
"From: Hello, World! <hello_world@example.com>"
```

Без указания набора символов

Исходный код

```
decode_mime_q!("?b?SGVsbG8sIFdvcmxkIQ==")
```

Результат

```
"Hello, World!"
```

decode_percent

Декодирует строку `value`, закодированную в формате percent, как URL.

Спецификация функции

```
decode_percent(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для декодирования.		да

Пример

Декодирование строки в процентном кодировании

Исходный код

```
decode_percent("foo%20bar%3F")
```

Результат

```
"foo bar?"
```

decode_punycode

Декодирует строку `value`, закодированную в формате punycode, как IDN (интернационализированное доменное имя).

Спецификация функции

```
decode_punycode(value: <строка>, [validate: <логическое значение>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для декодирования.		да
`validate`	логическое значение	Если `true`, проверяет, является ли входная строка валидным именем домена.	`true`	нет

Ошибки

Функция `decode_punycode` **возвращает ошибку**, для которой требуется обработка, если `value` не является корректной строкой punycode.

Примеры

Декодирование интернационализированного доменного имени, закодированного в punycode

Исходный код

```
decode_punycode!("xn--e1afmkfd.xn--80akhbyknj4f")
```

Результат

```
"пример.испытание"
```

Декодирование строки из символов ASCII

Исходный код

```
decode_punycode!("example.com")
```

Результат

```
"example.com"
```

Декодирование без валидации

Исходный код

```
decode_punycode!("xn--8hbb.xn--fiba.xn--8hbf.xn--eib.", validate: false)
```

Результат

```
"□□.□□.□□.□."
```

decode_snappy

Декодирует `value`, представленное в формате Snappy, в исходную строку.

Спецификация функции

```
decode_snappy(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные Snappy для декодирования.		да

Ошибки

Функция `decode_snappy` **возвращает ошибку**, для которой требуется обработка, если `value` не

является корректной строкой, закодированной в формате Snappy.

Пример

Исходный код

```
encoded_text = decode_base64!("G2hBIHVzZXIgYWNjb3VudCB3YXMgY3JlYXRlZC4=")
decode_snappy!(encoded_text)
```

Результат

```
"A user account was created."
```

decode_zlib

Декодирует `value` (строка в формате Zlib) в исходную строку.

Спецификация функции

```
decode_zlib(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Zlib для декодирования.		да

Ошибки

Функция `decode_zlib` **возвращает ошибку**, для которой требуется обработка, если `value` не является корректной закодированной строкой в формате Zlib.

Пример

Декодирование данных в формате Zlib

Исходный код

```
encoded_text =
decode_base64!("eJwNy4ENwCAIBMCNXIlQ/KqplUSgCdvXAS41qPMHshCB2R1zJlWIVlR6UURX2+wx2YcuK3
kAb9C1wd6dn7Fa+QH9gRxr")
decode_zlib!(encoded_text)
```

Результат

```
"you have successfully decoded me"
```

decode_zstd

Декодирует `value` (строка в формате Zstandard) в исходную строку.

Спецификация функции

```
decode_zstd(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Zstandard для декодирования.		да

Ошибки

Функция `decode_zstd` **возвращает ошибку**, для которой требуется обработка, если `value` не является корректной закодированной строкой в формате Zstd.

Пример

Декодирование данных в формате Zstd

Исходный код

```
encoded_text =
decode_base64!("KLUv/QBY/QEAYsQOFKClbQBedqXsb96EWDax/f/F/z+gNU4ZTInaUeAj82KqPFjUzKqhcf
DqAIsLvAsnY1bI/N2mHzDixRQA")
decode_zstd!(encoded_text)
```

Результат

```
"you have successfully decoded me"
```

Функции кодирования

encode_base16

Кодирует `value` в Base16.

Спецификация функции

```
encode_base16(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да

Пример

Кодирование в Base16

Исходный код

```
encode_base16("please encode me")
```

Результат

```
"706c6561736520656e636f6465206d65"
```

encode_base64

Кодирует `value` в Base64.

Спецификация функции

```
encode_base64(value: <строка>, [padding: <логическое значение>, charset: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да
`padding`	логическое значение	Дополняется ли вывод Base64.	`true`	нет
`charset`	строка	Набор символов, используемый при кодировании данных: • `standard` — стандартный формат Base64; • `url_safe` — измененный Base64 для URL.	`standard`	нет

Примеры

Кодирование в Base64 (по умолчанию)

Исходный код

```
encode_base64("please encode me")
```

Результат

```
"cGxlYXNlIGVuY29kZSBtZQ=="
```

Кодирование в Base64 (без дополнения)

Исходный код

```
encode_base64("please encode me, no padding though", padding: false)
```

Результат

```
"cGxlYXNlIGVuY29kZSBtZSwgbm8gcGFkZGluZyB0aG91Z2g"
```

Кодирование в измененный Base64 для URL

Исходный код

```
encode_base64("please encode me, but safe for URLs", charset: "url_safe")
```

Результат

```
"cGxlYXNlIGVuY29kZSBtZSwgYnV0IHNhZmUgZm9yIFVSTHM="
```

encode_charset

Преобразует строку, закодированную в UTF-8, в строку заданной кодировки.

Спецификация функции

```
encode_charset(value: <строка>, to_charset: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка в UTF-8, которую нужно кодировать.		да
`to_charset`	строка	Набор символов, используемый для кодирования. Допустимые значения представлены в спецификации Encoding Standard.		да

Ошибки

Функция `encode_charset` **возвращает ошибку**, для которой требуется обработка, если `to_charset` не является корректным названием кодировки.

Примеры

Кодирование строки UTF-8 в EUC-KR

Исходный код

```
encode_base64(encode_charset!("□□□□□", "euc-kr"))
```

Результат

```
"vsiz58fPvLy/5A=="
```

Кодирование строки UTF8 в EUC-JP

Исходный код

```
encode_base64(encode_charset!("こんにちは", "euc-jp"))
```

Результат

```
"pLOk86TLpMGkzw=="
```

Кодирование строки UTF-8 в GB2312

Исходный код

```
encode_base64(encode_charset!("□好", "gb2312"))
```

Результат

```
"xOO6ww=="
```

encode_gzip

Кодирует `value` в формат Gzip.

Спецификация функции

```
encode_gzip(value: <строка>, [compression_level: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да
`compression_level`	целое значение	Уровень сжатия по умолчанию.	`6`	нет

Примеры

Кодирование в Gzip

Исходный код

```
encoded_text = encode_gzip("please encode me")
encode_base64(encoded_text)
```

Результат

```
"H4sIAAAAAAAA/yvISU0sTlVIzUvOT0lVyE0FAI4R4vcQAAAA"
```

encode_percent

Преобразует `value` в формат с использованием процентного кодирования для безопасного использования с URL.

Спецификация функции

```
encode_percent(value: <строка>, [ascii_set: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо преобразовать.		да
`ascii_set`	строка	Набор символов ASCII, используемый при преобразовании данных.	`NON_ALPHANUMERIC`	нет

Примеры

Процентное кодирование всех символов, не являющихся буквенно-цифровыми (по умолчанию)

Исходный код

```
encode_percent("foo bar?")
```

Результат

```
"foo%20bar%3F"
```

Процентное кодирование только управляющих символов

Исходный код

```
encode_percent("foo bar", ascii_set: "CONTROLS")
```

Результат

```
"foo %09bar"
```

encode_proto

Кодирует `value` в payload — полезную нагрузку Protocol Buffers.

Спецификация функции

```
encode_proto(value: <объект>, desc_file: <строка>, message_type: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект для преобразования в payload Protocol Buffers.		да
`desc_file`	строка	Путь к файлу набора дескрипторов Protocol Buffers. Должен быть строковым литерал. Этот файл является результатом работы команды `protoc -o`.		да
`message_type`	строка	Имя типа сообщения, используемого для сериализации. Должно быть строковым литералом.		да

Ошибки

Функция `encode_proto` **может возвращать ошибки**, для которых требуется обработка:

- Файл, указанный в аргументе `desc_file`, не существует.
- Тип сообщения, указанный в аргументе `message_type`, отсутствует в файле дескрипторов.

Пример

Исходный код

```
.payload = encode_base64(encode_proto!({"name": "someone", "phones": [{"number":
```

```
"123456"}]}, "resources/protobuf_descriptor_set.desc", "test_protobuf.Person"))
```

Результат

```
"Cgdzb21lb25lIggKBjEyMzQ1Ng=="
```

encode_punycode

Кодирует `значение` в формат punycode. Полезно для IDN.

Спецификация функции

```
encode_punycode(value: <строка>, [validate: <логическое значение>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для кодирования.		да
`validate`	логическое значение	Проверять ли, что входная строка является валидным именем домена.	`true`	нет

Ошибки

Функция `encode_punycode` **возвращает ошибку**, для которой требуется обработка, если `value` нельзя закодировать в punycode.

Примеры

Кодирование интернационализированного доменного имени

Исходный код

```
encode_punycode!("пример.испытание")
```

Результат

```
"xn--e1afmkfd.xn--80akhbyknj4f"
```

Кодирование интернационализированного доменного имени в смешанном регистре

Исходный код

```
encode_punycode!("пРИМер.испытание")
```

Результат

```
"xn--e1afmkfd.xn--80akhbyknj4f"
```

Кодирование строки из символов ASCII

Исходный код

```
encode_punycode!("example.com")
```

Результат

```
"example.com"
```

Кодирование без валидации

Исходный код

```
encode_punycode!("xn--8hbb.xn--fiba.xn--8hbf.xn--eib.", validate: false)
```

Результат

```
"xn--8hbb.xn--fiba.xn--8hbf.xn--eib."
```

Кодирование доменного имени в смешанном регистре без валидации

Исходный код

```
encode_punycode!("www.CAFé.example")
```

Результат

```
"www.xn--caf-dma.example"
```

encode_snappy

Кодирует `value` в формат Snappy.

Спецификация функции

```
encode_snappy(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для кодирования.		да

Ошибки

Функция `encode_snappy` **возвращает ошибку**, для которой требуется обработка, если `value` нельзя закодировать в формат Snappy.

Пример

Исходный код

```
encoded_text = encode_snappy!("A user account was created.")
encode_base64(encoded_text)
```

Результат

```
"G2hBIHVzZXIgYWNjb3VudCB3YXMgY3JlYXRlZC4="
```

encode_zlib

Кодирует `value` в формат Zlib.

Спецификация функции

```
encode_zlib(value: <строка>, [compression_level: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да
`compression_level`	целое число	Уровень сжатия по умолчанию.	`6`	нет

Пример

Кодирование в формат Zlib

Исходный код

```
encoded_text = encode_zlib("please encode me")
encode_base64(encoded_text)
```

Результат

```
"eJwryElNLE5VSM1Lzk9JVchNBQA0RQX7"
```

encode_zstd

Кодирует `value` в формат Zstandard.

Спецификация функции

```
encode_zstd(value: <строка>, [compression_level: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да
`compression_level`	целое число	Уровень сжатия по умолчанию.	`3`	нет

Пример

Кодирование в формат Zstandard

Исходный код

```
encoded_text = encode_zstd("please encode me")
encode_base64(encoded_text)
```

Результат

```
"KLUv/QBYgQAAcGxlYXNlIGVuY29kZSBtZQ=="
```

Функции сериализации

encode_json

Кодирует `value` в JSON.

Спецификация функции

```
encode_json(value: <любой тип>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо преобразовать в JSON-строку.		да

Пример

Кодирование в JSON

Исходный код

```
.payload = encode_json({"hello": "world"})
```

Результат

```
"{"hello":"world"}"
```

encode_key_value

Кодирует `value` в формат "ключ — значение" с настраиваемыми разделителями. По умолчанию используются разделители, соответствующие формату logfmt.

Спецификация функции

```
encode_key_value(value: <объект>, [fields_ordering: <массив>, key_value_delimiter:
<строка>, field_delimiter: <строка>, flatten_boolean: <логическое значение>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Значение, которое необходимо преобразовать в строку.		да
`fields_ordering`	массив	Упорядочивание полей для сохранения порядка. Любые поля, отсутствующие в этом списке, будут отображаться неупорядоченными, после всех упорядоченных полей.		нет
`key_value_delimiter`	строка	Строка, разделяющая ключ и значение.	`=`	нет
`field_delimiter`	строка	Строка, разделяющая каждую пару "ключ — значение".	(пробел)	нет
`flatten_boolean`	логическое значение	Указывает, следует ли кодировать "ключ — значение" с логическим значением как отдельный ключ.	`false`	нет

Ошибки

Функция `encode_key_value` **возвращает ошибку**, для которой требуется обработка, если в `fields_ordering` содержится элемент, не являющийся строкой.

Если `fields_ordering` не указан, функция не возвращает ошибок.

Примеры

Кодирование с разделителями по умолчанию, без упорядочивания полей

Исходный код

```
encode_key_value({"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl":
"info"})
```

Результат

```
"lvl=info msg=\"This is a message\" ts=2021-06-05T17:20:00Z"
```

Кодирование с разделителями по умолчанию и упорядочиванием полей

Исходный код

```
encode_key_value!({"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl":
"info", "log_id": 12345}, ["ts", "lvl", "msg"])
```

Результат

```
"ts=2021-06-05T17:20:00Z lvl=info msg=\"This is a message\" log_id=12345"
```

Кодирование с разделителями по умолчанию (вложенные поля)

Исходный код

```
encode_key_value({"agent": {"name": "vector"}, "log": {"file": {"path": "my.log"}},
"event": "log"})
```

Результат

```
"agent.name=vector event=log log.file.path=my.log"
```

Кодирование с разделителями по умолчанию (вложенные поля с упорядочиванием)

Исходный код

```
encode_key_value!({"agent": {"name": "vector"}, "log": {"file": {"path": "my.log"}},
"event": "log"}, ["event", "log.file.path", "agent.name"])
```

Результат

```
"event=log log.file.path=my.log agent.name=vector"
```

Кодирование с пользовательскими разделителями, без упорядочивания полей

Исходный код

```
encode_key_value(
{"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl": "info"},
field_delimiter: ",",
key_value_delimiter: ":"
)
```

Результат

```
"lvl:info,msg:\"This is a message\",ts:2021-06-05T17:20:00Z"
```

Кодирование с пользовательскими разделителями и сглаживанием логических значений

Исходный код

```
encode_key_value(
{"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl": "info", "beta":
true, "dropped": false},
field_delimiter: ",",
key_value_delimiter: ":",
flatten_boolean: true
)
```

Результат

```
"beta,lvl:info,msg:\"This is a message\",ts:2021-06-05T17:20:00Z"
```

encode_logfmt

Кодирует `value` в формат logfmt.

Спецификация функции

```
encode_logfmt(value: <объект>, [fields_ordering: <массив>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Значение, которое необходимо преобразовать в строку в формате logfmt.		да
`fields_ordering`	массив	Упорядочивание полей для сохранения порядка. Любые поля, отсутствующие в этом списке, будут отображаться неупорядоченными, после всех упорядоченных полей.		нет

Ошибки

Функция `encode_logfmt` **возвращает ошибку**, для которой требуется обработка, если в `fields_ordering` содержится элемент, не являющийся строкой.

Если `fields_ordering` не указан, функция не возвращает ошибок.

Примеры

Кодирование в формат logfmt без упорядочивания полей

Исходный код

```
encode_logfmt({"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl":
"info"})
```

Результат

```
"lvl=info msg=\"This is a message\" ts=2021-06-05T17:20:00Z"
```

Кодирование в формат logfmt с упорядочиванием полей

Исходный код

```
encode_logfmt!({"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl":
"info", "log_id": 12345}, ["ts", "lvl", "msg"])
```

Результат

```
"ts=2021-06-05T17:20:00Z lvl=info msg=\"This is a message\" log_id=12345"
```

Кодирование в формат logfmt (вложенные поля)

Исходный код

```
encode_logfmt({"agent": {"name": "vector"}, "log": {"file": {"path": "my.log"}},
"event": "log"})
```

Результат

```
"agent.name=vector event=log log.file.path=my.log"
```

Кодирование в формат logfmt (вложенные поля с упорядочиванием)

Исходный код

```
encode_logfmt!({"agent": {"name": "vector"}, "log": {"file": {"path": "my.log"}},
"event": "log"}, ["event", "log.file.path", "agent.name"])
```

Результат

```
"event=log log.file.path=my.log agent.name=vector"
```

Функции событий и активных списков



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции секретов

get_secret

Возвращает значение заданного секрета из события.

Спецификация функции

```
get_secret(key: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
key	строка	Имя секрета.		да

Пример

Получение ключа API Datadog из метаданных события

Исходный код

```
get_secret("datadog_api_key")
```

Результат

```
"secret value"
```

remove_secret

Удаляет секрет из события.

Спецификация функции

```
remove_secret(key: <строка>)
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
key	строка	Имя секрета для удаления.		да

Пример

Удаление ключа API Datadog из события

Исходный код

```
remove_secret("datadog_api_key")
```

Результат

```
null
```

set_secret

Устанавливает указанный секрет в событие.

Спецификация функции

```
set_secret(key: <строка>, secret: <строка>)
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
key	строка	Имя секрета		да
secret	строка	Значение секрета		да

Пример

Установка значения ключа API Datadog

Исходный код

```
set_secret("datadog_api_key", "abc122")
```

Результат

```
null
```

set_semantic_meaning

Устанавливает семантический смысл для события.



Эта функция устанавливает значение при запуске и **не выполняется** поведения во время работы программы. Рекомендуется размещать все вызовы этой функции в начале программы VRL.

Функцию нельзя вызывать внутри условия. Например, нельзя предотвратить присваивание значения с использованием оператора `if`.

Спецификация функции

```
set_semantic_meaning(target: <путь>, meaning: <строка>)
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`target`	путь	Путь к значению, которому будет назначено семантическое значение.		да
`meaning`	строка	Имя назначаемого значения.		да

Пример

Присваивание полю пользовательского семантического смысла

Исходный код

```
set_semantic_meaning(.foo, "bar")
```

Результат

```
null
```

Функции таблиц обогащения

find_enrichment_table_records

Поиск строк в таблице обогащения, соответствующих предоставленному условию.



Поскольку условие используется при загрузке таблицы для создания индексов в данных, оно должно быть представлено литералом, без использования переменных.

- Для таблиц обогащения типа `file` это условие должно быть объектом VRL, в котором ключи и значения указывают на поле для поиска и значение для поиска в этом поле. Эта функция возвращает строки, соответствующие предоставленным условиям. Все поля должны совпадать, чтобы строки были возвращены; если хотя бы одно поле не совпадает, строки не возвращаются.

В настоящее время существуют следующие формы критериев поиска:

- **Поиск точного совпадения**. Заданное поле должно точно совпадать со значением. Чувствительность к регистру может быть указана с помощью аргумента `case_sensitive`. Точный поиск может использовать индекс прямо в наборе данных, что делает его менее ресурсозатратным с точки зрения производительности.
- **Поиск по диапазону дат**. Заданное поле должно попадать в диапазон от даты `from` до даты `to` включительно.



Поиск по диапазону дат включает последовательное сканирование строк, которые были найдены с помощью любых точных критериев. Такая операция может оказаться ресурсозатратной, если много строк возвращается любыми точными критериями. Рекомендуется использовать диапазоны дат как **единственные** критерии, когда набор данных обогащения очень мал.

- Для таблиц обогащения `geoip` это условие должно быть объектом VRL с одной парой "ключ — значение", значение которой должно быть действительным IP-адресом. Пример: `{"ip": .ip }`. Если ожидается поле с результатом без значения, будет использовано значение `null`.

▼ *Поля, возвращаемые из таблицы*

- Базы данных ISP:
  - `autonomous_system_number`;
  - `autonomous_system_organization`;
  - `isp`;
  - `organization`.
- Базы данных City:
  - `city_name`;
  - `continent_code`;
  - `country_code`;
  - `country_name`;
  - `region_code`;
  - `region_name`;
  - `metro_code`;
  - `latitude`;
  - `longitude`;
  - `postal_code`;
  - `timezone`.
- Базы данных Connection-Type:
  - `connection_type`.

Спецификация функции

```
find_enrichment_table_records(table: <строка>,
condition: <объект>,
[select: <массив>,]
[case_sensitive: <логическое значение>])
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`table`	строка	Таблица обогащения для поиска.		да
`condition`	объект	Условие для поиска.		да
`select`	массив	Подмножество возвращаемых полей из таблицы обогащения. Если не указано, будут возвращены все поля.		нет
`case_sensitive`	логическое значение	Определяет, нужно ли точное совпадение текстовых полей.	`true`	нет

Примеры

Точное совпадение

Исходный код

```
find_enrichment_table_records!("test",
{"surname": "smith",},
case_sensitive: false)
```

Результат

```
[
  {
    "id": 1,
    "firstname": "Bob",
    "surname": "Smith"
  },
  {
    "id": 2,
    "firstname": "Fred",
    "surname": "Smith"
  }
]
```

Поиск по диапазону дат

Исходный код

```
find_enrichment_table_records!("test",
{
```

```
  "surname": "Smith",
  "date_of_birth": {
  "from": t'1985-01-01T00:00:00Z',
  "to": t'1985-12-31T00:00:00Z'
  }
})
```

Результат

```
[
  {
    "id": 1,
    "firstname": "Bob",
    "surname": "Smith"
  },
  {
    "id": 2,
    "firstname": "Fred",
    "surname": "Smith"
  }
]
```

get_enrichment_table_record

Функция ищет строку в таблице обогащения, которая соответствует указанному условию. Должна быть найдена одна строка. Если строка не найдена или найдено более одной строки, возвращается ошибка.



Поскольку условие используется при загрузке таблицы для создания индексов в данных, оно должно быть представлено литералом, без использования переменных.

- Для таблиц обогащения типа `file` это условие должно быть объектом VRL, в котором ключи и значения указывают на поле для поиска и значение для поиска в этом поле. Эта функция возвращает строки, соответствующие предоставленным условиям. Все поля должны совпадать, чтобы строки были возвращены; если хотя бы одно поле не совпадает, строки не возвращаются.

  В настоящее время существуют следующие формы критериев поиска:

  - **Поиск точного совпадения**. Заданное поле должно точно совпадать со значением. Чувствительность к регистру может быть указана с помощью аргумента `case_sensitive`. Точный поиск может использовать индекс прямо в наборе данных, что делает его менее ресурсозатратным с точки зрения производительности.
  - **Поиск по диапазону дат**. Заданное поле должно попадать в диапазон от даты `from` до даты `to` включительно.



Поиск по диапазону дат включает последовательное сканирование строк, которые были найдены с помощью любых точных критериев. Такая операция может оказаться ресурсозатратной, если много строк

возвращается любыми точными критериями. Рекомендуется использовать диапазоны дат как **единственные** критерии, когда набор данных обогащения очень мал.

- Для таблиц обогащения `geoip` это условие должно быть объектом VRL с одной парой "ключ — значение", значение которой должно быть действительным IP-адресом. Пример: `{"ip": .ip }`. Если ожидается поле с результатом без значения, будет использовано значение `null`.

  ▼ *Поля, возвращаемые из таблицы*

  - Базы данных ISP:
    - `autonomous_system_number`;
    - `autonomous_system_organization`;
    - `isp`;
    - `organization`.
  - Базы данных City:
    - `city_name`;
    - `continent_code`;
    - `country_code`;
    - `country_name`;
    - `region_code`;
    - `region_name`;
    - `metro_code`;
    - `latitude`;
    - `longitude`;
    - `postal_code`;
    - `timezone`.
  - Базы данных Connection-Type:
    - `connection_type`.

Спецификация функции

```
get_enrichment_table_record(table: <строка>,
condition: <объект>,
[select: <массив>,]
[case_sensitive: <логическое значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`table`	строка	Таблица обогащения для поиска.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`condition`	объект	Условие для поиска.		да
`select`	массив	Подмножество возвращаемых полей из таблицы обогащения. Если не указано, будут возвращены все поля.		нет
`case_sensitive`	логическое значение	Должны ли текстовые поля соответствовать регистру точно.	`true`	нет

Ошибки

Функция `get_enrichment_table_record` **возвращает ошибки**, для которых требуется обработка:

- Строка не найдена;
- Найдено несколько строк, соответствующих условию.



Эта ошибка не возвращается для искусственных таблиц обогащения в поле `setup_tests` элементов экспертизы.

Примеры

Точное совпадение

Исходный код

```
get_enrichment_table_record!("test",
{
  "surname": "bob",
  "firstname": "Ivan"
},
case_sensitive: false)
```

Результат

```
{
  "id": 1,
  "firstname": "Bob",
  "surname": "Smith"
}
```

Поиск по диапазону даты

Исходный код

```
get_enrichment_table_record!("test",
{
  "surname": "Smith",
  "date_of_birth": {
```

```
    "from": t'1985-01-01T00:00:00Z',
    "to": t'1985-12-31T00:00:00Z'
  }
})
```

Результат

```
{
  "id": 1,
  "firstname": "Bob",
  "surname": "Smith"
}
```

Функции активных списков

active_record_excluded

Проверяет, есть ли запись с указанным ключом `key` в активном списке `active_list`. Значения полей в ключе могут быть указаны с выражением объединения, что приводит к генерации разных вариантов ключа. При проверке функция ищет не только совпадения с переданными значениями полей, но и со значением `*`.

Функция возвращает `true`, если хотя бы один из вариантов ключа совпадает с записью из списка исключений.

Спецификация функции

```
active_record_excluded(active_list: <строка>, key: <объект>)
:: <логическое значение>, <ошибка>
```

Аргумент	Тип	Описание	Обязателен
`active_list`	строка	Название схемы активного списка, в котором выполняется проверка. Название должно быть представлено литералом строки, без использования переменных.	да
`key`	объект	Ключ записи.	да

Ошибки

Функция `active_record_excluded` **может возвращать ошибки**, для которых требуется обработка:

- Параметр `active_list` не закодирован в UTF-8.
- Параметр `key` не является объектом.

Пример

Фрагмент схемы активного списка

```
...
schema:
  exclusion_id: string
  user: string
  host: string
  notes: string
  ...
key:
  - rule_id
  - user
  - host
content:
  - exclusion_id: "admin_actions"
    user: "admin"
    host: "*"
    ...
```

Фрагмент события

```
{
  "user": "admin",
  "hostname": "localhost"
}
```

Исходный код

```
active_record_excluded!("user_whitelist", { "user": .user ?? "", "host":
downcase(.hostname) ?? ""})
```

Результат

```
true
```

add_active_record

Добавляет запись в активный список. Если запись с таким ключом уже существует, значение обновляется. В противном случае создается новая запись.



Следует заполнять все поля записи, предусмотренные схемой активного списка, в противном случае запись добавлена не будет.

Спецификация функции

```
add_active_record(list: <строка>, value: <объект>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`list`	строка	название схемы активного списка, в который добавляется запись		да
`value`	объект	значение записи		да

Примеры

Добавление записи в активный список

Исходный код

```
add_active_record!("suspicious_ips", { "source_ip": .source_ip, "is_suspicious": true
})
```

Результат

```
true
```

Обновление записи в активном списке

Исходный код

```
failure_count = get_active_record!("auth_failure_counts", { "user_id": .user_id
}).failure_count || 0
add_active_record!("auth_failure_counts", { "user_id": .user_id, {"failure_count":
failure_count + 1 })
```

Результат

```
true
```

get_active_record

Извлекает значение из активного списка по указанному ключу. Возвращает значение, ассоциированное с ключом, или `null`, если запись отсутствует.

Спецификация функции

```
get_active_record(list: <строка>, key: <объект>)
:: <любое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`list`	строка	активный список, из которого нужно извлечь запись		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`key`	объект	уникальный ключ записи		да

Примеры

В примерах используется событие со следующими полями:

Событие

```
{
  "sourceIp": 198.51.100.12,
  "ip": 198.51.100.23,
}
```

Получение записи из активного списка (запись найдена)

Исходный код

```
get_active_record!("allowed_ips", { "ip": .sourceIp })
# Проверяется совпадение с полем sourceIp.
```

Результат

```
{
  "ip": 198.51.100.12,
  "is_suspicious": true
}
```

Получение записи из активного списка (запись не найдена)

Исходный код

```
get_active_record!("allowed_ips", { "ip": .ip })
# Проверяется совпадение с полем ip.
```

Результат

```
null
```

remove_active_record

Удаляет запись из активного списка. Если запись с таким ключом отсутствует, ничего не происходит.

Спецификация функции

```
remove_active_record(list: <строка>, key: <объект>)
```

```
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`list`	строка	активный список, из которого необходимо удалить запись		да
`key`	объект	уникальный ключ записи, которую необходимо удалить		да

Пример

Удаление записи из активного списка

Исходный код

```
remove_active_record!("auth_failure_counts", { "user_id": .user_id })
```

Результат

```
true
```

Функции коллекций



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции для работы с коллекциями предоставляют набор операций для манипуляции массивами и объектами.

Функции извлечения данных

includes

Определяет, включает ли массив `value` указанный элемент `element`.

Спецификация функции

```
includes(value: <массив>, element: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Массив.		да
`element`	любой	Элемент для проверки.		да

Пример

Массив включает указанный элемент

Исходный код

```
includes(["apple", "orange", "banana"], "banana")
```

Результат

```
true
```

keys

Возвращает ключи переданного в функцию объекта.

Спецификация функции

```
keys(value: <объект>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект, из которого нужно извлечь ключи.		да

Пример

Получение ключей из объекта

Исходный код

```
keys({"key1": "val1", "key2": "val2"})
```

Результат

```
[
  "key1",
  "key2"
]
```

length

Возвращает длину `value`. Результат зависит от типа `value`:

- для массива возвращает количество элементов;
- для объекта возвращает количество ключей на верхнем уровне;
- для строки возвращает количество байтов в строке. Если нужно получить количество символов, используйте функцию `strlen`.

Спецификация функции

```
length(value: <массив | объект | строка>)
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект строка	Массив, объект или строка.		да

Примеры

Длина объекта

Исходный код

```
length({
  "portland": "Trail Blazers",
  "seattle": "Supersonics"
})
```

Результат

```
2
```

Длина объекта со вложенными объектами

Исходный код

```
length({
  "home": {
    "city": "Portland",
    "state": "Oregon"
  },
  "name": "Trail Blazers",
  "mascot": {
```

```
    "name": "Blaze the Trail Cat"
  }
})
```

Результат

```
3
```

Длина массива

Исходный код

```
length(["Trail Blazers", "Supersonics", "Grizzlies"])
```

Результат

```
3
```

Длина строки

Исходный код

```
length("The Planet of the Apes Musical")
```

Результат

```
30
```

values

Возвращает значения из переданного объекта.



Чтобы получить срез массива, используйте функцию `slice`.

Спецификация функции

```
values(value: <объект>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект, из которого извлекаются значения.		да

Пример

Получение значений из объекта

Исходный код

```
values({"key1": "val1", "key2": "val2"})
```

Результат

```
[
  "val1",
  "val2"
]
```

Функции фильтрации

compact

Сжимает `value`, то есть удаляет пустые элементы. В аргументах функции можно указать, какие значения следует считать пустыми.

Спецификация функции

```
compact(value: <массив | объект>, [recursive: <логическое значение>, null: <логическое
значение>, string: <логическое значение>, object: <логическое значение>, array:
<логическое значение>, nullish: <логическое значение>])
:: <массив | объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Объект или массив для сжатия.		да
`recursive`	логическое значение	Будет ли сжатие рекурсивным, то есть удаляются ли пустые элементы вложенных объектов и массивов.	`true`	нет
`null`	логическое значение	Рассматривается ли `null` как пустое значение.	`true`	нет
`string`	логическое значение	Рассматривается ли пустая строка как пустое значение.	`true`	нет
`object`	логическое значение	Рассматривается ли объект без элементов как пустое значение.	`true`	нет
`array`	логическое значение	Рассматривается ли массив без элементов как пустое значение.	`true`	нет

Аргумент	Тип	Описание	По умолчанию	Обязателен
`nullish`	логическое значение	Рассматриваются ли "нулевые" значения, которые определены функцией `is_nullish`, как пустые.	`false`	нет

Примеры

Сжатие массива

Исходный код

```
compact(["foo", "bar", "", null, [], "buzz"], string: true, array: true, null: true)
```

Результат

```
[
  "foo",
  "bar",
  "buzz"
]
```

Сжатие объекта

Исходный код

```
compact({"field1": 1, "field2": "", "field3": [], "field4": null}, string: true,
array: true, null: true)
```

Результат

```
{
  "field1": 1
}
```

filter

Отбирает элементы из коллекции.

Эта функция не поддерживает рекурсивную итерацию.

Функция использует синтаксис замыкания для доступа к ключу и значению или индексу и значению каждого элемента в коллекции.



К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока

замыкания, недоступны за его пределами.

Спецификация функции

```
filter(value: <массив | объект>) -> |<переменная ключа или индекса>, <переменная
значения>| {
    <блок>
}
:: <массив | объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Массив или объект для фильтрации.		да

Пример

Фильтрация элементов

Исходный код

```
filter(array!(.tags)) -> |_index, value| {
  # Оставить элементы, которые не равны "foo".
  value != "foo"
}
```

Результат

```
[
  "bar",
  "baz"
]
```

rv_coalesce_nullish

Возвращает первый элемент массива, значение которого не является "nullish", то есть первый элемент со значимым содержанием. Значениями "nullish" считаются следующие:

- `null`;
- `"-"`;
- состоящие только из пробелов, то есть символов со свойством Unicode `White_Space`.

Если все значения в массиве являются "nullish", функция возвращает `null`.

Спецификация функции

```
rv_coalesce_nullish(value: <массив>)
:: <любой тип>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	массив, в котором производится поиск элементов, не являющихся "nullish"		да

Ошибки

Функция `rv_coalesce_nullish` **возвращает ошибку**, для которой требуется обработка, если `value` не является массивом.

Примеры

Есть и значения "nullish", и значимые

Исходный код

```
rv_coalesce_nullish(["  ", "-", "test"])
```

Результат

```
"test"
```

Все элементы имеют значимое содержание

Исходный код

```
rv_coalesce_nullish(["123132", "test"])
```

Результат

```
"123132"
```

Все элементы являются "nullish"

Исходный код

```
rv_coalesce_nullish(["-", "   ", "", null, ""])
```

Результат

```
null
```

unique

Возвращает уникальные значения для массива.

Сохраняется только первое вхождение каждого элемента.

Спецификация функции

```
unique(value: <массив>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Массив, из которого будут возвращены уникальные элементы.		да

Пример

Уникальные значения

Исходный код

```
unique(["foo", "bar", "foo", "baz"])
```

Результат

```
[
  "foo",
  "bar",
  "baz"
]
```

Функции итерации по элементам коллекции



Чтобы избежать лишних вычислений, старайтесь выносить за пределы цикла все вычисления, не зависящие от него.



Функция `join`, объединяющая элементы массива в строку, описана в разделе join.

all

Проверяет, что все элементы `value` соответствуют условию, указанному в синтаксисе замыкания следующего вида:

```
|<key>, <value>| { <condition> }
```

Здесь:

- `<key>` — ключ (для объектов) или индекс (для массивов) аргумента.
- `<value>` — значение аргумента.
- `<condition>` — условие. В условии можно изменять внешние переменные.

Выполнение прекращается при первом значении `false` для любого элемента, дальнейшая проверка не производится.

Спецификация функции

```
all(value: <массив или объект>) -> |<переменная ключа или индекс>, <переменная
значения>| { <блок> }
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Массив или объект для проверки		да

Примеры

Итерация объекта

Исходный код

```
all({ "a": 1, "b": 2 }) -> |_key, value| { value > 0 }
```

Результат

```
true
```

Итерация массива

Исходный код

```
all([1, 2, 3, 4, 5]) -> |index, value| { value > 0 }
```

Результат

```
true
```

any

Проверяет, что хотя бы один элемент `value` соответствует условию, указанному в синтаксисе замыкания следующего вида:

```
|<key>, <value>| { <condition> }
```

Здесь:

- `<key>` — ключ (для объектов) или индекс (для массивов) аргумента.
- `<value>` — значение аргумента.
- `<condition>` — условие. В условии можно изменять внешние переменные.

Выполнение прекращается при первом значении `true` для любого элемента, дальнейшая проверка не производится.

Спецификация функции

```
any(value: <массив или объект>) -> |<переменная ключа или индекс>, <переменная
значения>| { <блок> }
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Массив или объект для проверки		да

Примеры

Итерация объекта

Исходный код

```
any({ "a": 1, "b": 2 }) -> |_key, value| { value == 2 }
```

Результат

```
true
```

Итерация массива

Исходный код

```
all([1, 2, 3, 4, 5]) -> |index, value| { value == 2 }
```

Результат

```
true
```

for_each

Итерирует по коллекции.

В настоящее время функция **не поддерживает** рекурсивную итерацию.

Функция использует синтаксис замыкания для доступа к идентификатору и значению каждого элемента в коллекции.



К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока замыкания, недоступны за его пределами.

Спецификация функции

```
for_each(value: <массив | объект>) -> |<переменная ключа или индекса>, <переменная
значения>| {
    <блок>
}
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
value	массив объект	Массив или объект для итерации.		да

Примеры

Подсчет элементов

Исходный код

```
tally = {}
for_each(array!(.tags)) -> |_index, value| {
  # Получение текущего подсчета для `value` или установка его в `0`.
  count = int(get!(tally, [value])) ?? 0
  # Увеличение подсчета для значения на `1`.
  tally = set!(tally, [value], count + 1)
}
tally
```

Результат

```
{
  "foo": 2,
  "bar": 1,
```

```
  "baz": 1
}
```

map_keys

Отображает ключи объекта.

Если включен параметр `recursive`, функция обрабатывает вложенные объекты согласно следующим правилам:

1. Итерация начинается с корневого объекта.
2. Для каждого вложенного объекта:
    - Сначала возвращается ключ самого объекта.
    - Затем происходит рекурсивный вызов функции для объекта, и происходит возврат к пункту 1.
    - Любые изменения, сделанные во вложенном объекте **до** его рекурсивного вызова, сохраняются.
3. Для каждого вложенного массива:
    - Сначала возвращается ключ самого массива.
    - Затем происходит поиск всех объектов внутри массива и применение к каждому из них правил из пункта 2.

На практике эти правила означают, что `map_keys` с включенным параметром `recursive` найдет **все** ключи в целевом объекте, независимо от того, находятся ли вложенные объекты внутри массивов.

Функция использует синтаксис замыкания для доступа к ключу каждого элемента в объекте.



К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока замыкания, недоступны за его пределами.

Спецификация функции

```
map_keys(value: <объект>, [recursive: <логическое значение>]) -> |<переменная ключа>|
{ <блок> }
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект для итерации.		да
`recursive`	логическое значение	Определяет, следует ли выполнять рекурсивную итерацию по коллекции.	`false`	нет

Примеры

Приведение ключей к верхнему регистру

Исходный код

```
map_keys(.) -> |key| { upcase(key) }
```

Результат

```
{
  "FOO": "foo",
  "BAR": "bar"
}
```

Замена точек в ключах на подчеркивания

Исходный код

```
map_keys(., recursive: true) -> |key| { replace(key, ".", "_") }
```

Результат

```
{
  "labels": {
    "app_kubernetes_io/name": "mysql"
  }
}
```

map_values

Отображает значения внутри коллекции.

Если включен параметр `recursive`, функция обрабатывает вложенные коллекции согласно следующим правилам:

1. Итерация начинается с корневой коллекции.
2. Для каждой вложенной коллекции:
   - Сначала возвращается сам тип коллекции.
   - Затем происходит рекурсивный вызов функции для коллекции, и происходит возврат к пункту 1 в этом списке.
   - Любые изменения, сделанные в коллекции **до** ее рекурсивного вызова, сохраняются.

Функция использует синтаксис замыкания, чтобы изменить значение каждого элемента в объекте.



К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока

замыкания, недоступны за его пределами.

Спецификация функции

```
map_values(value: <массив | объект>, [recursive: <логическое значение>]) ->
|<переменная значения>| { <блок> }
:: <массив | объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Коллекция (массив или объект) для итерации.		да
`recursive`	логическое значение	Определяет, следует ли выполнять рекурсивную итерацию по коллекции.	`false`	нет

Пример

Приведение значений к верхнему регистру

Исходный код

```
map_values(.) -> |value| { upcase!(value) }
```

Результат

```
{
  "foo": "FOO",
  "bar": "BAR"
}
```

match_datadog_query

Сопоставляет объект `value` с запросом Datadog Search Syntax.

Спецификация функции

```
match_datadog_query(value: <объект>, query: <строка> )
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Коллекция (объект)		да
`query`	строка	Запрос Datadog Search Syntax		да

Примеры

Запрос OR

Исходный код

```
match_datadog_query({"message": "contains this and that"}, "this OR that")
```

Результат

```
true
```

Запрос AND

Исходный код

```
match_datadog_query({"message": "contains only this"}, "this AND that")
```

Результат

```
false
```

Символ подстановки в атрибутах

Исходный код

```
match_datadog_query({"name": "foobar"}, "@name:foo*")
```

Результат

```
true
```

Диапазон тегов

Исходный код

```
match_datadog_query({"tags": ["a:x", "b:y", "c:z"]}, s'b:["x" TO "z"]')
```

Результат

```
true
```

Функции изменения структуры коллекции

chunks

Разбивает `value` на части длиной `chunk_size` в байтах.

Спецификация функции

```
chunks(value: <массив | строка>, chunk_size: <целое число>)
:: <массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив строка	Массив байтов или строка для разделения.		да
`chunk_size`	целое число	Желаемая длина каждого фрагмента в байтах, не менее 1 байта. Длина может быть ограничена архитектурой хост-платформы.		да

Характеристики функции

При делении на фрагменты границы кодовых точек Unicode не соблюдаются.

Ошибки

Функция `chunks` **может возвращать ошибки**, для которых требуется обработка:

- `chunk_size` менее 1 байта.
- `chunk_size` слишком велик.

Примеры

Разделение строки на фрагменты

Исходный код

```
chunks("abcdefgh", 4)
```

Результат

```
[
  "abcd",
  "efgh"
]
```

Деление без учета границ кодовой точки Unicode

Исходный код

```
chunks("ab好", 4)
```

Результат

```
[
  "ab",
  "好"
]
```

flatten

Выполняет преобразование `value` в представление с одним уровнем.

Спецификация функции

```
flatten(value: <массив | объект>, [separator: <строка>])
:: <массив | объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Массив или объект, которые необходимо преобразовать.		да
`separator`	строка	Разделитель, используемый для объединения вложенных ключей.	`.`	нет

Примеры

Преобразование массива

Исходный код

```
flatten([1, [2, 3, 4], [5, [6, 7], 8], 9])
```

Результат

```
[
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9
]
```

Преобразование объекта

Исходный код

```
flatten({
  "parent1": {
    "child1": 1,
    "child2": 2
  },
  "parent2": {
    "child3": 3
  }
})
```

Результат

```
{
  "parent1.child1": 1,
  "parent1.child2": 2,
  "parent2.child3": 3
}
```

unflatten

Преобразует `value` во вложенное представление. Функция может быть полезна для возвращения плоского объекта, полученного с помощью функции flatten, в исходную форму.

Спецификация функции

```
unflatten(value: <объект>, [separator: <строка>, recursive: <логическое значение>])
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Массив или объект, которые необходимо преобразовать.		да
`separator`	строка	Разделитель, используемый для разделения ключей.	`.`	нет
`recursive`	логическое значение	Требуется ли рекурсивно преобразовывать значения объекта.	`true`	нет

Характеристики

Игнорирование неконсистентных значений ключей

Если у нескольких значений один и тот же ключ является префиксом, значения ключей верхнего уровня вложенности отбрасываются и сохраняется только вложенный объект.

В примере ниже для ключа `a` есть два возможных значения — число 3 и объект с ключами `b` и `c`. При

применении функции `unflatten` числовое значение отбрасывается, сохраняется только объект.

Исходный код

```
unflatten({
  "a": 3,
  "a.b": 2,
  "a.c": 4
})
```

Результат

```
{
  "a": {
    "b": 2,
    "c": 4
  }
}
```

Примеры

Преобразование во вложенное представление

Исходный код

```
unflatten({
  "foo.bar.baz": true,
  "foo.bar.qux": false,
  "foo.quux": 42
})
```

Результат

```
{
  "foo": {
    "bar": {
      "baz": true,
      "qux": false
    },
    "quux": 42
  }
}
```

Рекурсивное преобразование во вложенное представление

Исходный код

```
unflatten({
  "flattened.parent": {
```

```
    "foo.bar": true,
    "foo.baz": false
  }
})
```

Результат

```
{
  "flattened": {
    "parent": {
      "foo": {
        "bar": true,
        "baz": false
      }
    }
  }
}
```

Нерекурсивное преобразование во вложенное представление

Исходный код

```
unflatten({
  "flattened.parent": {
    "foo.bar": true,
    "foo.baz": false
  }
},
recursive: false)
```

Результат

```
{
  "flattened": {
    "parent": {
      "foo.bar": true,
      "foo.baz": false
    }
  }
}
```

merge

Обновляет поля в объекте `to` на основе содержимого объекта `from`. При этом:

- Поля, которые есть только в одном из объектов, возвращаются в результирующем объекте без изменений.
- Если поле существует в обоих объектах, значение из объекта `from` заменяет значение из объекта `to`. Чтобы объединить при слиянии вложенные поля, используйте параметр `deep=true`.

Спецификация функции

```
merge(to: <объект>, from: <объект>, [deep: <логическое значение>])
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`to`	объект	Объект, в который выполняется слияние.		да
`from`	объект	Объект, из которого выполняется слияние.		да
`deep`	логическое значение	Выполняется глубокое слияние, если `true`, в противном случае слияние происходит только на верхнем уровне.	`false`	нет

Примеры

Поверхностное слияние объектов

Исходный код

```
merge(
  {
    "parent1": {
      "child1": 1,
      "child2": 2
    },
    "parent2": {
      "child3": 3
    }
  },
  {
  "parent1": {
    "child2": 4,
    "child5": 5
  }
})
```

Результат

```
{
  "parent1": {
    "child2": 4,
    "child5": 5
  },
  "parent2": {
    "child3": 3
  }
```

```
}
```

Глубокое слияние объектов

Исходный код

```
merge(
  {
    "parent1": {
      "child1": 1,
      "child2": 2
  },
   "parent2": {
     "child3": 3
   }
  },
  {
    "parent1": {
      "child2": 4,
      "child5": 5
    }
  },
deep: true
)
```

Результат

```
{
  "parent1": {
    "child1": 1,
    "child2": 4,
    "child5": 5
  },
   "parent2": {
    "child3": 3
  }
}
```

object_from_array

Создает объект из пар "ключ — значение", итерируя либо единый массив массивов, либо два отдельных массива. Ключи, равные `null`, игнорируются.

Если указан только один аргумент, он должен представлять собой массив, содержащий массивы с ключом и значением. Вместо пропущенных значений записывается `null`.

Если используются два массива, длина результирующего массива равна длине самого короткого входного массива, а все остальные элементы отбрасываются.

Спецификация функции

```
object_from_array(values: <массив>, [keys: <массив>])
:: <объект>, <error>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`values`	массив	Массив значений, или массив входных массивов, если других аргументов не передано.		да
`keys`	массив	Массив ключей. Если аргумент не указан, массивы и ключей, и значений должны быть переданы в первом аргументе.		нет

Ошибки

Функция `object_from_array` **может возвращать ошибки**, для которых требуется обработка:

- `values` или `keys` не являются массивами;
- `values` является единственным аргументом и содержит элементы, не являющиеся массивами.

Примеры

Создание объекта из массива массивов

Исходный код

```
object_from_array([["a", 1], [null, 2], ["b"], ["c", true, 3, 4]])
```

Результат

```
{
  "a": 1,
  "b": null,
  "c": true
}
```

Создание объекта из отдельных массивов ключей и значений

Исходный код

```
object_from_array([1, 2, 3, 4, 5], keys: ["one", null, "two"])
```

Результат

```
{
  "one": 1,
  "two": 3
```

```
}
```

rv_rename_keys

Переименовывает ключи в объекте согласно таблице обогащения.

Спецификация функции

```
rename_keys(value: <объект>, table: <строка>, from: <строка>, to <строка>, [separator:
<строка>, join_separator:  <строка>, case_sensitive: <логическое>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект, в котором нужно переименовать поля.		да
`table`	строка	Таблица обогащения.		да
`from`	строка	Столбец с текущими именами. Строка должна быть представлена литералом, без использования переменных.		да
`to`	строка	Столбец с новыми именами. Строка должна быть представлена литералом, без использования переменных.		да
`separator`	строка	Строка, разделяющая части сложных ключей. Строка должна быть представлена литералом, без использования переменных.		нет
`join_separator`	строка	Строка, соединяющая части сложных ключей после переименования. Если параметр не указан, то используется значение `separator`. Строка должна быть представлена литералом, без использования переменных.		нет

Аргумент	Тип	Описание	По умолчанию	Обязателен
`case_sensitive`	логическое значение	Указывает, будет ли поиск ключей в таблице обогащения регистрозависимым. Значение должно быть представлено литералом `true` или `false`, без использования переменных.	`false`	нет

Ошибки

Функция `rv_rename_keys` **может возвращать ошибки**, для которых требуется обработка:

- Не найдена таблица обогащения.
- Таблица не закодирована в UTF-8.
- `case_sensitive` не является логическим значением.
- `from` не является строкой
- `to` не является строкой
- `separator` не является строкой.
- `join_separator` не является строкой.
- Указан `join_separator` без параметра `separator`.
- Не удается разобрать таблицу.

Примеры

Исходный код

```
rename_keys!(
  { "a/b": 123 },
  table: "table_name",
  from: "from_value",
  to: "to_value",
  separator: "/"
)
```

Результат

```
{
  "c/d": 123
}
```

unnest

Разворачивает поле объекта, которое является массивом. В результате функция возвращает массив объектов. В каждом объекте значением указанного поля будет один из элементов массива, значения остальных полей сохраняются прежними. Это аналог explode в некоторых языках.

Присваивание возвращенного массива корневому пути `.` приведет к созданию нескольких событий из `remap`.

Спецификация функции

```
unnest(path: <путь>)
:: <массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`path`	путь	Путь к полю, которое нужно развернуть.		да

Ошибки

Функция `unnest` **возвращает ошибку**, для которой требуется обработка, если поле, на которое ссылается путь, не является массивом.

Примеры

Развернуть поле

Исходный код

```
. = {"hostname": "localhost", "events": [{"message": "hello"}, {"message": "world"}]}
. = unnest(.events)
```

Результат

```
[
  {
    "events": {
      "message": "hello"
    },
    "hostname": "localhost"
  },
  {
    "events": {
      "message": "world"
    },
    "hostname": "localhost"
  }
]
```

Развернуть вложенное поле

Исходный код

```
. = unnest!(.event.messages)
```

zip

Объединяет элементы из нескольких итерируемых массивов в группы по одному элементу от каждого. Функция возвращает массив массивов, где каждый вложенный массив содержит элементы с одинаковыми индексами из всех входных массивов. Длина результирующего массива равна длине самого короткого входного массива, а все остальные элементы отбрасываются.

Функция аналогична функции `zip` в Python, а также методам `zip` в Ruby и Rust.

Если указан только один аргумент, он должен представлять собой массив, содержащий все массивы, которые нужно слить.

Спецификация функции

```
zip(array_0: <массив>, [array_1: <массив>])
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`array_0`	массив	Первый массив элементов, или массив входных массивов, если других аргументов не передано.		да
`array_1`	массив	Второй массив элементов. Если аргумент не указан, оба массива должны быть переданы в первом аргументе.		нет

Ошибки

Функция `zip` **может возвращать ошибки**, для которых требуется обработка:

- `array_0` или `array_1` не являются массивами;
- `array_0` является единственным аргументом и содержит элементы, не являющиеся массивами.

Примеры

Слияние двух массивов разной длины

Исходный код

```
zip([1, 2, 3], [4, 5, 6, 7])
```

Результат

```
[[1, 4], [2, 5], [3, 6]]
```

Слияние трех массивов

Исходный код

```
zip([[1, 2], [3, 4], [5, 6]])
```

Результат

```
[[1, 3, 5], [2, 4, 6]]
```

Функции действий с отдельными элементами

append

Добавляет каждый элемент из массива `items` в конец массива `value`.

Спецификация функции

```
append(value: <массив>, items: <массив>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Исходный массив		да
`items`	массив	Элементы для добавления		да

Пример

Соединение массивов

Исходный код

```
append([1, 2], [3, 4])
```

Результат

```
[
  1,
  2,
  3,
  4
]
```

del

Удаляет из цели поле, указанное статическим путем `path`.



Функция `del` модифицирует текущее событие и возвращает значение удаленного поля.



Чтобы удалить поле по динамическому пути, используйте функцию `remove`.

Спецификация функции

```
del(path: <путь>, [compact: <логическое значение>])
:: <любой тип>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`path`	путь	Путь к удаляемому полю.		да
`compact`	логическое значение	При значении `compact`, равном `true`, после удаления целевого поля функция рекурсивно удаляет родительские поля, вплоть до корневого, если они становятся пустыми.	`false`	нет

Примеры

Удаление поля

Исходный код

```
del(.field1)
```

Переименование поля

Исходный код

```
.new_field = del(.old_field)
```

exists

Проверяет, существует ли путь в цели.

Эта функция отличает отсутствующий путь от существующего пути со значением `null`. Обычные выражения пути, такие как `.foo`, не позволяют различить эти случаи, так как они возвращают значение `null`, если путь не существует.

Спецификация функции

```
exists(path: <путь>)
:: <логическое значение>
```

Примеры

Проверка существования поля

Исходный код

```
exists(.field)
```

Результат

```
true
```

Проверка существования элемента массива

Исходный код

```
exists(.array[2])
```

Результат

```
true
```

get

Динамически получает значение по указанному пути.



Если точный путь известен, используйте выражение пути, такое как `.foo.bar[1]`, чтобы получить значение. Используйте функцию `get`, если имена сегментов пути не известны заранее.

Спецификация функции

```
get(value: <объект | массив>, path: <массив>)
:: <любой тип>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект массив	Объект или массив для запроса.		да
`path`	массив	Массив из сегментов пути для поиска значения.		да

Ошибки

Функция `get` **возвращает ошибку**, для которой требуется обработка, если сегмент пути не является строкой или целым числом.

Примеры

Поле верхнего уровня с путем из одного сегмента

Исходный код

```
get!(value: { "foo": "bar" }, path: ["foo"])
```

Результат

```
bar
```

Вложенное поле с путем из нескольких сегментов

Исходный код

```
get!(value: { "foo": { "bar": "baz" } }, path: ["foo", "bar"])
```

Результат

```
baz
```

Индекс массива

Исходный код

```
get!(value: ["foo", "bar", "baz"], path: [-2])
```

Результат

```
bar
```

push

Добавляет `item` в конец массива `value`.

Спецификация функции

```
push(value: <массив>, item: <любой тип>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Целевой массив.		да
`item`	любой	Элемент для добавления.		да

Примеры

Добавление элемента в массив

Исходный код

```
push([1, 2], 3)
```

Результат

```
[
  1,
  2,
  3
]
```

Добавление массива в массив

Исходный код

```
push([1, 2], [3, 4])
```

Результат

```
[
  1,
  2,
  [
    3,
    4
  ]
]
```

remove

Динамически удаляет значение по указанному пути.



Если точный путь известен, используйте более производительную функцию `del` с выражением пути, таким как `.foo.bar[1]`, чтобы удалить значение. Используйте функцию `remove`, если имена сегментов пути не известны заранее.

Спецификация функции

```
remove(value: <объект | массив>, path: <массив>, [compact: <логическое значение>])
:: <объект | массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект массив	Объект или массив, из которого требуется удалить данные.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`path`	массив	Массив сегментов пути для удаления значения.		да
`compact`	логическое значение	При значении `compact`, равном `true`, после удаления целевого поля функция рекурсивно удаляет родительские поля, вплоть до корневого, если они становятся пустыми.	`false`	нет

Ошибки

Функция `remove` **возвращает ошибку**, для которой требуется обработка, если сегмент пути не является строкой или целым числом.

Примеры

Удаление значения поля верхнего уровня с путем из одного сегмента

Исходный код

```
remove!(value: { "foo": "bar" }, path: ["foo"])
```

Результат

```
{}
```

Удаление значения вложенного поля с путем из нескольких сегментов

Исходный код

```
remove!(value: { "foo": { "bar": "baz" } }, path: ["foo", "bar"])
```

Результат

```
{
  "foo": {}
}
```

Удаление по индексу массива

Исходный код

```
remove!(value: ["foo", "bar", "baz"], path: [-2])
```

Результат

```
[
```

```
  "foo",
  "baz"
]
```

Удаление со сжатием

Исходный код

```
remove!(value: { "foo": { "bar": [42], "baz": true } }, path: ["foo", "bar", 0],
compact: true)
```

Результат

```
{
  "foo": {
    "baz": true
  }
}
```

set

Динамически присваивает значение по указанному пути в объекте или массиве.



Если точный путь известен, используйте присваивание по статическому пути, такое как `.foo.bar[1] = true`. Используйте функцию `set`, если имена сегментов пути не известны заранее.

Спецификация функции

```
set(value: <объект | массив>, path: <массив>, data: <любой тип>)
:: <объект | массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект массив	Объект или массив для вставки данных.		да
`path`	массив	Массив сегментов пути для вставки значения.		да
`data`	любой	Вставляемые данные.		да

Ошибки

Функция `set` **возвращает ошибку**, для которой требуется обработка, если сегмент пути не является строкой или целым числом.

Примеры

Присвоение значения полю верхнего уровня с путем из одного сегмента

Исходный код

```
set!(value: { "foo": "bar" }, path: ["foo"], data: "baz")
```

Результат

```
{
  "foo": "baz"
}
```

Присвоение значения вложенному полю с путем из нескольких сегментов

Исходный код

```
set!(value: { "foo": { "bar": "baz" } }, path: ["foo", "bar"], data: "qux")
```

Результат

```
{
  "foo": {
    "bar": "qux"
  }
}
```

Массив

Исходный код

```
set!(value: ["foo", "bar", "baz"], path: [-2], data: 42)
```

Результат

```
[
  "foo",
  42,
  "baz"
]
```

Функции строк



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.

- После `::` в угловых скобках указаны типы значений, которые возвращает функция.



Функции, разбирающие строки согласно регулярному выражению, описаны в разделе Функции разбора по регулярному выражению.

Функции поиска и сопоставления



Выбирайте функцию в зависимости от количества условий и требований к производительности.

contains

Определяет, содержит ли строка `value` указанную подстроку `substring`.



Чтобы проверить наличие нескольких подстрок, используйте функцию `rv_contains_any`. Функция вернет `true`, если хотя бы одна из переданных подстрок присутствует в строке.

Спецификация функции

```
contains(value: <строка>, substring: <строка>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст для поиска.		да
`substring`	строка	Подстрока, которую нужно найти в `value`.		да
`case_sensitive`	логическое значение	Должен ли поиск учитывать регистр символов.	`true`	нет

Примеры

Проверка наличия подстроки в строке с учетом регистра

Исходный код

```
contains("The Needle In The Haystack", "Needle")
```

Результат

```
true
```

Проверка наличия подстроки в строке без учета регистра

Исходный код

```
contains("The Needle In The Haystack", "needle", case_sensitive: false)
```

Результат

```
true
```

contains_all

Определяет, содержит ли строка `value` все указанные подстроки (`substrings`).



Чтобы проверить наличие в строке хотя бы одной из подстрок, используйте функцию `rv_contains_any`. Функция вернет `true`, если одна из переданных подстрок присутствует в строке.

Спецификация функции

```
contains_all(value: <строка>, substrings: <массив>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст для поиска.		да
`substrings`	массив	Массив подстрок, которые нужно найти в `value`.		да
`case_sensitive`	логическое значение	Должен ли поиск учитывать регистр символов.	`true`	нет

Примеры

Проверка наличия подстрок в строке с учетом регистра

Исходный код

```
contains_all("The Needle In The Haystack", ["Needle", "Haystack"])
```

Результат

```
true
```

Проверка наличия подстроки в строке без учета регистра

Исходный код

```
contains_all("the NEEDLE in the haystack", ["needle", "haystack"], case_sensitive:
false)
```

Результат

```
true
```

rv_contains_any

Определяет, содержится ли в строке `value` хотя бы одна подстрока из массива `substrings`.



Массив строк должен быть представлен литералом, без использования переменных. Это позволяет оптимизировать выполнение, заранее построив структуру для поиска конкретных подстрок.

Спецификация функции

```
rv_contains_any(value: <строка>, substrings: <массив>, [case_sensitive: <логическое
значение>])
:: <логическое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, в которой производится поиск подстрок.		да
`substrings`	массив	Литерал массива строк, которые ищутся в `value`.		да
`case_sensitive`	логическое значение	Определяет, учитывается ли регистр символов при поиске.	`true`	нет

Ошибки

Функция `rv_contains_any` **возвращает ошибку**, для которой требуется обработка, если подстроки содержат недопустимые данные.

Примеры

Одна из подстрок содержится в значении

Исходный код

```
rv_contains_any("eqwas_)xz!123", ["asd", "xz"])
```

Результат

```
true
```

Одна из подстрок содержится в значении без учета регистра

Исходный код

```
rv_contains_any("eqwas_)xZ!123", ["asd", "Xz"], case_sensitive: false)
```

Результат

```
true
```

Ни одна подстрока не содержится в значении

Исходный код

```
rv_contains_any("eqwas_)xz!123", ["asd", "xz1"])
```

Результат

```
false
```

starts_with

Определяет, начинается ли `value` с подстроки `substring`.



Чтобы проверить несколько подстрок, используйте функцию `rv_starts_with_any`. Функция вернет `true`, если строка начинается хотя бы с одной из переданных подстрок.

Спецификация функции

```
starts_with(value: <строка>, substring: <строка>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска.		да
`substring`	строка	Подстрока, с которой `value` должно начинаться.		да
`case_sensitive`	логическое значение	Определяет, должен ли поиск быть чувствительным к регистру.	`true`	нет

Примеры

Строка начинается с подстроки с учетом регистра

Исходный код

```
starts_with("The Needle In The Haystack", "The Needle")
```

Результат

```
true
```

Строка начинается с подстроки без учета регистра

Исходный код

```
starts_with("The Needle In The Haystack", "the needle", case_sensitive: false)
```

Результат

```
true
```

rv_starts_with_any

Определяет, начинается ли строка `value` хотя бы с одной подстроки из массива `substrings`.



Массив строк должен быть представлен литералом, без использования переменных. Это позволяет оптимизировать выполнение, заранее построив структуру для поиска конкретных подстрок.

Спецификация функции

```
rv_starts_with_any(value: <строка>, substrings: <массив>, [case_sensitive: <логическое значение>])
:: <логическое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, в которой производится поиск подстрок.		да
`substrings`	массив	Литерал массива строк, которые ищутся в `value`.		да
`case_sensitive`	логическое значение	Определяет, учитывается ли регистр символов при поиске.	`true`	нет

Ошибки

Функция `rv_starts_with_any` **возвращает ошибку**, для которой требуется обработка, если подстроки содержат недопустимые данные.

Примеры

Значение начинается с одной из подстрок

Исходный код

```
rv_starts_with_any("xzeqwas_)xz!123", ["asd", "xz"])
```

Результат

```
true
```

Значение начинается с одной из подстрок без учета регистра

Исходный код

```
rv_starts_with_any("xZeqwas_)xz!123", ["asd", "Xz"], case_sensitive: false)
```

Результат

```
true
```

Значение не начинается ни с одной из подстрок

Исходный код

```
rv_starts_with_any("xzeqwas_)xz1!123", ["asd", "xz1"])
```

Результат

```
false
```

ends_with

Определяет, заканчивается ли строка `value` указанной подстрокой `substring`.



Чтобы проверить несколько подстрок, используйте функцию `rv_ends_with_any`. Функция вернет `true`, если строка заканчивается хотя бы одной из переданных подстрок.

Спецификация функции

```
ends_with(value: <строка>, substring: <строка>, [case_sensitive: <логическое
значение>])
```

```
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска.		да
`substring`	строка	Подстрока, которой должно заканчиваться `value`.		да
`case_sensitive`	логическое значение	Определяет, должен ли поиск быть чувствительным к регистру.	`true`	нет

Примеры

Строка заканчивается указанной подстрокой с учетом регистра

Исходный код

```
ends_with("The Needle In The Haystack", "The Haystack")
```

Результат

```
true
```

Строка заканчивается указанной подстрокой без учета регистра

Исходный код

```
ends_with("The Needle In The Haystack", "the haystack", case_sensitive: false)
```

Результат

```
true
```

rv_ends_with_any

Определяет, заканчивается ли строка `value` хотя бы одной подстрокой из массива `substrings`.



Массив строк должен быть представлен литералом, без использования переменных. Это позволяет оптимизировать выполнение, заранее построив структуру для поиска конкретных подстрок.

Спецификация функции

```
rv_ends_with_any(value: <строка>, substrings: <массив>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, в которой производится поиск подстрок.		да
`substrings`	массив	Литерал массива строк, которые ищутся в `value`.		да
`case_sensitive`	логическое значение	Определяет, учитывается ли регистр символов при поиске.	`true`	нет

Ошибки

Функция `rv_ends_with_any` **возвращает ошибку**, для которой требуется обработка, если подстроки содержат недопустимые данные.

Примеры

Значение заканчивается одной из подстрок

Исходный код

```
rv_ends_with_any("eqwas_)xz!123xz", ["asd", "xz"])
```

Результат

```
true
```

Значение заканчивается одной из подстрок без учета регистра

Исходный код

```
rv_ends_with_any("eqwas_)xz!123xZ", ["asd", "Xz"], case_sensitive: false)
```

Результат

```
true
```

Значение не заканчивается ни одной из подстрок

Исходный код

```
rv_ends_with_any("eqwas_)xz!123xz", ["asd", "xz1"])
```

Результат

```
false
```

find

Определяет позицию начала первого найденного элемента в строке `value`, слева направо, который соответствует шаблону `pattern`, или возвращает -1, если элемент не найден.

Спецификация функции

```
find(value: <строка>, pattern: <регулярное выражение | строка>, [from: <целое число>])
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска паттерна.		да
`pattern`	регулярное выражение строка	Регулярное выражение или строка для сравнения.		да
`from`	целое число	Смещение, с которого начинается поиск.	`0`	нет

Примеры

Поиск текста

Исходный код

```
find("foobar", "foo")
```

Результат

```
0
```

Поиск по регулярному выражению

Исходный код

```
find("foobar", r'b.r')
```

Результат

```
3
```

Нет совпадений

Исходный код

```
find("foobar", "baz")
```

Результат

```
-1
```

С учетом смещения

Исходный код

```
find("foobarfoobarfoo", "bar", 4)
```

Результат

```
9
```

match

Определяет, соответствует ли `value` указанному шаблону `pattern`.

Спецификация функции

```
match(value: <строка>, pattern: <регулярное выражение>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Значение для проверки на соответствие.		да
`pattern`	регулярное выражение	Регулярное выражение для проверки соответствия.		да

Пример

Проверка на соответствие регулярному выражению

Исходный код

```
match("I'm a little teapot", r'teapot')
```

Результат

```
true
```

match_any

Определяет, соответствует ли `value` хотя бы одному из указанных шаблонов `patterns`. Все шаблоны проверяются за один проход по целевой строке, что дает этой функции преимущество в производительности перед многократным вызовом функции `match`.

Спецификация функции

```
match_any(value: <строка>, patterns: <массив>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Значение для проверки на соответствие.		да
`patterns`	массив	Массив регулярных выражений для проверки соответствия. Массив регулярных выражений должен быть представлен литералом, без использования переменных.		да

Пример

Проверка на соответствие хотя бы одному регулярному выражению

Исходный код

```
match_any("I'm a little teapot", [r'frying pan', r'teapot'])
```

Результат

```
true
```

Функции анализа строк

shannon_entropy

Вычисляет энтропию Шеннона для заданной строки. Функция может вычислять энтропию на основе байтов строки, кодовых точек или графем.



Разбиение на байты — самое быстрое, но может дать нежелательные результаты при работе с UTF-8. Разбиение на графемы — самое медленное, но наиболее точное в таких случаях.

Спецификация функции

```
shannon_entropy(value: <строка>, [segmentation: <string>])
:: <число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Входная строка.		да
`segmentation`	строка	Способ разбиения строки для вычисления энтропии. Допустимые значения: • `"bytes"`; • `"codepoints"`; • `"graphemes"`.	`"bytes"`	нет

Примеры

Вычисление энтропии с разбиением строки на байты

Исходный код

```
floor(shannon_entropy("vector.dev"), precision: 4)
```

Результат

```
2.9219
```

Вычисление энтропии строки в UTF-8 с разбиением на байты

Исходный код

```
floor(shannon_entropy("test123%456.□□□□□.net."), precision: 4)
```

Результат

```
4.0784
```

Вычисление энтропии строки в UTF-8 с разбиением на графемы

Исходный код

```
floor(shannon_entropy("test123%456.□□□□□.net.", segmentation: "grapheme"), precision: 4)
```

Результат

```
3.9362
```

strlen

Возвращает количество символов UTF-8 в строке `value`.



Для подсчета количества байтов в строке используйте функцию `length`.



Подсчет значений Unicode иногда может отличаться от кодовых точек Unicode.

Спецификация функции

```
strlen(value: <строка>)
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка		да

Пример

Длина строки

Исходный код

```
strlen("ñandú")
```

Результат

```
5
```

Функции разбиения и слияния массивов и строк

join

Объединяет все строковые значения в массиве `value` в одну строку с возможным разделителем `separator`.



Чтобы упростить структуру кода, вместо функции `join` используйте шаблонизацию строк.

Спецификация функции

```
join(value: <массив>, [separator: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Массив строк, которые нужно объединить.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`separator`	строка	Строка, разделяющая каждый элемент при объединении.		нет

Примеры

Объединение массива без разделителя

Исходный код

```
join!(["bring", "us", "together"])
```

Результат

```
"bringustogether"
```

Объединение массива с разделителем в виде запятой

Исходный код

```
join!(["sources", "transforms", "sinks"], separator: ", ")
```

Результат

```
"sources, transforms, sinks"
```

slice

Возвращает срез массива или строки `value` между позициями `start` и `end`.

Если параметры `start` и `end` отрицательные, они отсчитываются справа от строки или массива. Если `end` указывает на позицию, которая больше длины строки или массива, возвращается срез до конца строки или массива.

Спецификация функции

```
slice(value: <массив | строка>, start: <целое число>, [end: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив строка	Строка или массив для среза.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`start`	целое число	Включительная начальная позиция. Индекс с нуля, который может быть отрицательным.		да
`end`	целое число	Исключительная конечная позиция. Индекс с нуля, который может быть отрицательным.	длина строки	нет

Примеры

Срез строки с положительным индексом

Исходный код

```
slice!("Supercalifragilisticexpialidocious", start: 5, end: 13)
```

Результат

```
"califrag"
```

Срез строки с отрицательным индексом

Исходный код

```
slice!("Supercalifragilisticexpialidocious", start: 5, end: -14)
```

Результат

```
"califragilistic"
```

split

Разделяет строку `value` с использованием шаблона `pattern`.



Чтобы разбить строку на части заданной длины в байтах, используйте функцию `chunks`.

Спецификация функции

```
split(value: <строка>, pattern: <строка | регулярное выражение>, [limit: <целое
число>])
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разделения.		да
`pattern`	строка регулярное выражение	Строка разделяется каждый раз, когда совпадает с этим шаблоном. Может быть строкой или регулярным выражением.		да
`limit`	целое число	Максимальное количество подстрок для возврата.		нет

Примеры

Разделение строки без ограничения

Исходный код

```
split("apples and pears and bananas", " and ")
```

Результат

```
[
  "apples",
  "pears",
  "bananas"]
```

Разделение строки с ограничением

Исходный код

```
split("apples and pears and bananas", " and ", limit: 2)
```

Результат

```
[
  "apples",
  "pears and bananas"
]
```

Функции редактирования строк

redact

Скрывает данные с помощью пользовательских шаблонов. Это помогает соблюдать правила и гарантировать, что конфиденциальные данные не покидают вашу сеть.

Поведение функции зависит от типа значения `value`:

- Для строк — скрывает конфиденциальные данные и возвращает новую строку.

- Для массивов — скрывает конфиденциальные данные в каждом элементе строки.
- Для объектов — маскирует конфиденциальные данные в каждом значении строки, но не в ключах.

В массивах и объектах функция рекурсивно применяет эти операции ко всем вложенным массивам и объектам. Любые нестроковые элементы будут пропущены.

Любой скрытый текст будет заменен на `[REDACTED]`.

Спецификация функции

```
redact(value: <строка | объект | массив>, filters: <массив>, [redactor: <строка |
объект> ]))
:: <строка | объект | массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка объект массив	Значение, в котором требуется скрыть конфиденциальные данные.		да
`filters`	массив	Список фильтров, которые будут применены к `value`. Каждый фильтр может быть указан как регулярное выражение или объект, содержащий массив регулярных выражений. Подробности приведены в разделе Фильтры.		да
`redactor`	строка объект	Значение, на которое будут заменены совпадения с фильтрами. Например, замена может быть произведена на указанную строку или хэш совпадения. Подробности приведены в разделе Способ замены.	`"full"`	нет

Фильтры

Каждый фильтр может быть указан одним из следующих способов:

- В виде регулярного выражения, которое будет использовано для скрытия соответствующего ему текста.
- В виде объекта `{"type": "pattern", "patterns": <массив>}`, где под ключом `patterns` указан массив регулярных выражений.



Данный параметр должен быть представлен литералом, без использования переменных или других динамических выражений. Это позволяет выполнить

проверку аргумента на этапе компиляции и исключить ошибки во время выполнения.

Способ замены

Аргумент задается как объект следующего вида:

```
{"type": <replacement_type>,
[replacement: <replacement_text>]
[variant: <algorithm_variant>, encoding: <encoding>]}
```

Здесь:

- `<replacement_type>` — тип используемой замены под ключом `type`. Дополнительные ключи задаются в зависимости от типа. Доступные типы:
  - `full` — замена совпадений на строку `"[REDACTED]"`.
  - `text` — замена на произвольную строку, указанную под обязательным ключом `replacement`.
  - `sha2` — хэширование совпадений по алгоритму SHA-2. Поддерживает необязательные параметры `variant` и `encoding`
  - `sha3` — хэширование совпадений по алгоритму SHA-3. Поддерживает необязательные параметры `variant` и `encoding`
- `<replacement_text>` — произвольная строка, на которую заменяются совпадения при типе `text`.
- `<algorithm_variant>` — вариант используемого алгоритма хэширования. Доступные варианты:
  - `SHA-224`;
  - `SHA-256`;
  - `SHA-384`;
  - `SHA-512` — используется по умолчанию для `sha3`;
  - `SHA-512/224` (только для `sha2`);
  - `SHA-512/256` (только для `sha2`) — используется по умолчанию для `sha2`.
- `<encoding>` — кодировка хэша в виде текста. Доступные варианты:
  - `base16`;
  - `base64` — используется по умолчанию.



Для часто используемых шаблонов замены можно использовать сокращения в виде строк:

- `"full"` эквивалентна записи `{"type": "full"}`
- `"sha2"` эквивалентна записи `{"type": "sha2", "variant": "SHA-512/256", "encoding": "base64"}`
- `"sha3"` эквивалентна записи `{"type": "sha3", "variant": "SHA3-512", "encoding": "base64"}`



Данный параметр должен быть представлен литералом, без использования переменных или других динамических выражений. Это позволяет выполнить проверку аргумента на этапе компиляции и исключить ошибки во время выполнения.

Примеры

Замена текста с помощью регулярного выражения

Исходный код

```
redact("my id is 123456", filters: [r'\d+'])
```

Результат

```
"my id is [REDACTED]"
```

Замена произвольным текстом

Исходный код

```
redact("my id is 123456",
filters: [r'\d+'],
redactor: {"type": "text", "replacement": "***"})
```

Результат

```
"my id is ***"
```

Замена хэшем SHA-3

Исходный код

```
redact("my id is 123456",
filters: [r'\d+'],
redactor: "sha3")
```

Результат

```
"my id is
ZNCdmTDI7PeeUTFnpYjLdUObdizo+bIupZdl8yqnTKGdLx6X3JIqPUlUWUoFBikX+yTR+OcvLtAqWO11NPlNJw
=="
```

Замена хэшем SHA-256 в шестнадцатеричной кодировке

Исходный код

```
redact("my id is 123456",
```

```
filters: [r'\d+'],
redactor: {"type": "sha2",
"variant": "SHA-256",
"encoding": "base16"})
```

Результат

```
"my id is 8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92"
```

replace

Заменяет все совпадения с шаблоном `pattern` из значения `value` на строку `with`.

В аргументе `with` могут использоваться именованные группы.

Спецификация функции

```
replace(value: <строка>, pattern: <регулярное выражение | строка>, with: <строка>,
[count: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Исходная строка.		да
`pattern`	регулярное выражение строка	Заменяет все совпадения с этим шаблоном. Может быть статической строкой или регулярным выражением.		да
`with`	строка	Строка, которой заменяются совпадения.		да
`count`	целое число	Максимальное количество замен. `-1` означает замену всех совпадений.	`-1`	нет

Примеры

Замена текста

Исходный код

```
replace("Apples and Bananas", "and", "not")
```

Результат

```
"Apples not Bananas"
```

Замена с помощью регулярного выражения

Исходный код

```
replace("Apples and Bananas", r'(?i)bananas', "Pineapples")
```

Результат

```
"Apples and Pineapples"
```

Замена только первого совпадения

Исходный код

```
replace("Bananas and Bananas", "Bananas", "Pineapples", count: 1)
```

Результат

```
"Pineapples and Bananas"
```

Замена с захватом групп

Исходный код

```
replace("foo123bar", r'foo(?P<num>\d+)bar', "$num")
```

Результат

```
"123"
```

replace_with

Заменяет все совпадения с шаблоном `pattern` из строки `value` с помощью замыкания.

Аргумент `pattern` принимает регулярное выражение, которое может использовать захват групп.

Синтаксис замыкания используется, чтобы вычислить строку, на которую заменяются совпадения. Замыкание принимает единственный аргумент, представляющий собой объект со следующими полями:

- `captures` — массив, элементами которого являются захваченные группы.

  Если группа является незахватывающей (то есть обозначена `?:`) то значение соответствующего элемента равно `null`.

- `string` — строка, включающая полное совпадение.
- поля, соответствующие именованным группам.

В результате функции возвращается новая строка с заменой совпадений, исходная строка не изменяется.

Спецификация функции

```
replace_with(value: <строка>, pattern: <регулярное выражение>, [count: <целое число>
]) -> |<переменная>| { <блок> }
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Исходная строка.		да
`pattern`	регулярное выражение	Шаблон для поиска совпадений. Может содержать захват групп. Нельзя использовать имена групп `captures` и `string`.		да
`count`	целое число	Максимальное количество замен. `-1` означает замену всех совпадений.	`-1`	нет

Примеры

Сделать первые буквы слов прописными

Исходный код

```
replace_with("no server available",
r'\b(\w)(\w*)') -> |match| {
    upcase!(match.captures[0]) + string!(match.captures[1])
}
```

Результат

```
"No Server Available"
```

Замена на хэш

Исходный код

```
replace_with("email from test@example.com", r'\w+@example.com') -> |match| {
    sha2(match.string, variant: "SHA-512/224")
}
```

Результат

```
"email from adf6e1bc4415d24912bd93072ad34ef825a7b6eb3bf53f68def1fc17"
```

Замена на хэш

Исходный код

```
replace_with("email from test@example.com", r'\w+@example.com') -> |match| {
    sha2(match.string, variant: "SHA-512/224")
}
```

Результат

```
"email from adf6e1bc4415d24912bd93072ad34ef825a7b6eb3bf53f68def1fc17"
```

Замена первого вхождения

Исходный код

```
replace_with("The file was not found, check the file name.", r'(?i)file', count: 1) ->
|match| {
    "object"
}
```

Результат

```
"The object was not found, check the file name."
```

Именованная группа

Исходный код

```
replace_with("level=error A message", r'level=(?P<level>\w+)') -> |match| {
    lvl = upcase!(match.level)
    "[{{lvl}}]"
}
```

Результат

```
"[ERROR] A message"
```

sieve

Сохраняет в `value` только совпадения с шаблоном `pattern`.

Функцию можно использовать для определения символов, которые разрешены в строке, и удаления всего остального.

Спецификация функции

```
sieve(value: <строка>, pattern: <регулярное выражение>, [replace_single: <строка>,
replace_repeated: <строка> ])
```

```
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Исходная строка.		да
`pattern`	регулярное выражение	Шаблон, совпадения с которым должны остаться в строке.		да
`replace_single`	строка	Строка, на которую будут заменены отдельные символы, не совпадающие с шаблоном.		нет
`replace_repeated`	строка	Строка, на которую будут заменены последовательности нескольких символов, не совпадающих с шаблоном.		нет

Примеры

Очистка строки с помощью регулярного выражения

Исходный код

```
sieve("test123%456.□□□□□.net.", r'[a-z0-9.]')
```

Результат

```
"test123456..net."
```

Очистка с указанием заменяющих строк

Исходный код

```
sieve("test123%456.□□□□□.net.", r'[a-z.0-9]', replace_single: "X", replace_repeated:
"<REMOVED>")
```

Результат

```
"test123X456.<REMOVED>.net."
```

truncate

Обрезает строку `value` до количества символов `limit`.

Спецификация функции

```
truncate(value: <строка>, limit: <целое число | число с плавающей точкой>, suffix:
<строка>)
```

```
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для обрезания.		да
`limit`	целое число число с плавающей точкой	Количество символов, после которых будет обрезана строка.		да
`suffix`	строка	Суффикс, который добавляется к обрезанным строкам.		нет

Примеры

Обрезка с многоточием

Исходный код

```
truncate("A rather long sentence.", limit: 11, suffix: "...")
```

Результат

```
"A rather lo..."
```

Обрезка с произвольным суффиксом

Исходный код

```
truncate("A rather long sentence.", limit: 11, suffix: "[TRUNCATED]")
```

Результат

```
"A rather lo[TRUNCATED]"
```

Функции изменения регистра

downcase

Преобразует строку `value` в нижний регистр, где "нижний регистр" определяется по свойству Unicode `Lowercase`.

Спецификация функции

```
downcase(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо преобразовать к нижнему регистру.		да

Пример

Приведение строки к нижнему регистру

Исходный код

```
downcase("Hello, World!")
```

Результат

```
"hello, world!"
```

upcase

Приводит `value` к верхнему регистру, который определяется согласно свойству Unicode `Uppercase`.

Спецификация функции

```
upcase(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к верхнему регистру		да

Пример

Приведение строки к верхнему регистру

Исходный код

```
upcase("Hello, World!")
```

Результат

```
"HELLO, WORLD!"
```

camelcase

Приводит строку `value` к camelCase ("верблюжьему" регистру). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
camelcase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к camelCase.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Пример

Приведение строки к camelCase

Исходный код

```
camelcase("input-string")
```

Результат

```
"inputString"
```

Приведение строки к camelCase с заданным исходным регистром

Исходный код

```
camelcase("input-string", "kebab-case")
```

Результат

```
"inputString"
```

kebabcase

Приводит строку `value` к kebab-case (регистру "шашлыка"). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
kebabcase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к kebab-case.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к kebab-case

Исходный код

```
kebabcase("InputString")
```

Результат

```
"input-string"
```

Приведение строки к kebab-case с заданным исходным регистром

Исходный код

```
kebabcase("InputString", "PascalCase")
```

Результат

```
"input-string"
```

screamingsnakecase

Приводит строку `value` к SCREAMING_SNAKE case (регистру "кричащей змеи"). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
screamingsnakecase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к SCREAMING_SNAKE case.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к SCREAMING_SNAKE case

Исходный код

```
screamingsnakecase("input-string")
```

Результат

```
"INPUT_STRING"
```

Приведение строки к SCREAMING_SNAKE case с заданным исходным регистром

Исходный код

```
screamingsnakecase("input-string", "kebab-case")
```

Результат

```
"INPUT_STRING"
```

snakecase

Приводит строку `value` к snake_case ("змеиному" регистру). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
snakecase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к snake_case.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к snake_case

Исходный код

```
snakecase("input-string")
```

Результат

```
"input_string"
```

Приведение строки к snake_case с заданным исходным регистром

Исходный код

```
snakecase("input-string", "kebab-case")
```

Результат

```
"input_string"
```

pascalcase

Приводит строку `value` к PascalCase (регистру Pascal). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
pascalcase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к PascalCase.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к PascalCase

Исходный код

```
pascalcase("input-string")
```

Результат

```
"InputString"
```

Приведение строки к PascalCase с заданным исходным регистром

Исходный код

```
pascalcase("input-string", "kebab-case")
```

Результат

```
"InputString"
```

Функции типов и форматирования



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.

- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции проверки типов

is_array

Проверяет, является ли `value` массивом.

Спецификация функции

```
is_array(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимый массив

Исходный код

```
is_array([1, 2, 3])
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_array("строка")
```

Результат

```
false
```

is_boolean

Проверяет, является ли `value` логическим значением.

Спецификация функции

```
is_boolean(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
value	любой	Значение, которое необходимо проверить		да

Примеры

Допустимое логическое значение

Исходный код

```
is_boolean(false)
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_boolean("строка")
```

Результат

```
false
```

is_empty

Проверяет, имеет ли объект, массив или строка нулевую длину.

Спецификация функции

```
is_empty(value: <объект | массив | строка>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
value	объект массив строка	Значение, которое необходимо проверить.		да

Примеры

Пустой массив

Исходный код

```
is_empty([])
```

Результат

```
true
```

Непустая строка

Исходный код

```
is_empty("строка")
```

Результат

```
false
```

Непустой объект

Исходный код

```
is_empty({"foo": "bar"})
```

Результат

```
false
```

is_float

Проверяет, является ли `value` числом с плавающей точкой.

Спецификация функции

```
is_float(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить.		да

Примеры

Допустимое число с плавающей точкой

Исходный код

```
is_float(0.577)
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_float("строка")
```

Результат

```
false
```

is_integer

Проверяет, является ли `value` целым числом.

Спецификация функции

```
is_integer(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить.		да

Примеры

Допустимое целое число

Исходный код

```
is_integer(1)
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_integer("строка")
```

Результат

```
false
```

is_ipv4

Проверяет, является ли `value` допустимым IPv4-адресом или нет.

IPv4-совместимые и IPv4-преобразованные IPv6-адреса не считаются допустимыми.

Спецификация функции

```
is_ipv4(value: <строка>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	IP-адрес, который необходимо проверить.		да

Примеры

Допустимый IPv4-адрес

Исходный код

```
is_ipv4("10.0.102.37")
```

Результат

```
true
```

Допустимый IPv6-адрес

Исходный код

```
is_ipv4("2001:0db8:85a3:0000:0000:8a2e:0370:7334")
```

Результат

```
false
```

Произвольная строка

Исходный код

```
is_ipv4("foobar")
```

Результат

```
false
```

is_ipv6

Проверяет, является ли `value` допустимым IPv6-адресом или нет.

Спецификация функции

```
is_ipv6(value: <строка>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	IP-адрес, который необходимо проверить.		да

Примеры

Допустимый IPv6-адрес

Исходный код

```
is_ipv6("2001:0db8:85a3:0000:0000:8a2e:0370:7334")
```

Результат

```
true
```

Допустимый IPv4-адрес

Исходный код

```
is_ipv6("10.0.102.37")
```

Результат

```
false
```

Произвольная строка

Исходный код

```
is_ipv6("foobar")
```

Результат

```
false
```

is_json

Проверяет, является ли `value` допустимым документом JSON.

Спецификация функции

```
is_json(value: <строка>, [variant: <строка>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Значение, которое необходимо проверить		да
`variant`	строка	Вариант типа JSON для явной проверки: • `object` — объект JSON; • `array` — массив JSON; • `string` — строковое значение в формате JSON, заключенное в кавычки; • `number` — целое число или число с плавающей запятой; • `bool` — `true` или `false`; • `null` — значение `null`.		нет

Примеры

Допустимый JSON-объект

Исходный код

```
is_json("{}")
```

Результат

```
true
```

Недопустимое значение

Исходный код

```
is_json("{")
```

Результат

```
false
```

Точный вариант

Исходный код

```
is_json("{}", variant: "object")
```

Результат

```
true
```

Недопустимый точный вариант

Исходный код

```
is_json("{}", variant: "array")
```

Результат

```
false
```

is_null

Проверяет, является ли значение `value` пустым (`null`) или нет.



Для более гибкой проверки используйте функцию `is_nullish`.

Спецификация функции

```
is_null(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Значение `null`

Исходный код

```
is_null(null)
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_null("строка")
```

Результат

```
false
```

is_nullish

Определяет, является ли `value` "nullish", то есть не имеющим значимого содержания. "Nullish" считаются значения:

- `null`;
- `"-"`;
- состоящие только из пробелов, то есть символов со свойством Unicode `White_Space`.

Спецификация функции

```
is_nullish(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить на отсутствие значимого содержания		да

Примеры

Обнаружение `null` (пустая строка)

Исходный код

```
is_nullish("")
```

Результат

```
true
```

Обнаружение `null` (строка с дефисом)

Исходный код

```
is_nullish("-")
```

Результат

```
true
```

Обнаружение `null` (пустая строка)

Исходный код

```
is_nullish("

")
```

Результат

```
true
```

is_object

Проверяет, является ли `value` объектом.

Спецификация функции

```
is_object(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимый объект

Исходный код

```
is_object({"foo": "bar"})
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_object("строка")
```

Результат

```
false
```

is_regex

Проверяет, является ли `value` регулярным выражением.

Спецификация функции

```
is_regex(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимое регулярное выражение

Исходный код

```
is_regex(r'pattern')
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_regex("строка")
```

Результат

```
false
```

is_string

Проверяет, является ли `value` строкой.

Спецификация функции

```
is_string(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимая строка

Исходный код

```
is_string("строка")
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_string([1, 2, 3])
```

Результат

```
false
```

is_timestamp

Проверяет, является ли `value` меткой времени.

Спецификация функции

```
is_timestamp(value: <любой тип>)
```

```
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить.		да

Примеры

Допустимая метка времени (timestamp)

Исходный код

```
is_timestamp(t'2021-03-26T16:00:00Z')
```

Результат

```
true
```

Несоответствующий тип

Исходный код

```
is_timestamp("строка")
```

Результат

```
false
```

rv_is_utf8

Проверяет, является ли `value` строкой, закодированной в UTF-8.

Спецификация функции

```
rv_is_utf8(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимая строка

Исходный код

```
rv_is_utf8("valid_string")
```

Результат

```
true
```

Недопустимая строка

Исходный код

```
rv_is_utf8(b"\xff123123")
```

Результат

```
false
```

Не строка

Исходный код

```
rv_is_utf8(123)
```

Результат

```
false
```

tag_types_externally

Добавляет информацию о типе ко всем вложенным скалярным значениям в предоставленном значении `value`.

Информация о типе добавляется в ключ, то есть после преобразования `value` имеет форму объекта с полями `"тип": значение`.

Спецификация функции

```
tag_types_externally(value: <любой тип>)
:: <объект | массив объектов | пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое нужно пометить типами.		да

Примеры

Пометить тип в скалярном значении

Исходный код

```
tag_types_externally(123)
```

Результат

```
{
  "integer": 123
}
```

Пометить типы в объекте

Исходный код

```
tag_types_externally({
  "message": "Hello world",
  "request": {
    "duration_ms": 67.9
  }
})
```

Результат

```
{
  "message": {
    "string": "Hello world"
  },
  "request": {
    "duration_ms": {
      "float": 67.9
    }
  }
}
```

Пометить типы в массиве

Исходный код

```
tag_types_externally(["foo", "bar"])
```

Результат

```
[
  {
    "string": "foo"
```

```
  },
  {
    "string": "bar"
  }
]
```

Типы в пустом значении

Исходный код

```
tag_types_externally(null)
```

Результат

```
null
```

Функции валидации типов

array

Возвращает `value`, если это массив, иначе возвращает ошибку. Это позволяет проверить, что возвращаемое значение является массивом и может использоваться в любой функции, которая его ожидает.

Спецификация функции

```
array(value: <любой тип>)
:: <массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, проверяемое на предмет того, является ли оно массивом.		да

Ошибки

Функция `array` **возвращает ошибку**, для которой требуется обработка, если `value` не является массивом.

Пример

Объявление массива

Исходный код

```
array!(.value)
```

Результат

```
[
  1,
  2,
  3
]
```

bool

Возвращает `value`, если это логическое значение, иначе возвращает ошибку. Это позволяет проверить, что возвращаемое значение является логическим и может использоваться в любой функции, которая его ожидает.

Спецификация функции

```
bool(value: <любой тип>)
:: <логическое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить на принадлежность к логическим.		да

Ошибки

Функция `bool` **возвращает ошибку**, для которой требуется обработка, если `value` не является логическим значением.

Пример

Объявление типа Boolean

Исходный код

```
bool!(.value)
```

Результат

```
false
```

float

Возвращает `value`, если это число с плавающей точкой, иначе возвращает ошибку. Это позволяет проверить, что возвращаемое значение является числом с плавающей точкой и может использоваться в любой функции, которая его ожидает.

Спецификация функции

```
float(value: <любое>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить на тип float.		да

Ошибки

Функция `float` **возвращает ошибку**, для которой требуется обработка, если `value` не является числом с плавающей точкой.

Пример

Объявление типа float

Исходный код

```
float!(.value)
```

Результат

```
42
```

int

Возвращает `value`, если это целое число, иначе возвращает ошибку. Это позволяет проверить, что возвращаемое значение является целым числом и может использоваться в любой функции, которая его ожидает.

Спецификация функции

```
int(value: <любой тип>)
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, для которого необходимо убедиться, что оно является целым числом.		да

Ошибки

Функция `int` **возвращает ошибку**, для которой требуется обработка, если `value` не является целым числом.

Пример

Объявление переменной типа целое число

Исходный код

```
int!(.value)
```

Результат

```
42
```

object

Возвращает `value`, если это объект, иначе возвращает ошибку. Это позволяет проверить, что возвращаемое значение является объектом и может использоваться в любой функции, которая его ожидает.

Спецификация функции

```
object(value: <любой тип>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое требуется проверить		да

Ошибки

Функция `object` **возвращает ошибку**, для которой требуется обработка, если `value` не является объектом.

Пример

Определение типа объекта

Исходный код

```
object!(.value)
```

Результат

```
{
  "field1": "value1",
  "field2": "value2"
}
```

string

Возвращает `value`, если это строка, иначе возвращает ошибку. Это позволяет проверить, что возвращаемое значение является строкой и может использоваться в любой функции, которая его ожидает.

Спецификация функции

```
string(value: <любой тип>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, для которого нужно гарантировать тип строки.		да

Ошибки

Функция `string` **возвращает ошибку**, для которой требуется обработка, если `value` не является строкой.

Пример

Объявление типа строки

Исходный код

```
string!(.message)
```

Результат

```
"{\"поле\": \"значение\"}"
```

timestamp

Возвращает `value`, если это метка времени, иначе возвращает ошибку. Это позволяет проверить, что возвращаемое значение является меткой времени и может использоваться в любой функции, которая его ожидает.

Спецификация функции

```
timestamp(value: <любой тип>)
:: <метка времени>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое нужно обеспечить меткой времени.		да

Ошибки

Функция `timestamp` **возвращает ошибку**, для которой требуется обработка, если `value` не является меткой времени.

Пример

Объявление типа метки времени

Исходный код

```
ok, err = timestamp(.timestamp)
```

Результат

```
function call error for "timestamp" at (10:31): expected timestamp, got string
```

Функции представлений чисел

format_int

Форматирует целое число `value` в строковое представление с использованием заданного основания.

Спецификация функции

```
format_int(value: <целое число>, [base: <целое число>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число	Число для форматирования.		да
`base`	целое число	Основание для форматирования числа. Должно быть между 2 и 36 включительно.	`10`	нет

Ошибки

Функция `format_int` **возвращает ошибку**, для которой требуется обработка, если основание не находится между 2 и 36.

Примеры

Форматирование как шестнадцатеричное число

Исходный код

```
format_int!(42, 16)
```

Результат

```
"2a"
```

Форматирование как отрицательное шестнадцатеричное число

Исходный код

```
format_int!(-42, 16)
```

Результат

```
"-2a"
```

format_number

Форматирует `value` в строковое представление числа.

Спецификация функции

```
format_number(value: <целое число | число с плавающей точкой>, [scale: <целое число>,
decimal_separator: <строка>, grouping_separator: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число для форматирования в строку.		да
`scale`	целое число	Количество десятичных знаков для отображения.		нет
`decimal_separator`	строка	Символ, используемый между целой и десятичной частями числа.	`.`	нет
`grouping_separator`	строка	Символ, используемый между каждой тысячной частью числа.	`,`	нет

Пример

Форматирование числа с тремя десятичными знаками

Исходный код

```
format_number(1234567.89, 3, decimal_separator: ".", grouping_separator: ",")
```

Результат

```
"1,234,567.890"
```

parse_float

Разбирает строку `value`, представляющую число в десятичной системе счисления, в число с плавающей точкой.

Спецификация функции

```
parse_float(value: <строка>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Ошибки

Функция `parse_float` **может возвращать ошибки**, для которых требуется обработка:

- `value` не может быть разобрано как число.
- Значение `NaN` не поддерживается.

Примеры

Разбор отрицательного числа

Исходный код

```
parse_float!("-42")
```

Результат

```
-42.0
```

Разбор числа с плавающей запятой

Исходный код

```
parse_float!("42.38")
```

Результат

```
42.38
```

Разбор экспоненциальной записи

Исходный код

```
parse_float!("2.5e3")
```

Результат

```
2500.0
```

parse_int

Разбирает строку `value`, представляющую число в указанной системе счисления, в целое число.

Спецификация функции

```
parse_int(value: <строка>, [base: <целое число>])
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`base`	целое число	Система счисления числа. Должна быть в диапазоне от 2 до 36 включительно. Если не указано, для определения системы счисления будет использоваться префикс строки: • `0b` — двоичная; • `0`, `0o` — восьмеричная; • `0x` — шестнадцатеричная; • в остальных случаях — десятеричная.		нет

Ошибки

Функция `parse_int` **может возвращать ошибки**, для которых требуется обработка:

- Значение `base` не находится в диапазоне от 2 до 36;
- Число не может быть разобрано в указанной системе счисления.

Примеры

Разбор десятичного числа

Исходный код

```
parse_int!("-42")
```

Результат

```
-42
```

Разбор двоичного числа

Исходный код

```
parse_int!("0b1001")
```

Результат

```
9
```

Разбор восьмеричного числа

Исходный код

```
parse_int!("0o42")
```

Результат

```
34
```

Разбор шестнадцатеричного числа

Исходный код

```
parse_int!("0x2a")
```

Результат

```
42
```

Разбор числа с явно указанной системой счисления

Исходный код

```
parse_int!("2a", 17)
```

Результат

```
44
```

to_float

Приводит `value` к типу с плавающей точкой.

Спецификация функции

```
to_float(value: <целое число | число с плавающей точкой | логическое значение | строка
| метка времени>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой логическое значение строка метка времени	Значение для преобразования в число с плавающей точкой. Должно быть преобразуемым в число с плавающей точкой, иначе будет сгенерирована ошибка.		да

Характеристики функции

Поведение функции зависит от типа `value`:

- При преобразовании логических значений `true` принимает значение 1,`false`-- 0.
- Строки преобразуются, только если состоят из десятичных чисел, знаков и десятичной точки.
- Метки времени преобразуются в формат времени Unix с указанием наносекунд.

Ошибки

Функция `to_float` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является поддерживаемым представлением числа с плавающей точкой.
- `value`, представленное меткой времени, находится вне допустимого диапазона для меток времени.

Примеры

Приведение к числу с плавающей точкой

Исходный код

```
to_float!("3.145")
```

Результат

```
3.145
```

Приведение метки времени к числу с плавающей точкой

Исходный код

```
to_float(t'2020-12-30T22:20:53.824727Z')
```

Результат

```
1609366853.824727
```

Приведение строки к числу с плавающей точкой

Исходный код

```
to_float!(".5")
```

Результат

```
0.5
```

to_int

Приводит `value` к целому числу.

Спецификация функции

```
to_int(value: <целое число | число с плавающей точкой | логическое значение | строка |
метка времени>)
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой логическое значение строка метка времени	Значение для преобразования в целое число.		да

Характеристики функции

Поведение функции зависит от типа `value`:

- При преобразовании чисел с плавающей точкой отбрасывается дробная часть.
- При преобразовании логических значений `true` преобразуется в `1`, `false` в `0`.
- Строки преобразуются, только если состоят из десятичных чисел и знаков.
- Метки времени преобразуются в формат времени Unix.

Ошибки

Функция `to_int` **может возвращать ошибки**, для которых требуется обработка:

- `value` является строкой, но текст не является целым числом
- `value` не является строкой, числом или меткой времени

Примеры

Приведение строки к целому числу

Исходный код

```
to_int!("-02")
```

Результат

```
2
```

Приведение метки времени к целому числу

Исходный код

```
to_int(t'2020-12-30T22:20:53.824727Z')
```

Результат

```
1609366853
```

Приведение числа с плавающей точкой к целому числу

Исходный код

```
to_int(2.6)
```

Результат

```
2
```

Функции преобразования чисел

abs

Вычисляет модуль числа `value`.

Спецификация функции

```
abs(value: <целое число | число с плавающей точкой>)
:: <целое число | число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число для вычисления модуля.		да

Примеры

Вычисление модуля целого числа

Исходный код

```
abs(-42)
```

Результат

```
42
```

Вычисление модуля числа с плавающей точкой

Исходный код

```
abs(-42.2)
```

Результат

```
42.2
```

ceil

Округляет `value` к большему до указанной точности `precision`.

Спецификация функции

```
ceil(value: <целое число | число с плавающей точкой>, [precision: <целое число>])
:: <целое число | число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число, которое нужно округлить вверх.		да
`precision`	целое число	Количество десятичных знаков для округления.	`0`	нет

Примеры

Округление числа к большему без указания точности

Исходный код

```
ceil(4.345)
```

Результат

```
5
```

Округление числа к большему с указанием точности

Исходный код

```
ceil(4.345, precision: 2)
```

Результат

```
4.35
```

floor

Округляет `value` вниз до указанной `precision` (точности).

Спецификация функции

```
floor(value: <целое число | число с плавающей точкой>, [precision: <целое число>])
:: <целое число | число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число, которое нужно округлить вниз.		да
`precision`	целое число	Количество десятичных знаков для округления.	`0`	нет

Примеры

Округление числа вниз без указания точности

Исходный код

```
floor(4.345)
```

Результат

```
4
```

Округление числа вниз с указанием точности

Исходный код

```
floor(4.345, precision: 2)
```

Результат

```
4.34
```

mod

Вычисляет остаток от деления `value` на `modulus`.

Спецификация функции

```
mod(value: <целое число | число с плавающей точкой>, modulus: <целое число | число с
плавающей точкой>)
:: <целое число | число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Значение, для которого применяется деление по модулю.		да
`modulus`	целое число число с плавающей точкой	Значение модуля.		да

Ошибки

Функция `mod` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является целым числом или числом с плавающей точкой.
- `modulus` не является целым числом или числом с плавающей точкой.
- `modulus` равен 0.

Пример

Вычисление остатка от деления двух целых чисел

Исходный код

```
remainder = mod(5, 2)
```

Результат

```
1
```

round

Округляет `value` до указанной точности `precision`.

Спецификация функции

```
round(value: <целое число | число с плавающей точкой>, [precision: <целое число>])
:: <целое число | число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число для округления.		да
`precision`	целое число	Количество знаков после запятой для округления.	`0`	нет

Примеры

Округление числа без указания точности

Исходный код

```
round(4.345)
```

Результат

```
4
```

Округление числа с указанием точности

Исходный код

```
round(4.345, precision: 2)
```

Результат

```
4.35
```

Функции форматирования меток времени

format_timestamp

Форматирует `value` в строковое представление времени.

Спецификация функции

```
format_timestamp(value: <метка времени>, format: <строка>, [timezone: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	метка времени	Метка времени для форматирования в текст.		да
`format`	строка	Строка формата, описанная в библиотеке Chrono.		да
`timezone`	строка	Часовой пояс, используемый при форматировании метки времени. Используйте идентификатор TZ или 'local'.		нет

Примеры

Форматирование метки времени согласно ISO8601/RFC 3339

Исходный код

```
format_timestamp!(t'2020-10-21T16:00:00Z', format: "%+")
```

Результат

```
"2020-10-21T16:00:00+00:00"
```

Форматирование метки времени (пользовательский формат)

Исходный код

```
format_timestamp!(t'2020-10-21T16:00:00Z', format: "%v %R")
```

Результат

```
"21-Oct-2020 16:00"
```

from_unix_timestamp

Преобразует целое число `value` из формата времени Unix в формат времени VRL (`timestamp`).

Преобразует из количества секунд с начала эпохи Unix по умолчанию, но также можно указать миллисекунды или наносекунды с помощью параметра `unit`.

Спецификация функции

```
from_unix_timestamp(value: <целое число>, [unit: <строка>])
```

```
:: <метка времени>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число	Целое число Unix-времени для преобразования		да
`unit`	строка	Единица измерения времени: • `seconds`; • `milliseconds`; • `microseconds`; • `nanoseconds`.	`seconds`	нет

Ошибки

Функция `from_unix_timestamp` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является целым числом.
- В аргументе `unit` указана неизвестная единица.

Примеры

Преобразование из формата Unix-времени (секунды)

Исходный код

```
from_unix_timestamp!(5)
```

Результат

```
1970-01-01T00:00:05Z
```

Преобразование из формата Unix-времени (миллисекунды)

Исходный код

```
from_unix_timestamp!(5000, unit: "milliseconds")
```

Результат

```
1970-01-01T00:00:05Z
```

Преобразование из формата Unix-времени (наносекунды)

Исходный код

```
from_unix_timestamp!(5000, unit: "nanoseconds")
```

Результат

```
1970-01-01T00:00:00.000005Z
```

parse_timestamp

Разбирает `value` в формате strptime.

По умолчанию функция использует системный часовой пояс для разбора меток времени. Для анализа временных меток без указания часового пояса, таких как 16/10/2019 12:00:00, можно использовать аргумент `timezone`.

Спецификация функции

```
parse_timestamp(value: <строка>, format: <строка>, [timezone: <строка>])
:: <метка времени>, <ошибка>
```

Аргумент	Тип	Описание	Обязателен
`value`	строка	Текст времени.	да
`format`	строка	Формат strptime.	да
`timezone`	строка	Часовой пояс в формате tz database.	нет

Ошибки

Функция `parse_timestamp` **может возвращать ошибки**, для которых требуется обработка:

- `value` не удалось разобрать с использованием предоставленного `format`;
- `value` не удалось разобрать с использованием предоставленного `timezone`.

Примеры

Разбор метки времени

Исходный код

```
parse_timestamp!("10-Oct-2020 16:00+00:00", format: "%v %R %:z")
```

Результат

```
2020-10-10T16:00:00Z
```

Разбор метки времени с указанием часового пояса

Исходный код

```
parse_timestamp!("16/10/2019 12:00:00", format: "%d/%m/%Y %H:%M:%S", timezone:
"Asia/Taipei")
```

Результат

```
2019-10-16T04:00:00Z
```

to_unix_timestamp

Преобразует значение метки времени (`value`) в метку времени Unix.

По умолчанию возвращает количество секунд с момента начала эпохи Unix, но можно также указать миллисекунды микросекунды или наносекунды в аргументе `unit`.

Спецификация функции

```
to_unix_timestamp(value: <метка времени>, [unit: <строка>])
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	метка времени	Метка времени для преобразования в Unix.		да
`unit`	строка	Единица измерения времени: • `seconds`; • `milliseconds`; • `microseconds`; • `nanoseconds`.	`seconds`	нет

Ошибки

Функция `to_unix_timestamp` **может возвращать ошибки**, для которых требуется обработка:

- `value` вне допустимого диапазона для меток времени.
- В аргументе `unit` указана неизвестная единица.

Примеры

Преобразование в метку времени Unix (секунды)

Исходный код

```
to_unix_timestamp(t'2021-01-01T00:00:00+00:00')
```

Результат

```
1609459200
```

Преобразование в метку времени Unix (миллисекунды)

Исходный код

```
to_unix_timestamp(t'2021-01-01T00:00:00Z', unit: "milliseconds")
```

Результат

```
1609459200000
```

Преобразование в метку времени Unix (наносекунды)

Исходный код

```
to_unix_timestamp(t'2021-01-01T00:00:00Z', unit: "nanoseconds")
```

Результат

```
1609459200000000000
```

Функции приведения строк и логических значений

to_bool

Приводит `value` к логическому значению.

Спецификация функции

```
to_bool(value: <логическое значение | целое число | число с плавающей точкой | пустое
значение | строка>)
:: <логическое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	логическое значение целое число число с плавающей точкой пустое значение строка	Значение для преобразования в логическое значение.		да

Ошибки

Функция `to_bool` **возвращает ошибку**, для которой требуется обработка, если `value` не является поддерживаемым представлением логического значения.

Примеры

Приведение строки к логическому значению

Исходный код

```
to_bool!("yes")
```

Результат

```
true
```

Приведение числа с плавающей точкой к логическому значению

Исходный код

```
to_bool(0.0)
```

Результат

```
false
```

Приведение целого числа к логическому значению

Исходный код

```
to_bool(0)
```

Результат

```
false
```

Приведение `null` к логическому значению

Исходный код

```
to_bool(null)
```

Результат

```
false
```

to_regex

Приводит `value` к регулярному выражению.



Компиляция регулярного выражения — это ресурсозатратная операция и может ограничивать пропускную способность Vector. Используйте эту функцию, только если вы уверены, что в задаче невозможно обойтись без нее.

Спецификация функции

```
to_regex(value: <строка>)
:: <регулярное выражение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Значение для преобразования в регулярное выражение		да

Ошибки

Функция `to_regex` **возвращает ошибку**, для которой требуется обработка, если `value` не является строкой.

Пример

Приведение к регулярному выражению

Исходный код

```
to_regex("^foo$") ?? r''
```

Результат

```
^foo$
```

to_string

Приводит `value` к строке.

Спецификация функции

```
to_string(value: <целое число | число с плавающей точкой | логическое значение |
строка | метка времени>)
:: <число с плавающей точкой | пустое значение>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой логическое значение строка метка времени пустое значение	Значение для преобразования в строку		да

Ошибки

Функция `to_string` **возвращает ошибку**, для которой требуется обработка, если `value` не является целым числом, числом с плавающей точкой, логическим значением, строкой, меткой времени или пустым значением.

Примеры

Приведение логического значения к строке

Исходный код

```
to_string(true)
```

Результат

```
"true"
```

Приведение целого числа к строке

Исходный код

```
to_string(52)
```

Результат

```
"52"
```

Приведение числа с плавающей точкой к строке

Исходный код

```
to_string(52.2)
```

Результат

```
"52.2"
```

Хэш-функции и криптографические функции



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции хэширования и контрольных сумм

crc

Вычисляет CRC — контрольную сумму `value`. Используемый алгоритм вычисления задается опционально.

Спецификация функции

```
crc(value: <строка>, [algorithm: <строка>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется контрольная сумма.		да
`algorithm`	строка	Алгоритм CRC. Допустимые значения представлены ниже.	`"CRC_32_ISO_HDLC"`	нет

▼ Допустимые алгоритмы CRC:

- `"CRC_3_GSM"`;
- `"CRC_3_ROHC"`;
- `"CRC_4_G_704"`;
- `"CRC_4_INTERLAKEN"`;
- `"CRC_5_EPC_C1G2"`;
- `"CRC_5_G_704"`;
- `"CRC_5_USB"`;
- `"CRC_6_CDMA2000_A"`;
- `"CRC_6_CDMA2000_B"`;
- `"CRC_6_DARC"`;
- `"CRC_6_GSM"`;
- `"CRC_6_G_704"`;
- `"CRC_7_MMC"`;
- `"CRC_7_ROHC"`;
- `"CRC_7_UMTS"`;
- `"CRC_8_AUTOSAR"`;
- `"CRC_8_BLUETOOTH"`;
- `"CRC_8_CDMA2000"`;
- `"CRC_8_DARC"`;
- `"CRC_8_DVB_S2"`;
- `"CRC_8_GSM_A"`;
- `"CRC_8_GSM_B"`;
- `"CRC_8_HITAG"`;
- `"CRC_8_I_432_1"`;
- `"CRC_8_I_CODE"`;
- `"CRC_8_LTE"`;
- `"CRC_8_MAXIM_DOW"`;
- `"CRC_8_MIFARE_MAD"`;
- `"CRC_8_NRSC_5"`;

- "CRC_8_OPENSAFETY";
- "CRC_8_ROHC";
- "CRC_8_SAE_J1850";
- "CRC_8_SMBUS";
- "CRC_8_TECH_3250";
- "CRC_8_WCDMA";
- "CRC_10_ATM";
- "CRC_10_CDMA2000";
- "CRC_10_GSM";
- "CRC_11_FLEXRAY";
- "CRC_11_UMTS";
- "CRC_12_CDMA2000";
- "CRC_12_DECT";
- "CRC_12_GSM";
- "CRC_12_UMTS";
- "CRC_13_BBC";
- "CRC_14_DARC";
- "CRC_14_GSM";
- "CRC_15_CAN";
- "CRC_15_MPT1327";
- "CRC_16_ARC";
- "CRC_16_CDMA2000";
- "CRC_16_CMS";
- "CRC_16_DDS_110";
- "CRC_16_DECT_R";
- "CRC_16_DECT_X";
- "CRC_16_DNP";
- "CRC_16_EN_13757";
- "CRC_16_GENIBUS";
- "CRC_16_GSM";
- "CRC_16_IBM_3740";
- "CRC_16_IBM_SDLC";
- "CRC_16_ISO_IEC_14443_3_A";
- "CRC_16_KERMIT";
- "CRC_16_LJ1200";

- "CRC_16_M17";
- "CRC_16_MAXIM_DOW";
- "CRC_16_MCRF4XX";
- "CRC_16_MODBUS";
- "CRC_16_NRSC_5";
- "CRC_16_OPENSAFETY_A";
- "CRC_16_OPENSAFETY_B";
- "CRC_16_PROFIBUS";
- "CRC_16_RIELLO";
- "CRC_16_SPI_FUJITSU";
- "CRC_16_T10_DIF";
- "CRC_16_TELEDISK";
- "CRC_16_TMS37157";
- "CRC_16_UMTS";
- "CRC_16_USB";
- "CRC_16_XMODEM";
- "CRC_17_CAN_FD";
- "CRC_21_CAN_FD";
- "CRC_24_BLE";
- "CRC_24_FLEXRAY_A";
- "CRC_24_FLEXRAY_B";
- "CRC_24_INTERLAKEN";
- "CRC_24_LTE_A";
- "CRC_24_LTE_B";
- "CRC_24_OPENPGP";
- "CRC_24_OS_9";
- "CRC_30_CDMA";
- "CRC_31_PHILIPS";
- "CRC_32_AIXM";
- "CRC_32_AUTOSAR";
- "CRC_32_BASE91_D";
- "CRC_32_BZIP2";
- "CRC_32_CD_ROM_EDC";
- "CRC_32_CKSUM";
- "CRC_32_ISCSI";

- "CRC_32_ISO_HDLC";
- "CRC_32_JAMCRC";
- "CRC_32_MEF";
- "CRC_32_MPEG_2";
- "CRC_32_XFER";
- "CRC_40_GSM";
- "CRC_64_ECMA_182";
- "CRC_64_GO_ISO";
- "CRC_64_MS";
- "CRC_64_REDIS";
- "CRC_64_WE";
- "CRC_64_XZ";
- "CRC_82_DARC";

Ошибки

Функция **может возвращать ошибки**, для которых требуется обработка:

- `value` не является строкой;
- `algorithm` не является поддерживаемым алгоритмом.

Примеры

Вычисление контрольной суммы CRC согласно алгоритму по умолчанию

Исходный код

```
crc("foo")
```

Результат

```
"2356372769"
```

Вычисление контрольной суммы CRC согласно алгоритму `CRC_32_CKSUM`

Исходный код

```
crc("foo", algorithm: "CRC_32_CKSUM")
```

Результат

```
"4271552933"
```

hmac

Вычисляет HMAC для значения `value` с использованием указанного ключа `key`. Используемый алгоритм хэширования можно указать опционально.

В большинстве случаев полученный байтовый поток следует закодировать в шестнадцатеричную строку, используя функцию `encode_base16`, или в строку в формате Base64, используя функцию `encode_base64`.

Спецификация функции

```
hmac(value: <строка>, key: <строка>, [algorithm: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для вычисления HMAC.		да
`key`	строка	Строка для использования в качестве криптографического ключа.		да
`algorithm`	строка	Алгоритм хэширования. Допустимые значения: • `"SHA1"`; • `"SHA-224"`; • `"SHA-256"`; • `"SHA-384"`; • `"SHA-512"`.	`SHA-256`	нет

Ошибки

Эта функция не возвращает ошибок, если в аргументе `algorithm` указано значение по умолчанию или строковый литерал с названием алгоритма, распознаваемым и допустимым на этапе компиляции. В противном случае функция будет возвращать ошибки.

Примеры

Вычисление сообщения HMAC с алгоритмом SHA-256 по умолчанию и кодирование в строку Base64

Исходный код

```
encode_base64(hmac("Hello there", "super-secret-key"))
```

Результат

```
"eLGE8YMviv85NPXgISRUZxstBNSU47JQdcXkUWcClmI="
```

Вычисление сообщения HMAC по алгоритму SHA-224 и кодирование в шестнадцатеричную строку

Исходный код

```
encode_base16(hmac("Hello there", "super-secret-key", algorithm: "SHA-224"))
```

Результат

```
"42fccbc2b7d22a143b92f265a8046187558a94d11ddbb30622207e90"
```

Вычисление сообщения HMAC по переменному алгоритму хэширования

Исходный код

```
.hash_algo = "SHA-256"
hmac_bytes, err = hmac("Hello there", "super-secret-key", algorithm: .hash_algo)
if err == null {
  .hmac = encode_base16(hmac_bytes)
}
```

Результат

```
"78b184f1832f8aff3934f5e0212454671b2d04d494e3b25075c5e45167029662"
```

md5

Вычисляет хэш md5 для `value`.

Спецификация функции

```
md5(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для вычисления хэша.		да

Пример

Создание хэша md5

Исходный код

```
md5("foo")
```

Результат

```
"acbd18db4cc2f85cedef654fccc4a4d8"
```

seahash

Вычисляет хэш Seahash для `value`.



Из-за ограничений в типах данных VRL-функция преобразует результат Seahash из беззнакового 64-битного целого числа в знаковый формат. Значения, превышающие максимальное значение знакового 64-битного целого числа, становятся отрицательными.

Спецификация функции

```
seahash(value: <строка>)
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется хэш.		да

Примеры

Вычисление хэша Seahash

Исходный код

```
seahash("foobar")
```

Результат

```
5348458858952426000
```

Вычисление отрицательного хэша Seahash

Исходный код

```
seahash("bar")
```

Результат

```
-2796170501982571500
```

sha1

Вычисляет хэш SHA-1 для `value`.

Спецификация функции

```
sha1(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется хэш.		да

Пример

Вычисление хэша SHA-1

Исходный код

```
sha1("foo")
```

Результат

```
0beec7b5ea3f0fdbc95d0dd47f3c5bc275da8a33
```

sha2

Вычисляет хэш SHA-2 для `value`.

Спецификация функции

```
sha2(value: <строка>, [variant: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется хэш.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`variant`	строка	Вариант алгоритма хэширования: • `"SHA-224"`; • `"SHA-256"`; • `"SHA-384"`; • `"SHA-512"`; • `"SHA-512/224"`; • `"SHA-512/256"`.	`SHA-512/256`	нет

Пример

Вычисление хэша SHA-2

Исходный код

```
sha2("foo", variant: "SHA-512/224")
```

Результат

```
d68f258d37d670cfc1ec1001a0394784233f88f056994f9a7e5e99be
```

sha3

Вычисляет хэш SHA-3 для `value`.

Спецификация функции

```
sha3(value: <строка>, [variant: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется хэш.		да
`variant`	строка	Вариант алгоритма хэширования: • `"SHA3-224"`; • `"SHA3-256"`; • `"SHA3-384"`; • `"SHA3-512"`.	`SHA3-512`	нет

Пример

Вычисление хэша SHA-3

Исходный код

```
sha3("foo", variant: "SHA3-224")
```

Результат

```
f4f6779e153c391bbd29c95e72b0708e39d9166c7cea51d1f10ef58a
```

Функции шифрования

encrypt

Шифрует строку с использованием симметричного алгоритма шифрования.

Поддерживаемые алгоритмы:

- AES-256-CFB (ключ = 32 байта, IV = 16 байт)
- AES-192-CFB (ключ = 24 байта, IV = 16 байт)
- AES-128-CFB (ключ = 16 байт, IV = 16 байт)
- AES-256-OFB (ключ = 32 байта, IV = 16 байт)
- AES-192-OFB (ключ = 24 байта, IV = 16 байт)
- AES-128-OFB (ключ = 16 байт, IV = 16 байт)
- AES-128-SIV (ключ = 32 байта, IV = 16 байт)
- AES-256-SIV (ключ = 64 байта, IV = 16 байт)
- AES-256-CTR-LE (ключ = 32 байта, IV = 16 байт)
- AES-192-CTR-LE (ключ = 24 байта, IV = 16 байт)
- AES-128-CTR-LE (ключ = 16 байт, IV = 16 байт)
- AES-256-CTR-BE (ключ = 32 байта, IV = 16 байт)
- AES-192-CTR-BE (ключ = 24 байта, IV = 16 байт)
- AES-128-CTR-BE (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-PKCS7 (ключ = 32 байта, IV = 16 байт)
- AES-192-CBC-PKCS7 (ключ = 24 байта, IV = 16 байт)
- AES-128-CBC-PKCS7 (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-ANSIX923 (ключ = 32 байта, IV = 16 байт)
- AES-192-CBC-ANSIX923 (ключ = 24 байта, IV = 16 байт)
- AES-128-CBC-ANSIX923 (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-ISO7816 (ключ = 32 байта, IV = 16 байт)
- AES-192-CBC-ISO7816 (ключ = 24 байта, IV = 16 байт)
- AES-128-CBC-ISO7816 (ключ = 16 байт, IV = 16 байт)

- AES-256-CBC-ISO10126 (ключ = 32 байта, IV = 16 байт)
- AES-192-CBC-ISO10126 (ключ = 24 байта, IV = 16 байт)
- AES-128-CBC-ISO10126 (ключ = 16 байт, IV = 16 байт)
- CHACHA20-POLY1305 (ключ = 32 байта, IV = 12 байт)
- XCHACHA20-POLY1305 (ключ = 32 байта, IV = 24 байта)
- XSALSA20-POLY1305 (ключ = 32 байта, IV = 24 байта)

Спецификация функции

```
encrypt(plaintext: <строка>, algorithm: <строка>, key: <строка>, iv: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`plaintext`	строка	Строка для шифрования.		да
`algorithm`	строка	Алгоритм, используемый для шифрования.		да
`key`	строка	Ключ для шифрования. Он должен представлять собой сырые (не закодированные) байты. Его длина должна соответствовать указанному алгоритму.		да
`iv`	строка	Вектор инициализации (IV) для шифрования. Он должен представлять собой сырые (не закодированные) байты IV. Его длина должна соответствовать указанному алгоритму.		да



Для каждого сообщения следует генерировать новый вектор IV. Вы можете использовать `random_bytes` для генерации криптографически безопасного случайного значения.

Ошибки

Функция `encrypt` **может возвращать ошибки**, для которых требуется обработка:

- Алгоритм `algorithm` не поддерживается.
- Длина `key` не соответствует требуемому размеру ключа для указанного алгоритма.
- Длина `iv` не соответствует требуемому размеру вектора инициализации для указанного алгоритма.

Пример

Шифрование значения

Исходный код

```
plaintext = "super secret message"
iv = "1234567890123456" # Обычно следует вызывать random_bytes(16).
key = "16_byte_keyxxxxx"
encrypted_message = encrypt!(plaintext, "AES-128-CBC-PKCS7", key, iv: iv)
encode_base64(encrypted_message)
```

Результат

```
"GBw8Mu00v0Kc38+/PvsVtGgWuUJ+ZNLgF8Opy8ohIYE="
```

decrypt

Дешифрует строку с помощью симметричного алгоритма шифрования.

Поддерживаемые алгоритмы:

- AES-256-CFB (ключ = 32 байта, iv = 16 байт)
- AES-192-CFB (ключ = 24 байта, iv = 16 байт)
- AES-128-CFB (ключ = 16 байт, iv = 16 байт)
- AES-256-OFB (ключ = 32 байта, iv = 16 байт)
- AES-192-OFB (ключ = 24 байта, iv = 16 байт)
- AES-128-OFB (ключ = 16 байт, iv = 16 байт)
- AES-128-SIV (ключ = 32 байта, IV = 16 байт)
- AES-256-SIV (ключ = 64 байта, IV = 16 байт)
- AES-256-CTR-LE (ключ = 32 байта, IV = 16 байт)
- AES-192-CTR-LE (ключ = 24 байта, IV = 16 байт)
- AES-128-CTR-LE (ключ = 16 байт, IV = 16 байт)
- AES-256-CTR-BE (ключ = 32 байта, IV = 16 байт)
- AES-192-CTR-BE (ключ = 24 байта, IV = 16 байт)
- AES-128-CTR-BE (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-PKCS7 (ключ = 32 байта, iv = 16 байт)
- AES-192-CBC-PKCS7 (ключ = 24 байта, iv = 16 байт)
- AES-128-CBC-PKCS7 (ключ = 16 байт, iv = 16 байт)
- AES-256-CBC-ANSIX923 (ключ = 32 байта, iv = 16 байт)
- AES-192-CBC-ANSIX923 (ключ = 24 байта, iv = 16 байт)
- AES-128-CBC-ANSIX923 (ключ = 16 байт, iv = 16 байт)
- AES-256-CBC-ISO7816 (ключ = 32 байта, iv = 16 байт)
- AES-192-CBC-ISO7816 (ключ = 24 байта, iv = 16 байт)

- AES-128-CBC-ISO7816 (ключ = 16 байт, iv = 16 байт)
- AES-256-CBC-ISO10126 (ключ = 32 байта, iv = 16 байт)
- AES-192-CBC-ISO10126 (ключ = 24 байта, iv = 16 байт)
- AES-128-CBC-ISO10126 (ключ = 16 байт, iv = 16 байт)
- CHACHA20-POLY1305 (ключ = 32 байта, IV = 12 байт)
- XCHACHA20-POLY1305 (ключ = 32 байта, IV = 24 байта)
- XSALSA20-POLY1305 (ключ = 32 байта, IV = 24 байта)

Спецификация функции

```
decrypt(ciphertext: <строка>, algorithm: <строка>, key: <строка>, iv: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`ciphertext`	строка	Строка, которую необходимо расшифровать. Она должна представлять собой сырые байты (не закодированные).		да
`algorithm`	строка	Алгоритм, который необходимо использовать.		да
`key`	строка	Ключ для расшифровки. Он должен представлять собой сырые (не закодированные) байты. Его длина должна соответствовать указанному алгоритму.		да
`iv`	строка	Вектор инициализации (IV) для расшифровки. Он должен представлять собой сырые (не закодированные) байты IV. Его длина должна соответствовать указанному алгоритму.		да



Для каждого сообщения следует генерировать новый вектор IV. Вы можете использовать `random_bytes` для генерации криптографически безопасного случайного значения. Значение должно совпадать с тем, что использовалось во время шифрования.

Ошибки

Функция `decrypt` **может возвращать ошибки**, для которых требуется обработка:

- Алгоритм `algorithm` не поддерживается.
- Длина `key` не соответствует требуемому размеру ключа для указанного алгоритма.
- Длина `iv` не соответствует требуемому размеру IV для указанного алгоритма.

Пример

Расшифровка значения

Исходный код

```
ciphertext = decode_base64!("5fLGcu1VHdzsPcGNDio7asLqE1P43QrVfPfmP4i4zOU=")
iv = decode_base64!("fVEIRkIiczCRWNxaarsyxA==")
key = "16_byte_keyxxxxx"
decrypt!(ciphertext, "AES-128-CBC-PKCS7", key, iv: iv)
```

Результат

```
"super_secret_message"
```

Функции генерации случайных значений и ID



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции генерации случайных значений

random_bool

Возвращает случайное логическое значение.

Спецификация функции

```
random_bool()
:: <логическое значение>
```

Пример

Случайное логическое значение

Исходный код

```
is_boolean(random_bool())
```

Результат

```
true
```

random_int

Возвращает случайное целое число в диапазоне [`min`, `max`).



Для более удобного синтаксиса используйте функцию `rv_random_int`.

Спецификация функции

```
random_int(min: <целое число>, max: <целое число>)
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`min`	целое число	Минимальное значение (включительно)		да
`max`	целое число	Максимальное значение (исключительно)		да

Ошибки

Функция `random_int` **возвращает ошибку**, для которой требуется обработка, если `max` равен или меньше `min`.

Пример

Случайное целое число от 0 до 10 (не включая 10)

Исходный код

```
i = random_int(0, 10)
i >= 0 && i < 10
```

Результат

```
true
```

rv_random_int

Возвращает псевдослучайное целое число в диапазоне [`min`, `max`]. Любую из границ диапазона можно опустить, по умолчанию в качестве минимума используется `0`, а в качестве максимума — максимально возможное 64-битное число.

Спецификация функции

```
rv_random_int([min: <целое число>, max: <целое число>])
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`min`	целое число	Минимальное значение (включительно)	`0`	нет
`max`	целое число	Максимальное значение (включительно)	2^{63} - 1	нет

Пример

Случайное целое число от 0 до 10 (включая 10)

Исходный код

```
i = rv_random_int(0, 10)
i >= 0 && i <= 10
```

Результат

```
true
```

random_float

Возвращает случайное число с плавающей точкой в диапазоне [min, max).

Спецификация функции

```
random_float(min: <число с плавающей точкой>, max: <число с плавающей точкой>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`min`	число с плавающей точкой	Минимальное значение (включительно).		да
`max`	число с плавающей точкой	Максимальное значение (исключительно).		да

Ошибки

Функция `random_float` **возвращает ошибку**, для которой требуется обработка, если `max` равен или меньше `min`.

Пример

Случайное число с плавающей точкой от 0.0 до 10.0 (не включая 10.0)

Исходный код

```
f = random_float(0.0, 10.0)
f >= 0 && f < 10
```

Результат

```
true
```

random_bytes

Криптографически стойкий генератор случайных чисел. Возвращает строковое значение, содержащее запрошенное количество случайных байтов.

Спецификация функции

```
random_bytes(length: <целое число>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`length`	целое число	Количество генерируемых байт. Должно быть не больше 64 Кбайт.		да

Ошибки

Функция `random_bytes` **может возвращать ошибки**, для которых требуется обработка:

- `length` отрицательная
- `length` больше максимального значения (64 Кбайт)

Пример

Генерация случайных байтов в кодировке base64

Исходный код

```
encode_base64(random_bytes(16))
```

Результат

```
"LNu0BBgUbh7XAlXbjSOomQ=="
```

Функции генерации ID

uuid_from_friendly_id

Конвертирует Friendly ID (base62-кодировка 128-битного слова) в UUID.

Спецификация функции

```
uuid_from_friendly_id(value: <метка времени> )
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	метка времени	Строка, являющаяся Friendly ID		да

Ошибки

Функция `uuid_from_friendly_id` **может возвращать ошибки**, для которых требуется обработка:

- **value** является строкой, но в тексте используются символы, не принадлежащие классу `[0-9A-Za-z]`.
- **value** является base62-кодировкой целого числа, но число больше либо равно 2^{128}.

Примеры

Конвертация Friendly ID в UUID

Исходный код

```
uuid_from_friendly_id!("3s87yEvnmkiPBMHsj8bwwc")
```

Результат

```
"7f41deed-d5e2-8b5e-7a13-ab4ff93cfad2"
```

uuid_v4

Генерирует случайную строку UUIDv4.

Спецификация функции

```
uuid_v4()
:: <строка>
```

Пример

Создание UUIDv4

Исходный код

```
uuid_v4()
```

Результат

```
"1d262f4f-199b-458d-879f-05fd0a5f0683"
```

uuid_v7

Генерирует случайную строку UUIDv7. По умолчанию в качестве метки времени используется результат функции `now()`.

Спецификация функции

```
uuid_v7([timestamp: <метка времени>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`timestamp`	метка времени	Метка времени, используемая для генерации UUIDv7.	`now()`	нет

Ошибки

Функция `to_unix_timestamp` **возвращает ошибку**, для которой требуется обработка, если `timestamp` находится вне допустимого диапазона для меток времени.

Примеры

Создание UUIDv7 с меткой времени по умолчанию

Исходный код

```
uuid_v7()
```

Результат

```
"06338364-8305-7b74-8000-de4963503139"
```

Создание UUIDv7 с явно указанной `now`

Исходный код

```
uuid_v7(now())
```

Результат

```
06338364-8305-7b74-8000-de4963503139
```

Создание UUIDv7 с произвольной меткой времени

Исходный код

```
uuid_v7(t'2020-12-30T22:20:53.824727Z')
```

Результат

```
"0176b5bd-5d19-7394-bb60-c21028c6152b"
```

community_id

Генерирует ID потока данных на основе спецификации Community ID.

Спецификация функции

```
community_id(source_ip: <строка>, destination_ip: <строка>, protocol: <целое число>,
[source_port: <целое число>, destination_port: <целое число>, seed: <целое число> ])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`source_ip`	строка	IP-адрес источника.		да
`destination_ip`	строка	IP-адрес назначения.		да
`protocol`	целое число	Номер протокола.		да
`source_port`	целое число	Исходный порт или тип ICMP.		нет
`destination_port`	целое число	Порт назначения или код ICMP.		нет
`seed`	целое число	Произвольное входное число (seed) между 0 и 65 535.	`0`	нет

Ошибки

Функция `community_id` **может возвращать ошибки**, для которых требуется обработка:

- Невозможно разобрать IP-адрес источника.
- Невозможно разобрать IP-адрес назначения.
- Номер протокола вне диапазона [0; 255].
- Номер порта источника вне диапазона [0; 65535].

- Номер порта назначения вне диапазона [0; 65535].
- Указан `seed` вне диапазона [0; 65535].

Примеры

TCP

Исходный код

```
community_id!(source_ip: "192.0.2.0",
destination_ip: "198.51.100.0",
source_port: 1122,
destination_port: 3344,
protocol: 6)
```

Результат

```
"1:yQ6Snpz/rq5kekU4Yqbt5zCjHWM="
```

UDP

Исходный код

```
community_id!(source_ip: "192.0.2.0",
destination_ip: "198.51.100.0",
source_port: 1122,
destination_port: 3344,
protocol: 17)
```

Результат

```
"1:o1hyPRzWSLSK8hSMFSCDlkppQQ0="
```

ICMP

Исходный код

```
community_id!(source_ip: "192.0.2.0",
destination_ip: "198.51.100.0",
source_port: 1122,
destination_port: 3344,
protocol: 1)
```

Результат

```
"1:mrrlg1BzCI7PkjFBx+q1431HpLg="
```

RSVP

Исходный код

```
community_id!(source_ip: "192.0.2.0",
destination_ip: "198.51.100.0",
source_port: 1122,
destination_port: 3344,
protocol: 46)
```

Результат

```
"1:fktxP6ndCPNrcDJ2XpM5DFoWtT4="
```

Системные функции



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

get_env_var

Возвращает значение переменной окружения, указанной в `name`.

Спецификация функции

```
get_env_var(name: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`name`	строка	Имя переменной окружения		да

Ошибки

Функция `get_env_var` **может возвращать ошибки**, что означает, что обработка необходима для следующих ошибок:

- Переменная окружения `name` не существует.
- Значение переменной окружения `name` не является допустимым значением Unicode.

Пример

Получить значение переменной окружения

Исходный код

```
get_env_var!("HOME")
```

Результат

```
"/root"
```

get_hostname

Возвращает имя хоста локальной системы.

Спецификация функции

```
get_hostname()
:: <строка>, <ошибка>
```

Ошибки

Функция `get_hostname` **возвращает ошибку**, для которой требуется обработка, если не удалось определить имя хоста.

Пример

Получить имя хоста

Исходный код

```
.hostname = get_hostname!()
```

get_timezone_name

Возвращает название часового пояса, заданного в конфигурации Vector. Если в конфигурации установлено значение local, то функция пытается определить часовой пояс на основе операционной системы хоста. Если определить название согласно IANA не получилось, возвращается смещение местного часового пояса от UTC в формате `"[-+] ЧЧ:ММ"`, например, `"+02:00"`.

Спецификация функции

```
get_timezone_name()
:: <строка>, <ошибка>
```

Ошибки

Функция `get_timezone_name()` **возвращает ошибку**, для которой требуется обработка, если получение информации о местном часовом поясе завершилось неудачей.

Пример

Получить имя местного часового пояса согласно IANA

Исходный код

```
.vector_timezone = get_timezone_name!()
```

now

Возвращает текущую отметку времени в часовом поясе UTC с точностью до наносекунд.

Спецификация функции

```
now()
:: <метка времени>
```

Пример

Получение текущей отметки времени

Исходный код

```
now()
```

Результат

```
2021-03-04T10:51:15.928937Z
```

Функции отладки



В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

assert

Проверяет условие `condition`, которое должно возвращать логическое значение. Если условие возвращает `false`, программа завершается с сообщением `message`.



Функцию `assert` следует использовать автономно и только тогда, когда вы хотите прервать программу. Избегайте ее использования в логических выражениях и других ситуациях, когда программа должна продолжать выполнение, даже если утверждение в условии является ложным.

Спецификация функции

```
assert(condition: <логическое значение | сравнение | логическое выражение>, [message:
<строка>])
:: <пустое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`condition`	логическое значение сравнение логическое выражение	Условие для проверки, должно возвращать логическое значение.		да
`message`	строка	Пользовательское сообщение об ошибке.		нет

Ошибки

Функция `assert` **возвращает ошибку**, для которой требуется обработка, если `condition` возвращает значение `false`

Примеры

Истинное утверждение

Исходный код

```
ok, err = assert("foo" == "foo", message: "\"foo\" must be \"foo\"!")
```

Результат

```
true
```

Ложное утверждение

Исходный код

```
assert!("foo" == "bar", message: "\"foo\" must be \"foo\"!")
```

assert_eq

Проверяет, что два выражения, `left` и `right`, имеют одинаковое значение. Если они не равны, программа прерывается с сообщением `message`.



Функцию `assert_eq` следует использовать автономно и только тогда, когда вы хотите прервать программу. Избегайте ее использования в логических выражениях

> и других ситуациях, когда программа должна продолжать выполнение, даже если утверждение в условии является ложным.

Спецификация функции

```
assert_eq(left: <любой тип>, right: <любой тип>, [message: <строка>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`left`	любой	Значение для проверки на равенство с `right`.		да
`right`	любой	Значение для проверки на равенство с `left`.		да
`message`	строка	Пользовательское сообщение об ошибке.		нет

Примеры

Верное утверждение

Исходный код

```
assert_eq!(1, 1)
```

Результат

```
true
```

Ложное утверждение

Исходный код

```
assert_eq!(127, [1, 2, 3])
```

Ложное утверждение с пользовательским сообщением

Исходный код

```
assert_eq!(1, 0, message: "Unequal integers")
```

log

Выводит `value` в стандартный вывод Vector с указанным `level`.

Спецификация функции

```
log(value: <любой тип>, [level: <строка>, rate_limit_secs: <целое число>])
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение для вывода в лог.		да
`level`	строка	Уровень логирования.	`info`	нет
`rate_limit_secs`	целое число	Определяет, что сообщение будет выводиться не чаще указанного количества секунд. Чтобы отключить ограничение частоты, установите значение `0`.	`1`	нет

Примеры

Вывод сообщения в лог

Исходный код

```
log("Hello, World!", level: "info", rate_limit_secs: 60)
```

Результат

```
null
```

Вывод ошибки в лог

Исходный код

```
ts, err = to_int(.field)
if err != null {
  log(err, level: "error")
}
```

Результат

```
null
```

Справочник ошибок

100 Unhandled root runtime error

Корневое выражение может вызывать ошибку времени выполнения, и она не обрабатывается в программе VRL.

Обоснование

VRL надежно обрабатывает ошибки и требует, чтобы все возможные ошибки времени выполнения были обработаны. Это гарантирует безотказную работу VRL и помогает обеспечить надежное выполнение программ VRL при их развертывании.

Решение

Обработайте ошибку времени выполнения с помощью одной из следующих операций:

- Присваивание
- Объединение
- Вызов функции

Примеры

Необработанная ошибка времени выполнения корневого выражения (присвоение)

Исходный код

```
get_env_var("HOST")
```

Как исправить

```
.host = get_env_var("HOST")
```

101 Malformed regex literal

Литерал регулярного выражения сформирован неправильно.

Обоснование

Недопустимые регулярные выражения не компилируются.

Решение

Регулярные выражения часто приводят к ошибкам синтаксиса. Если вы обрабатываете общий формат лога (Common Log Format), рекомендуется использовать одну из функций `parse_*` VRL.

Если вы не обнаружили подходящей функции для вашего формата, попробуйте использовать инструмент тестирования регулярных выражений Rust для тестирования и исправления вашего регулярного выражения.

Примеры

Неверный литерал регулярного выражения (общий формат)

Исходный код

```
. |= parse_regex!(.message, r'^(?P<host>[\w\.]+) - (?P<user>[\w]+) (?P<bytes_in>[\d]+)
\[?P<timestamp>.*)\] "(?P<method>[\w]+) (?P<path>.*)" (?P<status>[\d]+)
```

```
(?P<bytes_out>[\d]+)$')
```

Как исправить

```
. |= parse_common_log!(.message)
```

102 Non-boolean if expression predicate

Предикат условного выражения не вычисляется в значение логического типа.

Обоснование

VRL не поддерживает "истинные" значения (нелогические значения, которые приводятся к значению `true`, например, `1`), так как они могут привести к неожиданному поведению при использовании в условных выражениях. Это гарантирует безопасную работу VRL и надежное выполнение программ VRL при их развертывании.

Решение

Измените предикат условного выражения так, чтобы он вычислялся в значение логического типа. Полезной функцией для решения этой проблемы является `is_nullish`.

Примеры

Предикат условного выражения с логическим типом (строки)

Событие журнала

```
{
  "message": "key=value"
}
```

Исходный код

```
if .message {
. |= parse_key_value!(.message)
}
```

Как исправить

```
if exists(.message) {
. |= parse_key_value!(.message)
}
```

103 Unhandled fallible assignment

Правая часть этого присваивания может вызывать ошибку времени выполнения, но ошибка не обрабатывается.

Обоснование

VRL надежно обрабатывает ошибки и требует, чтобы все возможные ошибки времени выполнения были обработаны. Это гарантирует безотказную работу VRL и помогает обеспечить надежное выполнение программ VRL при их развертывании.

Решение

Обработайте ошибку времени выполнения с помощью одной из следующих операций:

- Присваивание
- Объединение
- Вызов функции

Примеры

Необработанное присваивание с возвратом ошибки (объединение)

Событие журнала

```
{
  "message": "key=value"
}
```

Исходный код

```
. = parse_key_value(.message)
```

Как исправить

```
. = parse_key_value(.message) ?? {}
```

Необработанное присваивание с возвратом ошибки (вызов ошибки)

Событие журнала

```
{
  "message": "key=value"
}
```

Исходный код

```
. = parse_key_value(.message)
```

Как исправить

```
. = parse_key_value!(.message)
```

Необработанное присваивание с возвратом ошибки (присвоение)

Событие журнала

```
{
  "message": "key=value"
}
```

Исходный код

```
. = parse_key_value(.message)
```

Как исправить

```
., err = parse_key_value(.message)
```

Необработанное присваивание после изменения типа переменной внутри замыкания

Исходный код

```
parsed_value = {}
for_each(object!(.value)) -> |key, value| {
    kv = parse_json!(value)
    parsed_value = set!(parsed_value, [key], kv)
}
result = ""
for_each(object!(parsed_value)) -> |_asd, value| {
  result = result + value.id + value.level + " "
}
```

Как исправить

```
result = result + value.id + value.level + " " ?? "fallback"
```

104 Unnecessary error assignment

Левая часть выражения присваивания обрабатывает ошибки, хотя правая часть **не может** вызвать ошибку.

Обоснование

Присваивание ошибки, когда это невозможно, фактически является мертвым кодом, который затрудняет чтение программы. Удаление присваивания ошибки упростит вашу программу.

Решение

Удалите присваивание ошибки.

Примеры

Ненужное присваивание ошибки (строки)

Исходный код

```
.message, err = downcase(.message)
```

Как исправить

```
.message = downcase(.message)
```

105 Undefined function

Вызывается неизвестная функция.

Решение

Ошибка обычна связана с опечатками. Исправление имени функции должно решить эту проблему.

Примеры

Неопределенная функция (опечатка)

Исходный код

```
parse_keyvalue(.message)
```

Как исправить

```
parse_key_value(.message)
```

106 Function argument arity mismatch

Выражение вызова функции вызывает функцию слишком большим перечнем аргументов.

Решение

Удалите лишние аргументы, чтобы соответствовать задокументированной сигнатуре функции.

Примеры

Несоответствие количества аргументов функции

Исходный код

```
parse_json(.message, pretty: true)
```

Как исправить

```
parse_json(.message)
```

107 Required function argument missing

Выражение вызова функции не передает обязательный аргумент.

Решение

Укажите все обязательные аргументы функции, чтобы соответствовать задокументированной сигнатуре функции.

Примеры

Отсутствует обязательный аргумент функции

Исходный код

```
parse_timestamp(.timestamp)
```

Как исправить

```
parse_timestamp(.timestamp, format: "%D")
```

108 Unknown function argument keyword

Выражение вызова функции содержит неизвестное именованное значение аргумента.

Решение

Исправьте имя так, чтобы оно соответствовало задокументированным именам аргументов функции.

Примеры

Неизвестное ключевое слово аргумента функции

Исходный код

```
parse_timestamp(.timestamp, fmt: "%D")
```

Как исправить

```
parse_timestamp(.timestamp, format: "%D")
```

110 Invalid argument type

Аргумент, переданный выражению функции, не является поддерживаемым типом.

Обоснование

VRL надежно обрабатывает ошибки и требует, чтобы все возможные ошибки времени выполнения были обработаны. Это гарантирует безотказную работу VRL и помогает обеспечить надежное выполнение программ VRL при их развертывании.

Решение

Обеспечьте тип переменной, используя соответствующий тип или функцию приведения типов.

Примеры

Неверный тип аргумента (защита с использованием значений по умолчанию)

Исходный код

```
downcase(.message)
```

Как исправить

```
.message = string(.message) ?? ""
downcase(.message)
```

Неверный тип аргумента (защита с использованием ошибок)

Исходный код

```
downcase(.message)
```

Как исправить

```
downcase(string!(.message))
```

111 Unhandled predicate error

Предикат вызывает ошибку времени выполнения, и она не обрабатывается в программе VRL.

Обоснование

VRL надежно обрабатывает ошибки и требует, чтобы все возможные ошибки времени выполнения были обработаны. Это гарантирует безотказную работу VRL и помогает обеспечить надежное выполнение программ VRL при их развертывании.

Решение

Обработайте ошибку времени выполнения с помощью одной из следующих операций:

- Присваивание
- Объединение
- Вызов функции

Примеры

Необработанная ошибка предиката (предикат)

Исходный код

```
if contains(.field, "thing") {
  log("thing")
}
```

Как исправить

```
if contains(.field, "thing") ?? false {
  log("thing")
}
```

203 Unrecognized token

Ваша программа VRL содержит токен (символ), который парсер VRL не распознает как допустимый.

Решение

Используйте допустимый токен.

Примеры

Нераспознанный токен

Исходный код

```
😁
```

Как исправить

```
#некоторое допустимое значение
"😁"
```

204 Unrecognized end-of-file (EOF)

Парсер VRL достиг конца программы в недопустимом состоянии, возможно, из-за опечатки или "висячего" выражения.

Решение

Убедитесь, что последнее выражение в программе является допустимым.

Примеры

Нераспознанный конец файла (EOF)

Исходный код

```
.field1 = "value1"
```

```
.field2 =
```

Как исправить

```
.field1 = "value1"
.field2 = "value2"
```

205 Reserved keyword

Вы использовали имя переменной, которое служит для другой цели в VRL или зарезервировано для возможного будущего использования.

Решение

Используйте другое имя переменной.

Примеры

Зарезервированное ключевое слово

Исходный код

```
else = "некое значение"
```

Как исправить

```
some_non_reserved_name = "некое значение"
```

206 Invalid numeric literal

Компилятор VRL не распознает числовой литерал как допустимый.

207 Invalid string literal

Программа VRL содержит строковый литерал, который парсер VRL не распознает как допустимый.

Решение

Убедитесь, что строка правильно заключена в одиночные или двойные кавычки.

Примеры

Недопустимый строковый литерал

Исходный код

```
"Houston, we have a problem'
```

Как исправить это

```
"Houston, we have a problem"
```

208 Invalid literal

Компилятор VRL не распознает литерал как допустимое значение.

209 Invalid escape character

Строка содержит символ экранирования, который компилятор VRL не распознает как допустимый.

300 Unexpected type

Компилятор VRL ожидал значение определенного типа, но обнаружил другой тип.

301 Type coercion error

Значение не может быть приведено к желаемому типу.

302 Remainder error

Типы не могут дать остаток.

303 Multiplication error

Типы не могут быть перемножены.

304 Division error

Левое значение не может быть разделено на правое значение.

305 Divide by zero

Попытка разделить целое число или число с плавающей точкой на ноль.

Обоснование

В отличие от некоторых других языков программирования, VRL не имеет понятия бесконечности, так как неясно, как это может быть применимо к использованию данных для наблюдения. Таким образом, деление на ноль не может иметь осмысленного результата.

Решение

Если вы знаете, что значение **обязательно** равно нулю, не делите на него. Если значение **может быть** нулем, обработайте потенциальную ошибку, вызванную операцией:

```
result, err = 27 / .some_value
if err != nil {
  # Обработать ошибку.
```

```
  ...
}
```

306 NaN float

Числа с плавающей точкой не могут иметь значение NaN (не число).

307 Addition error

Значения не могут быть сложены.



Для соединения массивов используйте функцию append.

308 Subtraction error

Правое значение не может быть вычтено из левого значения.

309 Or expression error

Значения не могут быть объединены в выражение `OR`.

310 And expression error

Значения не могут быть объединены в выражение `AND`.

311 Greater than error

Значения нельзя использовать в выражении "больше чем".

312 Greater than or equal to error

Значения нельзя использовать в выражении "больше или равно".

313 Less than error

Эти два значения нельзя использовать в выражении "меньше чем".

314 Less than or equal to error

Эти два значения нельзя использовать в выражении "меньше или равно".

315 mutation of read-only value

Это значение доступно только для чтения и не может быть удалено или изменено.

400 Unexpected expression

Компилятор VRL встретил тип выражения, которое здесь не ожидалось.

401 Invalid enum variant

VRL ожидает значение перечисления для этого аргумента, но значение, которое вы ввели для перечисления, недопустимо.

Решение

Проверьте документацию для этой функции в справочнике функций VRL, чтобы узнать, какие значения перечисления допустимы для этого аргумента.

402 Expected static expression for function argument

VRL ожидал статическое выражение для аргумента функции, но было предоставлено динамическое выражение (например, переменная).

Для некоторых аргументов функций VRL требуются статические выражения для проверки типов аргументов на этапе компиляции, чтобы избежать ошибок времени выполнения.

Решение

Замените динамический аргумент статическим выражением.

403 Invalid argument

Функции передан недопустимый аргумент. Строка ошибки содержит более подробные сведения о том, что было недопустимо.

Решение

Проверьте строку этой ошибки, чтобы узнать, что было недопустимо.

601 Invalid timestamp

Предоставленный литерал метки времени корректно оформлен (то есть использует синтаксис `t'…'`), но метка времени не соответствует формату RFC 3339.

Обоснование

Недопустимые метки времени не компилируются.

Решение

Приведите метку времени в соответствие с форматом RFC 3339.

Примеры

Неверное форматирование метки времени

Исходный код

```
.timestamp = format_timestamp!(t'next Tuesday', format: "%v %R")
```

Как исправить

```
.timestamp = format_timestamp!(t'2021-03-09T16:33:02.405806Z', format: "%v %R")
```

620 Aborting infallible function

Вы указали, что функция должна прерываться при ошибке, хотя функция не возвращает ошибки.

Обоснование

В VRL функции, которые не возвращают ошибки — это функции, которые не могут завершиться с ошибкой и не требуют обработки ошибок. Это означает, что не имеет смысла использовать `!` в вызове функции для прерывания при ошибке.

Решение

Удалите `!` из вызова функции.

Примеры

Прерывание невосприимчивой функции

Исходный код

```
encode_json!(["one", "two", "three"])
```

Как исправить

```
encode_json(["one", "two", "three"])
```

630 Fallible argument

Вы передали выражение, которое возвращает ошибки, в качестве аргумента функции.

Обоснование

В VRL выражения, которые вы передаете функциям в качестве аргументов, сами также не должны возвращать ошибки. В противном случае результат функции будет неопределенным.

Решение

Сделайте так, чтобы выражение, передаваемое в функцию, не могло возвращать ошибки, потенциально прервав его при возникновении ошибки с помощью `!`, объединив его ошибку с помощью `??` или с помощью некоторого другого метода.

Примеры

Аргумент, возвращающий ошибки

Исходный код

```
format_timestamp!(parse_timestamp!("2021-01-17T23:27:31.891948Z", "%+"), format: "%v
%R")
```

Как исправить

```
format_timestamp!(parse_timestamp!("2021-01-17T23:27:31.891948Z", "%+"), format: "%v
%R")
```

631 Fallible abort message expression

Вы передали выражение, которое возвращает ошибки, в качестве сообщения для прерывания.

Обоснование

Выражение, которое вы передаете для прерывания, не должно возвращать ошибки. В противном случае выражение прерывания может завершиться с ошибкой во время выполнения.

Решение

Сделайте так, чтобы выражение, передаваемое в функцию, не могло возвращать ошибки, потенциально прервав его при возникновении ошибки с помощью !, объединив его ошибку с помощью ?? или с помощью некоторого другого метода.

Примеры

Выражение с сообщением о прерывании (может возвращать ошибку)

Исходный код

```
abort to_syslog_level(0)
```

Как исправить

```
abort to_syslog_level(0) ?? "other"
```

640 No-op assignment

Вы присвоили значение чему-то, что не является ни переменной, ни путем.

Обоснование

Все присваивания в VRL должны быть либо путем, либо переменной. Если вы пытаетесь присвоить значение, например, символу _, эта операция считается "бездействующей" (no-op), так как она не имеет эффекта (и, следовательно, не является присваиванием).

Решение

Присвойте значение, содержащееся в правой части выражения, переменной или пути.

Примеры

Присваивание значения

Исходный код

```
_ = "пример строки"
```

Как исправить

```
# .some_string = "пример строки"
```

650 Chained comparison operators

Вы связали несколько операторов сравнения вместе таким образом, что не может получиться допустимое выражение.

Обоснование

Операторы сравнения могут работать только с двумя операндами, например, `1 != 2`. Связывание их вместе, например, `1 < 2 < 3`, приводит к бессмысленному неприменимому выражению.

Решение

Используйте операторы сравнения только с левым и правым значением. Вы **можете** связывать операции сравнения вместе при условии, что выражения правильно сгруппированы. Так, например, `a == b == c` не является допустимым, но `a == b && b == c` **является** допустимым, так как в нем используются отдельные логические выражения.

Примеры

Цепочка операторов сравнения

Исходный код

```
1 == 1 == 2
```

Как исправить

```
(1 == 1) && (1 == 2)
```

651 Unnecessary error coalescing operation

Вы использовали операцию объединения для обработки ошибки, но в данном случае левая операция не может возвращать ошибки, и правое значение после `??` никогда не достигается.

Обоснование

Операции объединения полезны, когда вы хотите указать, что должно произойти, если операция завершается с ошибкой. Например:

```
result = op1 ?? op2
```

В этом примере, если `op1` не возвращает ошибки (не может завершиться с ошибкой), то переменная `result` устанавливается в значение `op1`, а `op2` никогда не достигается.

Решение

Если левая операция не должна возвращать ошибки, удалите оператор `??` и правую операцию. Однако, если левая операция должна возвращать ошибки, удалите `!` из вызова функции и все остальное, что приводит к вызову ошибок.

652 Only objects can be merged

Вы пытаетесь слить два значения, но одно или оба из них не являются объектами.

Обоснование

Среди типов, доступных в VRL, только объекты могут быть слиты. Неясно, что означало бы слияние, например, объекта с логическим значением. Однако следует отметить, что для некоторых других типов VRL доступны операции, похожие на слияние:

- Строки можно складывать.
- Массивы можно присоединять к другим массивам функцией append.

Эти операции могут оказаться полезными, если вы случайно использовали `merge`.

Решение

Убедитесь, что оба значения, которые вы объединяете, являются объектами VRL. Если вы не уверены, является ли значение объектом, вы можете использовать функцию `object` для проверки.

660 Non-Boolean negation

Вы использовали оператор отрицания для отрицания нелогического выражения.

Обоснование

Оператор отрицания (`!`) может использоваться только с логическими значениями. Например, выражение `!false` приводит к значению `true`, в то время как `!"hello"` является бессмысленным выражением.

Решение

Используйте оператор отрицания только с логическими выражениями.

Примеры

Отрицание нелогических значений

Исходный код

```
!47
```

Как исправить

```
!(47 == null)
```

701 Call to Undefined Variable

Ссылка на неопределенную переменную.

Обоснование

Ссылка на переменную, которая не определена, приводит к неожиданному поведению и, скорее всего, связана с опечаткой.

Решение

Сначала присвойте переменной значение или исправьте опечатку в ссылке.

Примеры

Неопределенная переменная

Исходный код

```
my_variable
```

Как исправить

```
my_variable = true
my_variable
```

Неправильное имя переменной

Исходный код

```
my_variable = true
my_var
```

Как исправить

```
my_variable = true
my_variable
```

801 Usage of deprecated item

Ссылка на устаревший элемент. Обычно предлагается альтернатива, которую можно использовать вместо нее.

Обоснование

Этот элемент будет удален в будущем.

Решение

Примените предложенную альтернативу.

Советы по написанию кода на языке VRL

В данном разделе представлены рекомендации по написанию качественного кода на языке VRL.

Содержание раздела:

- Оптимизация кода
- Оформление кода

Оптимизация кода

В данном разделе рассматриваются методы оптимизации кода VRL, направленные на повышение производительности системы и упрощение работы с кодом.

Оптимизация VRL-кода включает минимизацию избыточных операций, таких как лишнее копирование объектов, неоптимальные вычисления и использование устаревших или неэффективных функций.

Снижение вычислительных затрат

Оптимизация вычислений внутри циклов

При обработке событий в секции `on_correlate` все вычисления, не зависящие от цикла, следует выносить за его пределы. Это позволяет избежать повторных затрат ресурсов на каждой итерации цикла и повышает производительность. Например, выражение `downcase(to_string(.msg) ?? "-")` не изменяется в ходе выполнения цикла, но будет вычисляться на каждой итерации, если его не вынести в отдельную переменную.

Пример с избыточными вычислениями

```
...
on_correlate: !vrl |
  # Список критичных объектов для мониторинга.
  critical_objects = ["управляемые устройства/задача обновления"]
  %state = .
  %state.check = false
  for_each(critical_objects) -> |_index, value| {
    if contains(downcase(to_string(.msg) ?? "-"), value) {
```

```
      %state.check = true
      %state.cs8 = .msg
      %state.cs8Label = "Antivirus Message"
      %state.msg = "На хосте {{ to_string(.dvc) ?? '-' }} было зафиксировано изменение
параметров критичной политики пользователем {{ to_string(.duser) ?? '-' }}"
    }
  }
...
```

В данном варианте `downcase(to_string(.msg) ?? "-")` выполняется на каждой итерации, что приводит к ненужным вычислительным затратам.

Оптимизированный код

```
on_correlate: !vrl |
  # Список критичных объектов для мониторинга.
  critical_objects = ["управляемые устройства/задача обновления"]
  # Оптимизация: предварительное вычисление значения.
  downcase_msg = downcase(to_string(.msg) ?? "-")
  %state = .
  %state.check = false
  for_each(critical_objects) -> |_index, value| {
    if contains(downcase_msg, value) {
      %state.check = true
      %state.cs8 = .msg
      %state.cs8Label = "Antivirus Message"
      %state.msg = "На хосте {{ to_string(.dvc) ?? '-' }} было зафиксировано изменение
параметров критичной политики пользователем {{ to_string(.duser) ?? '-' }}"
    }
  }
```

В данном примере выражение `downcase(to_string(.msg) ?? "-")` вычисляется один раз до начала цикла и сохраняется в переменной `downcase_msg`. Это устраняет необходимость повторного выполнения вычисления на каждой итерации. В результате оптимизации количество вызовов функций сокращается, что снижает общий объем вычислений, особенно при большом количестве элементов в массиве `critical_objects`.

Исключение избыточных вычислений при повторяющихся проверках

Когда в логике обработки данных необходимо многократно выполнять одну и ту же проверку, это может привести к избыточным вычислениям и увеличению нагрузки на систему. В VRL отсутствует механизм для прерывания цикла, но избыточные вычисления можно исключить, используя функции, такие как `rv_contains_any` или `rv_ends_with_any`, которые выполняют поиск за один проход по строке.

Код с использованием `rv_contains_any`

```
...
aliases:
  event:
```

```
    filter: !vrl |
      filePath = to_string!(.filePath) ?? ""
      rv_contains_any(filePath, [
        "/var/ossec/logs",
        "/var/run/utmp",
        "/var/log/messages",
        "/var/log/audit",
        "/var/log/secure",
        "/var/log/auth.log",
        "/var/log/kern.log",
        "/var/log/syslog",
        "/var/log/cron.log"
      ])
...
```

Код с использованием `rv_ends_with_any`

```
...
aliases:
  event:
    filter: !vrl |
      filePath = to_string!(.filePath) ?? ""
      rv_ends_with_any(filePath, [
        ".bash_history",
        ".log",
        ".audit"
      ])
...
```

Для обработки условий на наличие подстрок или совпадений строк можно также использовать функции contains или match-any. Выбор функции зависит от количества условий и требований к производительности.

Функция `contains` работает эффективно при небольшом количестве условий, например, 2–3 условия.

Код с использованием `contains`

```
...
aliases:
  event:
    filter: !vrl |
      filePath = to_string!(.filePath) ?? ""
      contains(filePath, "/var/log/audit") ||
      ends_with(filePath, ".bash_history")
...
```

При большом числе условий (5 и более) предпочтительно использовать `match_any`, поскольку это упрощает код и снижает вероятность ошибок при добавлении новых условий.

Код с использованием `match_any`

```
...
aliases:
  event:
    filter: !vrl |
      filePath = to_string!(.filePath) ?? ""
      match_any(filePath, [
        "/var/ossec/logs",
        "/var/run/utmp",
        "/var/log/messages",
        "/var/log/audit",
        "/var/log/secure",
        "/var/log/auth.log",
        "/var/log/kern.log",
        "/var/log/syslog",
        "/var/log/cron.log",
        ".bash_history"
      ])
...
```

Рекомендации по выбору функций

- `contains`: используйте для небольшого количества условий (2–3 условия), где нужно проверить наличие конкретных подстрок.
- `match_any`: предпочтительнее при большом количестве условий (5 и более условий), так как она упрощает код и повышает производительность.
- `rv_contains_any` или `rv_ends_with_any`: используйте для проверки наличия хотя бы одной подстроки или суффикса, так как эти функции выполняют поиск за один проход, обеспечивая лучшую производительность.



Учитывайте чувствительность к регистру, если это необходимо, и используйте дополнительный параметр `case_sensitive: false`.

Использование регулярных выражений

Для поиска подстрок и разбора строк рекомендуется использовать функции, которые являются наиболее оптимальными для решаемой задачи:

- Для поиска единичных подстрок — функции `starts_with`, `ends_with`, `contains`.
- * Для поиска сразу нескольких подстрок — функции `rv_starts_with_any`, `rv_ends_with_any`, `rv_contains_any`.
- Для разбора распространенных форматов логов — специализированные функции, например `parse_syslog` или `parse_grok`.
- Для случаев, когда задачу невозможно решить с помощью одной функции разбора формата — функции разбора по регулярному выражению. Скорость выполнения зависит от сложности выражения.

Пример использования регулярных выражений

```
...
on_correlate: !vrl |
  message = parse_regex_all!(.raw, r'(?P<key_value>(\S*))')
...
```

Оптимизированный код

```
...
on_correlate: !vrl |
  parsed_message = parse_key_value(.raw) ?? {}
...
```

Контроль структуры кода

Приоритеты операторов

При использовании нескольких логических операторов рекомендуется явно задавать приоритет операций с помощью скобок, чтобы избежать двусмысленных условий.

Если написать код без скобок:

```
.dvendor == "Microsoft" && .externalId == "4688" || .externalId == "4688"
```

Этот код будет интерпретирован как два отдельных блока:

- `.dvendor == "Microsoft"` И `.externalId == "4688"`
- `.externalId == "4688"`

Если цель заключается в том, чтобы условие `.dvendor == "Microsoft"` было связано с одним из значений `.externalId`, необходимо использовать скобки для группировки операторов.

Пример с использованием скобок

```
.dvendor == "Microsoft" && (.externalId == "4688" || .externalId == "1234")
```

Для сложных условий с несколькими процессами также требуется группировка:

```
.dvendor == "Microsoft" && (.dproc == "ipconfig.exe" || .dproc == "nltest.exe" ||
.dproc == "ping.exe")
```

Приведение полей к нужному типу

Рекомендуется приводить значения полей к ожидаемому типу данных. Это помогает избежать ошибок при обработке данных и обеспечивает корректное выполнение кода.

Рекомендуемые значения для различных типов данных:

Типы данных и их значения по умолчанию

Тип данных	Значение по умолчанию
`LCString`	пустая строка (`""`)
`DateTime`	`t"1970-01-01T00:00:00.000Z"`
`Enum`	`-1`
`IPv6`	`::`
`UInt`	`0`
`Array`	`[]`
`MAC`	пустая строка (`""`)

Примеры преобразования значений к нужным типам

```
to_string(.dproc) ?? ""                # Преобразование в строку LCString.
to_int(dvcpid) ?? 0                    # Преобразование в UInt.
ip_to_ipv6(agent.ip) ?? "::"           # Преобразование в IPv6.
```

Шаблонизация строк

Для формирования сообщений, генерируемых программой, рекомендуется использовать строковую шаблонизацию. Это позволяет улучшить читаемость кода и избежать ошибок, связанных с объединением строк вручную.

Вместо использования функции `join` для формирования строки, можно применить шаблонизированные строки, которые поддерживаются в VRL. В шаблонизированной строке переменные значения заключаются в фигурные скобки `{}`, и их значения автоматически подставляются в строку.

Пример кода с использованием функции `join`:

```
...
on_correlate: !vrl |
  %state = .
  %state.msg = join(["На хосте", (to_string(.dvc) ?? "-"),
                     "пользователем", (to_string(.duser) ?? "-"),
                     "было зафиксировано изменение параметров критичной политики"],
                     separator: " ") ?? "-"
...
```

В данном примере функция `join` используется для объединения строк и переменных. Этот подход требует явного указания разделителей и может приводить к сложностям при добавлении или изменении текста.

Пример кода с использованием шаблонизации:

```
...
```

```
on_correlate: !vrl |
  %state = .
  %state.msg = "На хосте {to_string(.dvc) ?? '-'} пользователем {to_string(.duser) ??
'-'} было зафиксировано изменение параметров критичной политики"
...
```

Использование шаблонизации позволяет подставлять значения переменных прямо в строку, что упрощает структуру кода и снижает вероятность ошибок.

Контроль дублирования кода

Дублирование фрагментов кода в правилах может затруднить их поддержку и привести к ошибкам при изменении кода. Избегайте повторения фрагментов или объединяйте их.

Код с дублированием

```
...
on_correlate: !vrl |
  . |= compact({
    "name": %state.name,
    "dhost": %state.dhost,
    "dntdom": %state.dntdom,  # Дублирование.
    "dvc": %state.dvc,
    "dntdom": %state.dntdom,  # Дублирование.
    ...
  })
...
```

Оптимизированный код

```
...
on_correlate: !vrl |
  . |= compact({
    "name": %state.name,
    "dhost": %state.dhost,
    "dntdom": %state.dntdom,
    "dvc": %state.dvc,
    ...
  })
...
```

Оформление кода

В этом разделе описаны требования к наименованию переменных, отступам и форматированию, которые должны быть соблюдены для улучшения читаемости кода и уменьшения вероятности ошибок в процессе разработки.

Названия переменных

Названия переменных должны быть логичными и отражать их содержание.

Пример плохого наименования

```
...
on_event: !vrl |
  x = .source_ip
...
```

Оптимизированный код

```
...
on_event: !vrl |
  source_ip_address = .source_ip
...
```

Отступы и форматирование

Отступы и форматирование должны быть аккуратными и последовательными, что упрощает чтение и понимание кода.

Пример плохого форматирования

```
if contains(.msg, "error"){state.msg = "Ошибка обнаружена"}
```

Оптимизированный код

```
if contains(.msg, "error") {
  state.msg = "Ошибка обнаружена"
}
```

Работа с VRL-сниппетами

Данный раздел описывает процесс работы с VRL-сниппетами в системе. Работа осуществляется в разделе **Инструменты** → **Сниппеты** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О сниппетах

Сниппет — это компонент системы, предназначенный для хранения шаблонов кода на языке VRL и их использования в VRL-программах.

Применение сниппетов ускоряет процесс написания программ, позволяя быстро добавлять часто

используемые фрагменты кода. Например, если один и тот же фрагмент кода планируется использовать в нескольких программах, его можно сохранить в виде сниппета, а затем с помощью этого сниппета добавить его в нужные места программ.

Вставка фрагмента кода из сниппета осуществляется в конфигурациях элементов экспертизы.



Если нужно неоднократно обращаться к блоку кода в правилах нормализации и декларативных правилах корреляции, можно создать глобальную функцию.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый сниппет.
- Поле **Поиск** предназначено для быстрого поиска сниппетов в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся сниппетов. Таблица содержит следующие столбцы:

- **Название** — имя сниппета, используемое для его идентификации в системе.
- **Описание** — краткое описание функциональности сниппета.
- **Дата создания** — дата и время создания сниппета.
- **Дата изменения** — дата и время последнего изменения сниппета.

При работе с таблицей сниппетов доступны следующие операции:

- Поиск сниппетов по полю **Название**.
- Сортировка сниппетов по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе сниппета в правой части рабочей области отображается его карточка с детальной информацией, кнопками ⋮ и **Просмотр**.

Кнопка ⋮ расположена в верхнем правом углу карточки. При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного сниппета: изменение и удаление.

Кнопка **Просмотр** расположена в нижней части карточки и предназначена для просмотра конфигурации сниппета.

Работа со сниппетами

Доступные операции над сниппетами:

- Создание сниппета
- Просмотр сниппета
- Изменение сниппета

- Удаление сниппета

Создание сниппета

Чтобы создать сниппет:

1. Перейдите в раздел **Инструменты** → **Сниппеты**. Система отобразит сведения об имеющихся сниппетах.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания сниппета.
3. Введите название сниппета. Название должно быть уникальным в рамках системы.
4. При необходимости введите описание сниппета.
5. Введите фрагмент кода на языке VRL, который будет храниться в сниппете.
6. Нажмите на кнопку **Создать**. Система создаст сниппет и отобразит соответствующее уведомление. Новый сниппет появится в списке раздела **Инструменты** → **Сниппеты** и станет доступен для вставки в элементах экспертизы.

Просмотр сниппета

Чтобы просмотреть сниппет:

1. Перейдите в раздел **Инструменты** → **Сниппеты**. Система отобразит сведения об имеющихся сниппетах.
2. Нажмите на строку сниппета в списке. Система отобразит в правой части экрана карточку этого сниппета с подробной информацией о нем.

В верхней части карточки отображается название сниппета.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка сниппета содержит следующие поля:

- **ID** — уникальный идентификационный код сниппета, генерируемый автоматически при его создании.
- **Название** — имя сниппета, используемое для его идентификации в системе.
- **Описание** — краткое описание функциональности сниппета.
- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится сниппет.
- **Дата создания** — дата и время создания сниппета.
- **Создал** — пользователь, инициировавший создание сниппета.
- **Дата изменения** — дата и время последнего изменения сниппета.
- **Изменил** — пользователь, выполнивший последнее изменение сниппета.

Чтобы просмотреть конфигурацию сниппета, нажмите на кнопку **Просмотр** в нижней части карточки. Откроется окно просмотра его настроек.

Изменение сниппета

Вы можете изменить сниппет следующими способами:

- через меню действий в карточке сниппета;
- через изменение полей в карточке сниппета.

Удаление сниппета

Вы можете удалить сниппет через меню действий в его карточке.

Работа с VRL-песочницей

Данный раздел описывает процесс работы с VRL-песочницей в системе. Работа осуществляется в разделе **Инструменты → VRL-песочница** веб-интерфейса системы.



Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О VRL-песочнице

VRL-песочница — инструмент системы, предназначенный для тестирования VRL-программ, используемых для работы с элементами экспертизы и ресурсами системы.

Песочница позволяет проверить работу программ через пользовательский интерфейс системы, который включает в себя отображение подсказок и выявление ошибок в коде.

Интерфейс раздела

Панель инструментов раздела содержит кнопку **Запустить**, которая позволяет выполнить VRL-программу.

Рабочая область раздела состоит из четырех панелей:

- **Программа** — панель для ввода кода программы на языке VRL.
- **Событие** — панель для указания события, которое будет обрабатываться программой, в формате JSON.
- **Мета** — панель для указания вспомогательных метаданных в формате JSON при их наличии. Может быть пустой.
- **Результат** — панель для вывода результата выполнения программы. Доступны следующие типы результата:
  - **Target** — результат обработки события.
  - **Metadata** — результат обработки метаданных.
  - **Expression** — результат выполнения последнего выражения программы. Применяется для проверки фильтров.



Если программа содержит ошибки или для выбранного формата результата

заданы некорректные исходные данные, то при выполнении программы на панели **Результат** отображается значение `null`.

Для расширения панелей **Программа**, **Событие** и **Мета** до размера всего экрана, используйте кнопку ⤢ в верхнем правом углу соответствующей панели. Чтобы вернуть панель в исходное положение, нажмите кнопку ⤡, также расположенную в верхнем правом углу.

Результат выполнения программы можно скопировать, используя кнопку ⧉, расположенную в верхнем правом углу панели **Результат**.

Работа с редактором кода

В разделе **VRL-песочница** доступен редактор для кода программ, событий и метаданных, расположенный на панелях **Программа**, **Событие** и **Мета** соответственно.

В редакторах кода реализованы дополнительные возможности, повышающие удобство работы с ними:

- **Нумерация строк** — каждая строка в редакторе кода имеет свой номер, начиная с 1.
- **Подсветка синтаксиса** — ключевые слова, функции и константы визуально выделяются благодаря цветовой дифференциации.
- **Подстановка функций** — система предлагает автоматическое дополнение кода при вводе, что экономит время на написание стандартных конструкций.
- **Валидация кода** — автоматическая проверка кода на ошибки с моментальным подсказками для их исправления.

Поиск и подстановка функций

В разделе **Инструменты** → **VRL-песочница** при работе на панелях **Программа**, **Событие** и **Мета** предусмотрена функция быстрого поиска и подстановки встроенных VRL-функций в код.



При обращении к глобальной функции, активному списку или таблице обогащения в VRL-песочнице будет возвращено значение `null`.

Для подстановки функции:

1. Откройте панель с перечнем встроенных функций одним из следующих способов:
   - Введите начальные буквы функции, и система автоматически покажет список функций, содержащих эти символы.
   - Используйте сочетание клавиш CTRL+SPACE для отображения списка всех доступных функций.
2. Используйте клавиши со стрелками для выбора нужной функции из списка.



Вы можете дополнительно уточнить поиск, дописывая символы в название функции.

3. Для вставки функции в код нажмите ENTER. Выбранная функция будет вставлена в код.

Просмотр и устранение ошибок

В разделе **VRL-песочница** происходит автоматическая валидация кода на панелях **Программа**, **Событие** и **Мета**. Система выделяет ошибки в коде и предлагает способы их исправления.

Для подробного просмотра ошибки:

1. Наведите курсор на подчеркнутую ошибку. Система отобразит всплывающую панель с кратким описанием ошибки.
2. Чтобы получить детальное описание, нажмите на кнопку **View Problem**. Отобразится панель с подробным описанием ошибки.

   На этой панели также отображается общее количество обнаруженных ошибок и порядковый номер текущей ошибки.

   Для навигации между ошибками используйте стрелки вверх и вниз, расположенные в верхнем правом углу панели.

   После исправления ошибок система перестанет их выделять. Панель просмотра ошибок останется открытой до тех пор, пока вы не закроете ее, нажав на крестик в верхнем правом углу.

Обработка специальных символов

Для корректной передачи строк, содержащих специальные символы, вместо стандартных двойных кавычек, необходимо применять конструкцию `s'<string_value>'`, где `<string_value>` — это строка, содержащая специальные символы.

Пример некорректного форматирования

```
{
  "raw": "CEF:0|Example|Test|1.0|Event|Test Event|1|src=192.168.0.1 dst=192.168.0.2
suser=example_user request=GET /api/test?param=1 HTTP/1.1\\r\\nHost:
example.com\\r\\nUser-Agent: test-agent\\r\\nContent-Type:
application/json\\r\\n\\r\\n{\"key\":\"value with spaces\",
\"another_key\":\"value_with_special_characters_!@#$%^&*()\"}"
}
```

Пример корректного форматирования с экранированием

```
{
  "raw": s'CEF:0|Example|Test|1.0|Event|Test Event|1|src=192.168.0.1 dst=192.168.0.2
suser=example_user request=GET /api/test?param=1 HTTP/1.1\\r\\nHost:
example.com\\r\\nUser-Agent: test-agent\\r\\nContent-Type:
application/json\\r\\n\\r\\n{"key":"value with spaces",
"another_key":"value_with_special_characters_!@#$%^&*()"}'
}
```

Тестирование VRL-программ

Для тестирования VRL-программ следуйте указанному алгоритму:

1. Перейдите в **Инструменты** → **VRL-песочница**.
2. Загрузите исходное событие в формате JSON на панели **Событие**.
3. Если требуется, добавьте метаданные в формате JSON на панели **Мета**.
4. Введите код программы на языке VRL на панели **Программа**.
5. Выберите требуемый тип объекта из выпадающего списка на панели **Результат**.
6. Запустите выполнение программы, нажав кнопку **Запустить**.

   После запуска, результаты выполнения программы отобразятся на панели **Результат** в выбранном формате.

Рассмотрим применение VRL-песочницы для тестирования следующих типов программ:

- нормализация;
- корреляция;
- условия фильтрации.

Тестирование нормализации

При выполнении нормализации происходит преобразование полей исходного события. Формат исходного события и правила преобразования задаются на панелях **Событие** и **Программа** соответственно.

Для тестирования нормализации:

1. Перейдите в раздел **Инструменты** → **VRL-песочница**.
2. Введите исходное событие в формате JSON на панели **Событие**.
3. Введите правило преобразования исходного события на языке VRL на панели **Программа**.
4. Выберите тип объекта **Target** из выпадающего списка на панели **Результат**.
5. Нажмите на кнопку **Запустить** для запуска выполнения программы.

   Если выполнение программы прошло успешно, обработанное событие отобразится на панели **Результат**.

Пример программы

VRL-программа (панель **Программа***)*

```
# Добавим временную метку.
.timestamp = now()

# Сохраним HTTP-код статуса в локальную переменную.
http_status_code = parse_int!(.http_status)

# Удалим HTTP-код статуса.
del(.http_status)

# Добавим поле статуса.
```

```
if http_status_code >= 200 && http_status_code <= 299 {
    .status = "success"
    .message = "error {{http_status_code}}"
} else {
    .status = "error"
}
```

Исходное событие (панель **Событие***)*

```
{
  "http_status": "200",
  "message": "Test event"
}
```

Пример результата (панель **Результат***, тип объекта* **Target***)*

```
{
  "message": "Test event",
  "status": "success",
  "timestamp": "2023-12-15T08:27:30.561242360Z"
}
```

В результате выполнения примера:

- Из исходного события будет удалено поле `http_status_code`.
- В исходное событие будет добавлено поле `status`, значение которого зависит от значения, хранившегося в поле `http_status_code`.
- В исходное событие будет добавлено поле `timestamp`, в котором хранится временная метка запуска выполнения программы.

Тестирование корреляции

В процессе корреляции происходит анализ исходного события, на основе которого формируется корреляционное событие. В VRL-песочнице можно задать параметры для этих процессов на различных панелях:

- Формат исходного события задается на панели **Мета**.
- Формат корреляционного события указывается на панели **Событие**.
- Анализ и обработка событий происходит на панели **Программа**.

Чтобы протестировать выполнение корреляции:

1. Перейдите в раздел **Инструменты** → **VRL-песочница**.
2. Введите исходное событие в формате JSON на панели **Мета**.
3. Введите корреляционное событие в формате JSON на панели **Событие**.
4. Введите правило обработки исходного события и создания корреляционного события на языке VRL на панели **Программа**.

5. Выберите тип объекта из выпадающего списка на панели **Результат**:
   - **Target** — для вывода созданного корреляционного события.
   - **Metadata** — для вывода обработанного исходного события.
6. Нажмите на кнопку **Запустить** для запуска выполнения программы.

   Если выполнение программы прошло успешно, обработанное событие или созданное корреляционное событие отобразится на панели **Результат**.

Пример программы

VRL-программа (панель **Программа***)*

```
# Сохраним поля из исходного события в локальную переменную.
dhost = to_string(%dhost) ?? ""
duser = to_string(%duser) ?? ""
dproc = to_string(%dproc) ?? ""

# Запишем значения полей в корреляционное событие.
.dhost = dhost
.duser = duser
.dproc = dproc

# Добавим поле с сообщением в корреляционное событие.
.message = "На хосте {{dhost}} под пользователем {{duser}} была запущена программа
{{dproc}}"
```

Корреляционное событие (панель **Событие***)*

```
{
  "id": "1",
}
```

Исходное событие (панель **Мета***)*

```
{
  "dhost": "arm",
  "dproc": "ipconfig.exe",
  "duser": "user",
  "id": "1"
}
```

Результат (панель **Результат***, тип объекта* **Target***)*

```
{
  "dhost": "arm",
  "dproc": "ipconfig.exe",
  "duser": "user",
  "id": "1",
```

```
  "message": "На хосте arm под пользователем user была запущена программа
ipconfig.exe"
}
```



В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий. Данные события находятся на панели **Событие**, а метаданные — на панели **Мета**.

В результате выполнения программы корреляции:

- В корреляционное событие будут добавлены значения полей `dhost`, `dproc`, и `duser`, взятые из исходного события.
- В корреляционное событие будет включено поле `message`, содержащее информацию о запущенной программе.

Проверка условий фильтрации

Проверка условий фильтрации в VRL-песочнице осуществляется на основе данных, введенных на панели **Событие** и/или **Мета**. Условие фильтрации задается на панели **Программа** на языке VRL.

Чтобы проверить условие фильтрации:

1. Перейдите в раздел **Инструменты** → **VRL-песочница**.
2. Введите требуемое событие в формате JSON на панели **Событие**.
3. Введите требуемые метаданные в формате JSON на панели **Мета** при их наличии.
4. Введите условие фильтрации на языке VRL на панели **Программа**.
5. Выберите тип объекта **Expression** из выпадающего списка на панели **Результат**.
6. Нажмите на кнопку **Запустить** для запуска выполнения программы.

   Если условие фильтрации выполнено, на панели **Результат** отобразится значение `true`, если не выполнено — `false`.

Пример программы

Условие фильтрации (панель **Программа***)*

```
.act == "login_failed"
```

Исходное событие (панель **Событие***)*

```
{
  "act": "login_failed",
  "dvendor": "Linux",
  "message": "Test event"
}
```

Результат (панель **Результат**, *тип объекта* **Expression**)

```
true
```

В результате выполнения примера произойдет проверка наличия значения `login_failed` в поле `act` исходного события.

Использование горячих клавиш

При работе с VRL-песочницей доступно использование горячих клавиш.

Горячие клавиши VRL-песочницы

Функция	**Клавиши и их комбинации**	**Область применения**
Отмена последнего изменения	CTRL+Z	Панели **Программа**, **Событие** и **Мета**
Повтор последнего отмененного действия	CTRL+Y	Панели **Программа**, **Событие** и **Мета**
Открытие панели со списком встроенных функций	CTRL+SPACE	Панели **Программа**, **Событие** и **Мета**
Открытие панели детального описания ошибок	ALT+F8	Панели **Программа**, **Событие** и **Мета**, где есть ошибки
Запуск выполнения программы	SHIFT+ENTER	Любое место песочницы

GraphQL API

Об API

API системы, основанный на технологии GraphQL, представляет собой комплексный интерфейс для выполнения запросов и манипуляций с данными в рамках системы.

Об API-токенах

Взаимодействие с API осуществляется через HTTP-запросы методами POST для отправки GraphQL-запросов и мутаций. Каждый запрос требует наличия авторизационного токена в заголовке запроса. API-токен выпускается системой посредством функционала ботов. Бот — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании бота, ему автоматически присваивается API-токен.

О GraphQL

GraphQL представляет собой язык запросов для API, обладающий следующими характеристиками:

- **Спецификация**: определяет правила валидации схемы на стороне сервера API, а также валидность вызовов клиента на основе этой схемы.
- **Строгая типизация**: схема GraphQL строго типизирована, описывая типы данных и отношения между объектами, что обеспечивает предсказуемость и точность ответов API.
- **Иерархичность**: структура запросов и ответов GraphQL отражает иерархическую структуру данных, позволяя клиентам получать именно те данные, которые необходимы, в одном запросе.
- **Уровень приложения**: GraphQL функционирует на уровне приложения, не завися от конкретных технологий хранения данных. Графическая структура запросов и ответов описывается в схеме API, определяющей объекты и их связи.

Начало работы

Для интеграции с API системы рекомендуется выполнение ряда действий, таких как подготовка окружения, получение токена и авторизация.

Подготовка и настройка окружения

Установите и настройте необходимые инструменты, которые поддерживают формирование и отправку GraphQL-запросов. Например, такие, как Postman, Insomnia или GraphQL IDE.

Также система включает GraphQL-песочницу — интерфейс, в котором можно просмотреть схему доступных GraphQL-запросов и выполнить их.



GraphQL-песочница по умолчанию недоступна. Чтобы открыть доступ к ней, обратитесь к разделу Открытие доступа к GraphQL-песочнице.

После открытия доступа GraphQL-песочница будет доступна из браузера по

следующему адресу:

```
<system_url>/api/graphql
```

Здесь:

- `<system_url>` — URL, по которому осуществляется доступ к веб-интерфейсу системы.

Аутентификация

Доступ к функциональности API системы требует аутентификации с использованием токена авторизации. Токен — это уникальный цифро-буквенный код, представленный в формате JWT (JSON Web Token), который включается в каждый запрос через HTTP заголовок `Authorization`.

Токен можно получить, создав учетную запись с типом Бот в веб-интерфейса системы.

Для выполнения запросов к API, необходимо включить следующие HTTP заголовки:

- `Authorization`: заголовок, содержащий токен авторизации в формате JWT (JSON Web Token). Токен предоставляет информацию о пользователе и его правах доступа.
- `x-tenant-id`: заголовок, указывающий идентификатор тенанта.

Токен передается в заголовке `Authorization` HTTP-запроса при обращении к API.



`Bearer` не является частью самого токена и добавляется в заголовок запроса в качестве префикса перед самим токеном для указания схемы аутентификации.

Необходимые HTTP заголовки для выполнения запросов:

```
{
  "Authorization": "Bearer <ваш_токен_доступа>",
  "x-tenant-id": "00000000-0000-0000-0000-000000000000"
}
```

Где `<ваш_токен_доступа>` — это актуальный токен доступа, полученный после аутентификации в системе.

Пример HTTP заголовков:

```
{
  "Authorization": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...,
  "x-tenant-id": "99311ef1-32a1-4b2d-9d38-7613be5e...."
}
```

Выполнение первого запроса

Пример запроса на получение детальной информации об элементе экспертизы:

```
query {
  expertise(id: "uniqueExpertiseId") {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

В этом запросе используется параметр:

- `id`: уникальный идентификатор элемента экспертизы, информацию о котором необходимо получить.

▼ Возвращаемый тип `Expertise`

Параметр	Описание
`id`	Уникальный идентификатор элемента экспертизы.
`sourceId`	Идентификатор исходного объекта, если применимо.
`folderId`	Идентификатор каталога, в который включен элемент экспертизы.
`type`	Тип элемента экспертизы.
`name`	Название элемента экспертизы.
`description`	Описание элемента экспертизы.
`draft`	Указывает, является ли объект черновиком.
`author`	Автор элемента экспертизы. Указывается автоматически на основе данных пользователя, выполнившего мутацию.
`status`	Статус элемента экспертизы.
`tags`	Массив тегов, ассоциированных с объектом.
`version`	Версия элемента экспертизы.
`severity`	Уровень серьезности элемента экспертизы.

Параметр	Описание
`createdAt`	Дата и время создания объекта.
`updatedAt`	Дата и время последнего обновления объекта.
`__typename`	Название типа объекта в системе GraphQL.

Выполнение запросов

Для выполнения запросов GraphQL, учитывая, что операции GraphQL состоят из многострочного JSON, рекомендуется использовать GraphQL Explorer. В качестве альтернативы можно использовать curl или любую другую HTTP-совместимую библиотеку. Все запросы GraphQL, будь то запросы (queries), мутации (mutations) или подписки (subscriptions) передаются в теле POST-запроса в формате JSON.

Пример использования curl для выполнения GraphQL запроса:

```
curl -H "Authorization: Bearer <access_token>" -H "x-tenant-id: <tenant_id>" -X POST
-d " \
 { \
   \"query\": \"query { <query> }\" \
 } \
" <link_to_graphql>
```

Здесь:

- `<access_token>` — токен доступа к системе.
- `<tenant_id>` — идентификатор тенанта в системе.
- `<query>` — запрос GraphQL, например, на получение списка событий или информации об объекте системы.
- `<link_to_graphql>` — адрес конечной точки для доступа к GraphQL API системы, например, `https://your-rvision-domain/api/graphql`.



Важно корректно экранировать специальные символы в теле JSON-запроса для обеспечения его правильной обработки сервером. Внешние двойные кавычки и экранированные внутренние двойные кавычки помогают избежать ошибок при разборе запроса.

При отсутствии необходимых разрешений у токена для доступа к запрашиваемому ресурсу, API вернет ошибку с указанием, какие разрешения необходимы для выполнения запроса.

Открытие доступа к GraphQL-песочнице

Чтобы открыть доступ к GraphQL-песочнице, выполните следующие действия:

1. Подключитесь к терминалу master-узла кластера Kubernetes, в котором установлена система.
2. Откройте ConfigMap `evo.streams-bff-env`, выполнив следующую команду:

```
kubectl -n <namespace> edit configmap evo.streams-bff-env
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

3. Установите для переменной `EVO_STREAMS_BFF_GRAPHQL_ENABLE_PLAYGROUND` значение `"true"`.
4. Сохраните изменения в ConfigMap.
5. Перезапустите сервис `bff`, выполнив следующую команду:

```
kubectl -n <namespace> rollout restart deployment/evo.streams.bff
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

После открытия доступа GraphQL-песочница будет доступна из браузера по следующему адресу:

```
<system_url>/api/graphql
```

Здесь:

- `<system_url>` — URL, по которому осуществляется доступ к веб-интерфейсу системы.

API-токены

Данный раздел описывает процесс работы с API-токенами в системе.

Об API-токенах

API системы обеспечивает программный доступ к функционалу системы через протокол HTTP. Доступные операции API включают выполнение запросов и мутаций с использованием GraphQL, что позволяет управлять данными и настройками системы.

Взаимодействие с API осуществляется через HTTP-запросы методами POST для отправки GraphQL-запросов и мутаций. Каждый запрос требует наличия авторизационного токена в заголовке запроса. API-токен выпускается системой посредством функционала ботов. Бот — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании бота, ему автоматически присваивается API-токен.

Работа с API-токенами

Доступные операции над API-токенами:

- Создание API-токена
- Копирование API-токена
- Перевыпуск API-токена
- Удаление API-токена

Создание API-токена

Чтобы выпустить API-токен, необходимо создать учетную запись типа **Бот**. При создании бота, ему автоматически присваивается API-токен, который можно использовать для создания запросов к системе.

Чтобы создать бота:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на кнопку **Добавить** (+) и выберите из выпадающего списка тип **Бот**. Система отобразит окно добавления бота.
3. Заполните следующие поля:
   - **Логин** — задайте уникальный логин для учетной записи пользователя в системе.
   - **Группа пользователей** (опционально) — выберите из выпадающего списка группы, в которые должен входить пользователь.
   - **Описание** (опционально) — добавьте краткое описание учетной записи пользователя.
4. Назначьте роли пользователю, чтобы определить перечень его разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.

   b. Выберите из выпадающего списка роль, которую необходимо назначить пользователю.

   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   > Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы удалить добавленную роль, нажмите на кнопку 🗑 в строке настройки роли.

   > Если активирован переключатель **Суперадминистратор**, учетной записи предоставляются все доступные в системе разрешения.

5. Нажмите на кнопку **Добавить**. Система создаст бота и отобразит соответствующее уведомление. Новый бот появится в списке раздела.

Копирование API-токена

Чтобы скопировать API-токен:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит

список доступных пользователей.

2. Нажмите на строку нужного бота в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.
3. Нажмите на кнопку **Скопировать токен** в нижней части карточки. Система скопирует выбранный токен в буфер обмена и отобразит соответствующее уведомление.

Перевыпуск API-токена

Чтобы сбросить период хранения токена, необходимо перевыпустить токен для бота, к которому относится нужный токен. Предыдущий токен будет удален и станет недоступным для использования. Срок жизни токена будет отсчитываться от даты перевыпуска API-токена.

Чтобы обновить API-токен:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит список доступных пользователей.
2. Нажмите на строку нужного бота в списке. Система отобразит в правой части экрана карточку этого бота с подробной информацией о нем.
3. Нажмите на кнопку **Перевыпустить токен** в нижней части карточки. Система выпустит новый API-токен и отобразит соответствующее уведомление. Предыдущий токен бота станет недействительным.

Удаление API-токена

Чтобы удалить API-токен, необходимо удалить связанного с ним бота:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Наведите курсор на строку нужного бота в списке. Нажмите на кнопку **Удалить** (), которая отображается в правой части строки. Отобразится окно подтверждения удаления бота.
3. Нажмите на кнопку **Удалить**. Система удалит бота и отобразит соответствующее уведомление. Удаленный бот будет исключен из списка пользователей. API-токен, связанный с ботом, станет недействительным.

Работа с GraphQL

В данном разделе представлено описание функционала выполнения запросов, мутаций и подписок для взаимодействия с данными системы через *GraphQL API*.

API позволяет выполнять следующие операции:

- Выгрузка событий: методы для извлечения данных о событиях, поддерживающие фильтрацию, сортировку и пагинацию.
- Управление элементами экспертизы: создание, обновление, публикация и удаление элементов экспертизы.
- Просмотр и управление каталогами: запросы для получения списка каталогов и управления ими.
- Фильтры и сортировка: применение фильтров для уточнения запросов и настройка порядка

сортировки результатов.

- Пагинация: методы для организации данных в пагинированном виде.

Выполнение запросов

Для выполнения запросов GraphQL, учитывая, что операции GraphQL состоят из многострочного JSON, рекомендуется использовать GraphQL Explorer. В качестве альтернативы можно использовать curl или любую другую HTTP-совместимую библиотеку. Все запросы GraphQL, будь то запросы (queries), мутации (mutations) или подписки (subscriptions) передаются в теле POST-запроса в формате JSON.

Пример использования curl для выполнения GraphQL запроса:

```
curl -H "Authorization: Bearer <access_token>" -H "x-tenant-id: <tenant_id>" -X POST
-d " \
 { \
   \"query\": \"query { <query> }\" \
 } \
" <link_to_graphql>
```

Здесь:

- `<access_token>` — токен доступа к системе.
- `<tenant_id>` — идентификатор тенанта в системе.
- `<query>` — запрос GraphQL, например, на получение списка событий или информации об объекте системы.
- `<link_to_graphql>` — адрес конечной точки для доступа к GraphQL API системы, например, `https://your-rvision-domain/api/graphql`.



Важно корректно экранировать специальные символы в теле JSON-запроса для обеспечения его правильной обработки сервером. Внешние двойные кавычки и экранированные внутренние двойные кавычки помогают избежать ошибок при разборе запроса.

При отсутствии необходимых разрешений у токена для доступа к запрашиваемому ресурсу, API вернет ошибку с указанием, какие разрешения необходимы для выполнения запроса.

Выгрузка событий

Позволяет осуществлять выгрузку событий через подписку `exportSearch`, поддерживая фильтрацию, сортировку и пагинацию.

Структура подписки `exportSearch`

```
subscription {
  exportSearch(
    eventStorageId: String!
    filters: [SearchFilter!]
```

```
    limit: Int
    rql: String!
    timePeriod: TimePeriodInput!
  ) {
    __typename
    searchId
    exportResult {
      ... on ExportSearchLimitWarning {
        limitWarning
        __typename
      }
      ... on ExportSearchResult {
        result {
          ... on SearchResultHeaders {
            names
            types
            __typename
          }
          ... on SearchResultRows {
            rows
            __typename
          }
          __typename
        }
        __typename
      }
      __typename
    }
  }
}
```

Параметры подписки `exportSearch`

Параметр	Тип данных	Обязателен	Описание
`eventStorageId`	Строка	Да	Идентификатор хранилища событий, указывает источник экспортируемых данных.
`filters`	Массив `[SearchFilter!]`	Да	Список фильтров, применяемых к выгрузке.
`limit`	Целое число	Нет	Ограничение на количество возвращаемых записей.
`rql`	Строка	Да	RQL-запрос для фильтрации данных.

Параметр	Тип данных	Обязателен	Описание
`timePeriod`	Объект `TimePeriodInput!`	Да	Временной интервал для выгрузки данных.

Рассмотрим структуру подписки:

- `subscription`: указывает на тип операции подписки в GraphQL, позволяющее получать данные в реальном времени.
- `exportSearch`: имя подписки, используемой для выгрузки данных событий.
- `eventStorageId`: обязательный параметр, принимающий идентификатор хранилища событий, из которого будут экспортированы данные.
- `filters`: массив фильтров, применяемых к выгрузке. Каждый фильтр задается объектом `SearchFilter`, который включает поля `inverted`, `name`, `operation`, и `value`.
- `limit`: ограничение на количество возвращаемых записей. Необязательный параметр.
- `rql`: строка RQL-запроса, используемая для дополнительной фильтрации выгружаемых данных.
- `timePeriod`: временной интервал, за который происходит выгрузка данных. Принимает объект `TimePeriodInput` с полями `from` и `to`, указывающими начало и конец интервала соответственно.

В ответ на подписку возвращаются следующие данные:

- `searchId`: идентификатор созданного поиска, по которому происходит выгрузка данных.
- `exportResult`: объект, содержащий результат выгрузки. Может представлять предупреждение о превышении лимита `ExportSearchLimitWarning` или непосредственно результаты поиска `ExportSearchResult`.
- `ExportSearchLimitWarning` содержит поле `limitWarning`, информирующее о превышении установленного лимита записей.
- `ExportSearchResult` включает в себя `result`, который может быть типом `SearchResultHeaders` с информацией о заголовках и типах данных или `SearchResultRows` с собственно данными строк.

Пример подписки `exportSearch`

```
subscription {
  exportSearch(
    eventStorageId: "34b1b60a-9cce-4a64-b69a-5837a433b730",
    filters: [
      {
        inverted: false,
        name: "collectorId",
        operation: EQ,
        value: "'a7e7a9c3-c69e-460e-967a-1e437601fd71'"
      }
    ],
    limit: 500,
    rql: "",
    timePeriod: {
      from: "2024-02-14T07:19:33.806Z",
```

```
      to: "2024-02-14T07:20:33.806Z"
    }
  ) {
    __typename
    searchId
    exportResult {
      ... on ExportSearchLimitWarning {
        limitWarning
        __typename
      }
      ... on ExportSearchResult {
        result {
          ... on SearchResultHeaders {
            names
            types
            __typename
          }
          ... on SearchResultRows {
            rows
            __typename
          }
          __typename
        }
        __typename
      }
      __typename
    }
  }
}
```

Как работает данная подписка:

- `subscription`: определяет начало подписки, которая будет активно слушать изменения и отправлять данные клиенту, как только они станут доступны.
- `exportSearch`: имя подписки, которое инициирует процесс выгрузки данных событий из системы.
- `eventStorageId`: идентификатор хранилища событий, который указывает, откуда будут экспортироваться данные. В данном случае "34b1b60a-9cce-4a64-b69a-5837a433b730".
- `filters`: список фильтров, применяемых к выгрузке. В примере используется один фильтр для поля "collectorId" с операцией сравнения EQ (равно) и значением "'a7e7a9c3-c69e-460e-967a-1e437601fd71'". Фильтр не инвертирован (`inverted: false`), что означает прямое сравнение.
- `limit`: ограничивает количество возвращаемых записей до 500.
- `rql`: параметр, представляющий RQL-запрос для дополнительной фильтрации данных. В данном случае он пуст, что означает отсутствие дополнительных условий фильтрации.
- `timePeriod`: задает временной интервал для выгрузки данных, начиная с "2024-02-14T07:19:33.806Z" и заканчивая "2024-02-14T07:20:33.806Z".
- `searchId`: возвращаемый идентификатор нового поиска, который может использоваться для отслеживания процесса выгрузки.

- `exportResult`: объект, содержащий результат выгрузки. Используется фрагмент `ExportSearchResult`, который может представлять либо предупреждение о превышении лимита (`ExportSearchLimitWarning`), либо непосредственно результаты поиска (`ExportSearchResult`).

Параметры `ExportSearchResult` и `SearchResult` определяют структуру возвращаемых данных:

- `ExportSearchLimitWarning` содержит `limitWarning`, информирующее о превышении заданного лимита выгрузки.
- `ExportSearchResult` включает `result`, который далее разделяется на `SearchResultHeaders` с информацией о заголовках и типах данных и `SearchResultRows` с данными строк.

`__typename` используется для определения конкретного типа возвращаемого объекта в ответе, что позволяет клиенту корректно обрабатывать различные типы результатов выгрузки.

Сохранение черновика элемента экспертизы

Мутация `saveExpertiseDraft` позволяет сохранить черновик элемента экспертизы в системе.

Структура мутации `saveExpertiseDraft`

```
mutation {
  saveExpertiseDraft(input: SaveExpertiseDraftInput!) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Параметры мутации `saveExpertiseDraft`

Параметр	Тип данных	Обязателен	Описание
`folderId`	Строка	Да	Идентификатор каталога, в который добавляется черновик.
`id`	Строка	Нет	Идентификатор существующего элемента экспертизы для обновления.

Параметр	Тип данных	Обязателен	Описание
sourceCode	Строка	Да	Исходный код элемента экспертизы.
type	ExpertiseType	Да	Тип элемента экспертизы, определяющий его категорию или классификацию.

▼ *Возвращаемый тип* Expertise

Параметр	Описание
id	Уникальный идентификатор элемента экспертизы, генерируемый системой.
sourceId	Идентификатор источника данных для элемента экспертизы.
folderId	Идентификатор каталога, к которому принадлежит объект.
type	Тип элемента экспертизы.
name	Название элемента экспертизы.
description	Описание элемента экспертизы.
draft	Статус черновика объекта.
author	Автор элемента экспертизы.
status	Текущий статус элемента экспертизы.
tags	Массив тегов, ассоциированных с объектом.
version	Версия элемента экспертизы.
severity	Уровень серьезности элемента экспертизы.
createdAt	Дата и время создания объекта.
updatedAt	Дата и время последнего обновления объекта.
__typename	Тип объекта в системе GraphQL.

Пример мутации saveExpertiseDraft

```
mutation {
  saveExpertiseDraft(input: {
    folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
    id: "01c3b9ef-b1fa-4ff9-a465-c89d43cb537b",
    sourceCode: "Вставьте исходный код элемента экспертизы",
    type: "CORRELATION_RULE"
  }) {
    id
    sourceId
    folderId
    type
    name
```

```
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Как работает данная мутация:

- `mutation`: указывает на начало операции мутации.
- `saveExpertiseDraft`: имя мутации, используемой для сохранения черновика элемента экспертизы.
- `input`: параметр, который принимает данные для сохранения черновика. Эти данные упакованы в объект `SaveExpertiseDraftInput`.
- `SaveExpertiseDraftInput`: входной объект мутации, содержащий необходимые данные для создания или обновления черновика элемента экспертизы. Включает в себя поля, такие как `folderId`, `id`, `sourceCode`, и `type`.
- `folderId`: идентификатор каталога, в который добавляется черновик. Этот параметр обязателен.
- `id`: необязательный параметр, указывающий идентификатор существующего элемента экспертизы, если требуется обновление черновика.
- `sourceCode`: текстовое содержание кода элемента экспертизы. Обязательный параметр.
- `type`: тип элемента экспертизы, определяющий его функциональную принадлежность или категорию. Этот параметр обязателен.

Возвращаемые поля мутации `saveExpertiseDraft`:

- `id`: уникальный идентификатор созданного или обновленного элемента экспертизы.
- `sourceId`, `folderId`, `type`, `name`, `description`: основные атрибуты элемента экспертизы, включая его исходный идентификатор, принадлежность к каталогу, тип, название, и описание.
- `draft`: статус черновика объекта, указывающий, находится ли объект в разработке.
- `author`: автор элемента экспертизы.
- `status`, `tags`, `version`, `severity`: дополнительные атрибуты объекта, такие как его статус, теги, версия и уровень серьезности.
- `createdAt`, `updatedAt`: даты создания и последнего обновления элемента экспертизы.
- `__typename`: служебное поле GraphQL, указывающее тип возвращаемого объекта.

Публикация версии элемента экспертизы

Мутация `publishExpertise` позволяет публиковать версии элементов экспертизы, переводя их из

статуса "Черновик" в статус "Опубликовано". Опубликованные объекты становятся доступными для просмотра и использования в рамках системы.

Структура мутации `publishExpertise`

```
mutation {
  publishExpertise(input: PublishExpertiseInput!) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Параметры мутации `publishExpertise`

Параметр	Тип данных	Обязателен	Описание
`folderId`	Строка	Да	Идентификатор каталога, к которому будет принадлежать опубликованный объект.
`id`	Строка	Нет	Идентификатор существующего черновика элемента экспертизы.
`sourceCode`	Строка	Да	Исходный код элемента экспертизы для публикации.
`type`	`ExpertiseType`	Да	Тип элемента экспертизы, указывающий на его функциональную категорию.

▼ *Возвращаемый тип* `Expertise`

Параметр	Описание
`id`	Уникальный идентификатор опубликованного элемента экспертизы.
`sourceId`	Идентификатор источника данных для элемента экспертизы.
`folderId`	Идентификатор каталога, к которому принадлежит опубликованный объект.
`type`	Тип опубликованного элемента экспертизы.
`name`	Название опубликованного элемента экспертизы.
`description`	Описание опубликованного элемента экспертизы.
`draft`	Статус черновика, указывающий на стадию разработки объекта.
`author`	Автор элемента экспертизы.
`status`	Текущий статус опубликованного элемента экспертизы.
`tags`	Массив тегов, ассоциированных с элементом экспертизы.
`version`	Версия элемента экспертизы.
`severity`	Уровень серьезности элемента экспертизы.
`createdAt`	Дата и время создания элемента экспертизы.
`updatedAt`	Дата и время последнего обновления элемента экспертизы.
`__typename`	Тип объекта в системе GraphQL.

Пример мутации `publishExpertise`

```
mutation {
  publishExpertise(input: {
    folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
    id: "01c3b9ef-b1fa-4ff9-a465-c89d43cb537b",
    sourceCode: "Вставьте исходный код элемента экспертизы",
    type: CORRELATION_RULE
  }) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
```

```
}
```

Как работает данная мутация:

- `mutation`: указывает на начало операции мутации.
- `publishExpertise`: имя мутации для публикации элемента экспертизы.
- `input`: параметр, принимающий данные для публикации черновика. Включает в себя `folderId`, `id` (если это обновление существующего объекта), `sourceCode`, и `type`.
- `folderId`: идентификатор каталога для нового элемента экспертизы.
- `id`: необязательный параметр, указывающий на существующий элемент экспертизы.
- `sourceCode`: исходный код элемента экспертизы.
- `type`: тип элемента экспертизы, определяющий его функциональную категорию.

Возвращаемые поля мутации `publishExpertise`:

- `id`: уникальный идентификатор опубликованного элемента экспертизы.
- `sourceId`, `folderId`, `type`, `name`, `description`: основные атрибуты элемента экспертизы, включая его исходный идентификатор, принадлежность к каталогу, тип, название и описание.
- `draft`: статус объекта.
- `author`: автор элемента экспертизы.
- `status`, `tags`, `version`, `severity`: дополнительные атрибуты объекта, такие как его статус, теги, версия и уровень серьезности.
- `createdAt`, `updatedAt`: даты создания и последнего обновления элемента экспертизы.
- `__typename`: служебное поле GraphQL, указывающее тип возвращаемого объекта.

Экспорт элементов экспертизы

Запрос `exportExpertise` предоставляет возможность экспортировать элементы экспертизы, определяемые их уникальными идентификаторами (UUID элемента экспертизы), позволяя пользователям системы сохранить эти данные вне системы.

Структура запроса `exportExpertise`

```
exportExpertise(ids: [String!]!) {
    content
    contentType
  }
}
```

Параметры запроса `exportExpertise`

Параметр	Тип данных	Обязателен	Описание
`ids`	Массив `[ID!]`	Да	Массив значений UUID элементов экспертизы для экспорта.

▼ *Возвращаемый тип* `ExportExpertiseResponse`

Параметр	Описание
`content`	Содержимое экспортированных элементов экспертизы в формате base64. Клиент должен декодировать это содержимое и сохранить как файл в формате ZIP.
`contentType`	Тип содержимого, обычно указывается как `application/zip`, обозначает формат возвращаемого файла экспорта.

Пример запроса `exportExpertise`

```
query {
  exportExpertise(ids: ["uuid1", "uuid2"]) {
    content
    contentType
  }
}
```

Этот запрос инициирует экспорт элементов экспертизы с указанными UUID. В ответе предоставляется закодированное содержимое экспортированных объектов в формате base64 и тип содержимого, что позволяет клиенту обработать и сохранить данные в виде ZIP-файла.

Как работает данный запрос:

- `query`: указывает на начало операции запроса.
- `exportExpertise`: имя запроса, используемого для экспорта элементов экспертизы.
- `ids`: параметр запроса, представляющий собой массив UUID элементов экспертизы, которые будут экспортированы.

Возвращаемые поля мутации `exportExpertise`:

- `content`: содержимое экспортированных объектов в формате base64, позволяет пользователям декодировать и сохранить информацию в виде файла.
- `contentType`: тип содержимого экспорта, указывающий на формат файла (например, `application/zip` для ZIP-архивов).

Просмотр каталогов

Запрос `resourceFolders` предлагает обзор всех каталогов, содержащих элементы экспертизы в системе.

Структура запроса `resourceFolders`

```
query {
  resourceFolders {
    id
    name
    parentFolderId
    createdAt
```

```
    updatedAt
    creator {
      id
      name
    }
    updater {
      id
      name
    }
  }
}
```

▼ *Возвращаемый тип* `ResourceFolder`

Параметр	Описание
`id`	Уникальный идентификатор каталога.
`name`	Название каталога.
`parentFolderId`	Идентификатор родительского каталога.
`createdAt`	Дата и время создания каталога.
`updatedAt`	Дата и время последнего обновления каталога.
`creator`	Информация о пользователе, создавшем каталог.
`updater`	Информация о последнем пользователе, обновившем каталог.

Пример запроса на получение списка каталогов

```
query {
  resourceFolders {
    id
    name
    parentFolderId
    createdAt
    updatedAt
    creator {
      name
    }
    updater {
      name
    }
  }
}
```

Данный запрос возвращает иерархический список всех каталогов, доступных в системе, включая информацию об их создании и последних изменениях.

Просмотр элемента экспертизы

Запрос `expertise` предоставляет детальную информацию о конкретном элементе экспертизы,

идентифицируемом по его уникальному идентификатору в системе.

Структура запроса `expertise`

```
query {
  expertise(id: ID!) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Параметры запроса `expertise`

Параметр	Тип данных	Обязателен	Описание
`id`	ID!	Да	Уникальный идентификатор элемента экспертизы для получения информации.

▼ *Возвращаемый тип* `Expertise`

Параметр	Описание
`id`	Уникальный идентификатор элемента экспертизы.
`sourceId`	Идентификатор исходного объекта, если применимо.
`folderId`	Идентификатор каталога, содержащего элемент экспертизы.
`type`	Тип элемента экспертизы, указывающий на его функциональную категорию.
`name`	Название элемента экспертизы.
`description`	Описание объекта, предоставляющее детальную информацию о его содержании.
`draft`	Статус черновика, указывающий на текущее состояние объекта.
`author`	Автор элемента экспертизы.
`status`	Текущий статус объекта в системе.

Параметр	Описание
`tags`	Массив тегов, ассоциированных с объектом.
`version`	Версия элемента экспертизы, отражающая его историю изменений.
`severity`	Уровень серьезности объекта, определяющий его важность.
`createdAt`	Дата и время создания объекта.
`updatedAt`	Дата и время последнего обновления объекта.
`__typename`	Служебное поле GraphQL, указывающее тип возвращаемого объекта.

Просмотр списка элементов экспертизы

Запрос `expertises` обеспечивает возможность получения списка элементов экспертизы в системе, поддерживая фильтрацию, сортировку и пагинацию для удобства навигации и анализа.

Структура запроса `expertises`

```
query {
  expertises(
    filter: ExpertisesFilterInput,
    pagination: PaginationInput,
    sorting: SortingInput
  ) {
    expertises {
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
```

```
  }
}
```

Параметры запроса `expertises`

Параметр	Тип данных	Обязателен	Описание
`filter`	`ExpertisesFilterInput`	Нет	Параметры фильтрации для настройки списка элементов экспертизы.
`pagination`	`PaginationInput`	Нет	Параметры пагинации для контроля количества отображаемых результатов.
`sorting`	`SortingInput`	Нет	Параметры сортировки для упорядочивания результатов запроса.

▼ *Возвращаемый тип* `Expertises`

Параметр	Описание
`expertises`	Список элементов экспертизы, соответствующих заданным критериям фильтрации, сортировки и пагинации.
`pageInfo`	Метаинформация о странице, включая общее количество доступных объектов и параметры текущей пагинации.
`totalCount`	Общее количество элементов экспертизы, соответствующих критериям фильтрации.
`__typename`	Служебное поле GraphQL, указывающее тип возвращаемого объекта.

Пример запроса на получение списка элементов экспертизы

```
query {
  expertises(
    filter: {
      folder: {
        folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
        withChildFolders: true
      },
      query: "",
      status: ENABLED,
      type: CORRELATION_RULE
    },
    pagination: {
      limit: 100,
      offset: 0
    },
    sorting: {
      order: DESC,
```

```
      orderBy: CREATED_AT
    }
  ) {
    expertises {
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
  }
}
```

Этот запрос инициирует вывод списка элементов экспертизы, применяя заданные критерии фильтрации, пагинации и сортировки.

Как работает данный запрос:

- `query`: указывает на начало операции запроса.
- `expertises`: имя запроса, используемое для получения списка элементов экспертизы.
- `filter`: параметр, принимающий критерии фильтрации для применения к запросу. Включает:
  - `folder`: объект с `folderId`, указывающий на конкретный каталог, и `withChildFolders`, указывающий, следует ли включать объекты из дочерних каталогов.
  - `query`: строка для поиска по элементам экспертизы, в данном примере пустая, что означает отсутствие фильтрации по текстовому запросу.
  - `status`: фильтр по статусу объектов, в данном случае `ENABLED`, что указывает на выборку активных объектов.
  - `type`: фильтр по типу объекта, в данном случае `CORRELATION_RULE`, ограничивающий выборку

объектами определенного типа.

- `pagination`: объект с параметрами пагинации, `limit` устанавливает максимальное количество объектов в ответе, `offset` — смещение от начала списка.
- `sorting`: параметры сортировки результатов запроса, в данном случае сортировка по дате создания (`CREATED_AT`) в обратном порядке (`DESC`).

Возвращаемые данные запроса `expertises`:

- `expertises`: массив элементов экспертизы, каждый из которых содержит:
  - `id`, `sourceId`, `folderId`, `type`, `name`, `description`, `draft`, `author`, `status`, `tags`, `version`, `severity`, `createdAt`, `updatedAt`: основные атрибуты элемента экспертизы.
  - `__typename`: служебное поле GraphQL, указывающее тип возвращаемого объекта.
- `pageInfo`: объект с информацией о пагинации и общим количеством объектов в ответе, включает:
  - `pagination`: внутренний объект с текущими параметрами пагинации (`limit`, `offset`).
  - `totalCount`: общее количество доступных элементов экспертизы с учетом примененных фильтров.
  - `__typename`: служебное поле для объекта пагинации и общего количества.

Удаление элемента экспертизы

Мутация `deleteExpertise` позволяет удалить элемент экспертизы из системы.

```
deleteExpertise(input: DeleteExpertiseInput!): DeleteExpertiseResponse!
```

Параметр	Тип данных	Обязателен	Описание
`input`	`DeleteExpertiseInput`	Да	Входные данные для удаления, включая `id` удаляемого элемента экспертизы.

Параметры мутации `DeleteExpertiseInput`

Параметр	Тип данных	Обязателен	Описание
`id`	Строка	Да	Идентификатор элемента экспертизы, который требуется удалить.

Возвращаемый тип `DeleteExpertiseResponse`

Параметр	Тип данных	Обязателен	Описание
`id`	Строка	Да	Идентификатор удаленного элемента экспертизы, подтверждающий успешное выполнение операции.

Пример использования мутации для удаления экспертизы:

```
mutation {
  deleteExpertise(input: { id: "uniqueExpertiseId" }) {
    id
  }
}
```

Этот запрос инициирует удаление элемента экспертизы с указанным идентификатором из системы. В ответе возвращается идентификатор удаленного элемента экспертизы.

Фильтры

GraphQL API системы поддерживает использование фильтров для уточненного поиска и отбора данных, возвращаемых при выполнении запросов или подписок. Они могут быть применены к различным полям и поддерживают разнообразные операции сравнения.

Использование фильтров

Для применения фильтров в запросах или подписках, используется параметр `filters`. Каждый фильтр определяет `name` поля для фильтрации, `operation` (тип операции), `value` (значение для сравнения) и, опционально, `inverted` (инвертирование условия).

Пример запроса с использованием фильтров

```
query {
  expertises(
    filter: {
      folder: {
        folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
        withChildFolders: true
      },
      query: "",
      status: ENABLED
      type: CORRELATION_RULE
    },
    pagination: {
      limit: 100,
      offset: 0
    },
    sorting: {
```

```
      order: DESC,
      orderBy: CREATED_AT
    }
    ) {
    expertises {
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
  }
}
```

Операции сравнения

В фильтрах используются следующие операции сравнения:

- `BETWEEN` — между двумя значениями;
- `EQ` — равно;
- `GE` — больше или равно;
- `GT` — больше;
- `IN` — входит в перечень;
- `LE` — меньше или равно;
- `LIKE` — соответствует шаблону;
- `LT` — меньше;
- `NE` — не равно.



Набор доступных операторов может варьироваться в зависимости от запроса.

Параметры фильтра

Параметры фильтра

Параметр	Тип	Описание
`inverted`	Логический	Определяет, будет ли условие инвертировано.
`name`	Строка	Имя поля, по которому применяется фильтр.
`operation`	Перечисление	Операция сравнения, используемая в фильтре.
`value`	Строка	Значение для сравнения с полем.

Сортировка и пагинация

В GraphQL API системы предусмотрены механизмы сортировки и пагинации, позволяющие управлять порядком представления данных и их разбивкой на удобные для восприятия части. Эти механизмы особенно важны при работе с запросами, возвращающими большое количество записей, например, `expertises`.

Сортировка

Сортировка в запросах GraphQL API системы позволяет упорядочить результаты запроса по определенным критериям. В запросе `expertises` параметр `sorting` используется для указания направления сортировки (`order`) и поля, по которому следует сортировать (`orderBy`).

Пример ниже демонстрирует, как можно запросить список экспертиз, отсортированный по дате создания (`CREATED_AT`) в убывающем порядке (`DESC`), с применением фильтра и пагинации:

Пример запроса с сортировкой

```
query {
  expertises(
    filter: {
      folder: {
        folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
        withChildFolders: true
      },
      query: "",
      status: ENABLED,
      type: CORRELATION_RULE
    },
    pagination: {
      limit: 100,
      offset: 0
    },
```

```
    sorting: {
      order: DESC,
      orderBy: CREATED_AT
    }
  ) {
    expertises {
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
  }
}
```

Параметры сортировки

Параметр	Тип	Описание
`order`	Перечисление	Направление сортировки, может быть `ASC` (по возрастанию) или `DESC` (по убыванию).
`orderBy`	Строка	Поле, по которому будет проводиться сортировка.

Пагинация

Пагинация в GraphQL API системы предоставляет механизм для контроля объема данных, возвращаемых в результате запроса при работе с большими наборами данных, позволяя разбивать их на управляемые "страницы".

Пример запроса ниже иллюстрирует, как получить список элементов экспертизы с использованием пагинации для ограничения количества возвращаемых записей и управления точкой начала выборки данных:

Пример запроса с пагинацией

```
query {
  expertises(
    filter: {
      folder: {
        folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
        withChildFolders: true
      },
      query: "",
      status: ENABLED,
      type: CORRELATION_RULE
    },
    pagination: {
      limit: 100,
      offset: 0
    },
    sorting: {
      order: DESC,
      orderBy: CREATED_AT
    }
    ) {
    expertises {
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
```

```
    }
    __typename
  }
}
```

Параметры пагинации

Параметр	Тип	Описание
`limit`	Целое число	Максимальное количество записей, возвращаемых за один запрос.
`offset`	Целое число	Смещение от начала списка записей, с которого начинается возвращение данных.

Обработка ошибок API

В процессе взаимодействия с API системы могут возникать различные ошибки. Ниже приведен список возможных ошибок с описаниями и рекомендациями по их устранению.

Код ошибки	Описание
AUTH.ISSUE_TOKEN.TOKEN_ALREADY_EXIST	Попытка создать токен для пользователя, для которого ранее уже был создан токен. Убедитесь, что предыдущий токен не активен или используйте его для работы с API.
ALERT.CHANGE_STATUS.CURRENT_STATUS	Попытка изменения статуса оповещения на текущий статус. Проверьте текущий статус оповещения и убедитесь, что изменение статуса необходимо.
ALERT.CHANGE_STATUS.REOPENING	Попытка изменения статуса оповещения на "Новое". Повторное открытие оповещения недопустимо.
COLLECTOR.DELETE.R_SPACE_ERROR	Произошла ошибка при удалении коллектора. Обратитесь в службу поддержки для устранения проблемы.
COLLECTOR.DISABLE.ALREADY_DISABLED	Попытка отключения уже отключенного коллектора. Проверьте текущее состояние коллектора перед его отключением.
COLLECTOR.ENABLE.ALREADY_ENABLED	Попытка включения уже включенного коллектора. Убедитесь, что коллектор отключен.
COLLECTOR.IMPORT.INCORRECT_DATA	Данные импортируемых коллекторов некорректны. Проверьте корректность данных коллекторов перед их импортом.

Код ошибки	Описание
COLLECTOR_ENRICHMENT_TABLE.CREATE.ALREADY_CREATED	Попытка добавления таблицы обогащения в коллектор, в который уже была добавлена эта таблица обогащения. Убедитесь, что таблица обогащения еще не добавлена в коллектор
DASHBOARD.IMPORT.INCORRECT_DATA	Данные импортируемых дашбордов некорректны. Проверьте корректность данных дашбордов перед их импортом.
EVENT_MODEL.IMPORT.INCORRECT_DATA	Данные импортируемых моделей событий некорректны. Убедитесь в правильности данных моделей событий перед импортом.
EVENT_SOURCE.DISABLE.ALREADY_DISABLED	Попытка отключения уже отключенного конвейера. Проверьте статус конвейера перед его отключением.
EVENT_SOURCE.DISABLE.COLLECTOR_DISABLED	Попытка отключения конвейера с отключенным связанным коллектором. Включите связанный коллектор перед отключением конвейера.
EVENT_SOURCE.ENABLE.ALREADY_ENABLED	Попытка включения уже включенного конвейера. Перепроверьте статус конвейера.
EVENT_SOURCE.ENABLE.COLLECTOR_DISABLED	Попытка включения конвейера с отключенным связанным коллектором. Включите коллектор перед включением конвейера.
EVENT_SOURCE.ENABLE.PIPELINE_ERRORS	Попытка включения конвейера с ошибками валидации. Исправьте все ошибки валидации перед включением конвейера.
EVENT_SOURCE.IMPORT.INCORRECT_DATA	Данные импортируемых конвейеров некорректны. Проверьте корректность данных конвейеров перед их импортом.
EVENT_SOURCE.UPDATE.DISABLE_TO_UPDATE	Попытка выполнить операцию для включенного конвейера. Отключите конвейер перед выполнением операции обновления.
EXPERTISE.DELETE.IN_USE	Попытка удаления элемента экспертизы, который используется в одном из конвейеров. Убедитесь, что элемент экспертизы не используется ни на одном из конвейеров перед его удалением.
EXPERTISE.ENABLE.NOT_PUBLISHED_YET	Опубликуйте элемент экспертизы, чтобы включить его. Убедитесь, что объект был опубликован перед его включением.
EXPERTISE.PUBLISH.DUPLICATED_ENTITY	Экспертиза с таким ID или с такой версией уже существует. Проверьте и измените ID и/или версию экспертизы перед публикацией.

Код ошибки	Описание
EXPERTISE.PUBLISH.EMPTY_CONTENT	Не указано содержимое элемента экспертизы. Укажите содержимое элемента экспертизы перед публикацией.
EXPERTISE.PUBLISH.EMPTY_SCHEMA	Не указана схема элемента экспертизы. Укажите схему элемента экспертизы перед публикацией.
EXPERTISE.PUBLISH.EMPTY_URL	Указана пустая ссылка на файл элемента экспертизы. Укажите корректную ссылку на файл элемента экспертизы.
EXPERTISE.PUBLISH.FAIL_DOWNLOAD	Произошла ошибка при загрузке содержимого таблицы обогащения по указанной ссылке. Проверьте доступность и корректность ссылки.
EXPERTISE.PUBLISH.INVALID_SOURCE_CODE	Ошибка в коде элемента экспертизы. Исправьте ошибки в коде элемента экспертизы перед публикацией.
EXPERTISE.PUBLISH.INVALID_VERSION	Укажите корректную версию элемента экспертизы. Убедитесь, что в поле версии указана корректная версия элемента экспертизы.
EXPERTISE.PUBLISH.LOWER_VERSION	Укажите версию больше текущей. Проверьте версию элемента экспертизы и укажите значение больше текущего.
EXPERTISE.PUBLISH.TESTS_FAILED	Во время выполнения тестов произошла ошибка. Исправьте ошибки, выявленные в ходе тестирования, перед публикацией элемента экспертизы.
EXPERTISE.PUBLISH.TRY_TO_CHANGE_ID	Невозможно изменить ID элемента экспертизы. Не пытайтесь изменять ID опубликованного элемента экспертизы.
EXPERTISE.PUBLISH.UNSUPPORTED_LOCATION	Указанный тип содержимого элемента экспертизы не поддерживается. Используйте поддерживаемые типы содержимого для элемента экспертизы.
EXPERTISE.RUN_TESTS.INVALID_SOURCE_CODE	Для запуска тестов необходимо исправить все ошибки в коде. Перед тестированием убедитесь в отсутствии ошибок в коде элемента экспертизы.
RESOURCE_FOLDER.DELETE.CONTAINS_RESOURCES	Попытка удаления папки, содержащей ресурсы. Удалите все ресурсы из папки перед ее удалением.
RESOURCE_FOLDER.DELETE.ROOT_FOLDER	Попытка удаления корневой папки. Удаление корневой папки невозможно.

Код ошибки	Описание
RESOURCE_FOLDER.PATCH.PARENT_FOLDER_AS_CHILD	Попытка присвоения родительской директории ID дочерней. Проверьте корректность структуры директорий.
RESOURCE_FOLDER.PATCH.PARENT_FOLDER_AS_FOLDER	Попытка присвоения родительской директории собственного ID. Убедитесь в корректности изменений в структуре директорий.
RESOURCE_FOLDER.PATCH.PARENT_FOLDER_NOT_EXIST	Попытка присвоения несуществующего ID в качестве родительской директории. Убедитесь в существовании директории с таким ID.
RESOURCE_FOLDER.PATCH.ROOT_FOLDER_CREATING	Попытка создания второго корневого каталога. Создание более одного корневого каталога невозможно.
RESOURCE_FOLDER.PATCH.ROOT_FOLDER_EDITING	Попытка редактирования корневого каталога. Изменение корневого каталога невозможно.

Устранение проблем

В данном разделе описаны возможные проблемы при работе с системой и приведены способы их решения.

- Устранение проблем при установке или обновлении системы
- Устранение проблем при установке или обновлении агентов
- Устранение проблем при эксплуатации системы

Устранение проблем при установке или обновлении системы

В данном разделе приведены возможные проблемы при установке или обновлении системы и способы их решения.

Установка Kubernetes с помощью Kubespray: несовместимые версии пакетов

Проявление: во время работы скрипта установки Kubernetes с помощью Kubespray возникают ошибки несовместимости системных пакетов и тех, что поставляются вместе с установщиком, например:

```
cannot install both python3-3.8.19-4.el7.x86_64 from deps-offline and python3-3.8.20-4.el7.x86_64 from @System
```

Причина: несовместимость версий пакетов, установленных в операционной системе узла кластера, с пакетами, необходимыми для работы установщика.

Решение:

1. На каждом из узлов кластера удалите конфликтующие пакеты, выполнив следующую команду для каждого из них:

   ```
   sudo dnf remove <package_name>
   ```

   Здесь:

   - `<package_name>` — имя пакета, установленного в системе (`@System`). В примере выше — `python3-3.8.20-4.el7.x86_64`.

   > В зависимости от операционной системы на узле кластера может использоваться другой пакетный менеджер вместо `dnf`, например, `rpm`.

2. Повторно запустите скрипт установки Kubernetes на машине, с которой запускали его ранее.

Longhorn: ошибка конфигурации

Проявление: при установке или обновлении системы возникает ошибка `Unable to attach or mount volumes: unmounted volumes=[data], unattached volumes=[kube-api-access scripts data config tz]: timed out waiting for the condition`. Установлен провайдер хранилища Longhorn.

Причина: некорректные настройки Longhorn или на узлах кластера не запущена служба `iscsid`.

Решение:

1. Проверьте настройки Longhorn. Для этого выполните следующие действия:
   a. Осуществите проброс портов. Для этого выполните на хосте с сетевым доступом к master-узлу кластера следующую команду:

      ```
      kubectl -n longhorn-system port-forward service/longhorn-frontend
      <local_port>:80
      ```

      Здесь:

      - `<local_port>` — порт на хосте, по которому будет осуществляться доступ к Longhorn.

   b. В браузере откройте URL `localhost:<local_port>`, где `<local_port>` — порт с предыдущего шага. Отобразится панель управления Longhorn.
   c. Чтобы просмотреть и отредактировать конфигурацию узлов (реплик) Longhorn, перейдите в раздел **Node**.
   d. Если количество узлов Longhorn больше, чем количество worker-узлов кластера, удалите лишние узлы. Для этого нажмите на кнопку **Remove Node** в выпадающем меню в столбце **Operation** для каждого из лишних узлов.
   e. Для каждого узла Longhorn проверьте, не превышает ли объем выделенного ему пространства размер диска узла кластера:
      i. В выпадающем меню в столбце **Operation** нажмите на кнопку **Edit node and disks**.
      ii. Если в поле **Storage Reserved** выделено больше пространства, чем размер диска узла кластера, уменьшите размер пространства и нажмите на кнопку **Save**.

      Рекомендуется выделять под Longhorn 30% диска узла кластера.

2. Если исправление конфигурации Longhorn не решило проблему, проверьте, запущена ли служба `iscsid` на всех узлах кластера:
   a. На каждом узле кластера выполните команду:

      ```
      systemctl status iscsid
      ```

   b. Если статус службы отличается от `active (running)`, запустите ее:

      ```
      systemctl start iscsid
      ```

Ошибка `cannot assign requested address` при установке helm-чартов

Проявление: установка helm-чартов завершается с ошибкой, содержащей сообщение `cannot assign requested address`, например:

```
Error: create: failed to create: Internal error occurred: Get
"https://[::1]:6443/api/v1/namespaces/local-path-provisioner": dial tcp [::1]:6443:
connect: cannot assign requested address
```

Причина: при развертывании кластера Kubernetes на одном или нескольких узлах не было отключено использование IPv6 или в кэше Kubernetes сохранились некорректные настройки.

Решение: убедитесь, что использование IPv6 отключено на всех узлах кластера, перезапустите узлы кластера и установщик системы. Для этого:

1. На каждом узле кластера выполните следующие действия:
    a. Откройте файл `/etc/sysctl.d/99-disable-ipv6.conf`.
    b. Проверьте, добавлены ли в файл следующие строки:

    ```
    net.ipv6.conf.all.disable_ipv6 = 1
    net.ipv6.conf.default.disable_ipv6 = 1
    net.ipv6.conf.lo.disable_ipv6 = 1
    ```

    c. Если строки отсутствуют:
        i. Добавьте строки в файл и сохраните его.
        ii. Примените изменения с помощью следующей команды:

        ```
        sudo sysctl --system
        ```

    d. Перезагрузите узел кластера.
2. Перезапустите установщик системы.

MetalLB: ошибка запуска в окружении Альт Сервер

Проявление: отсутствует доступ к веб-интерфейсу системы. На узлах кластера установлена ОС Альт Сервер или Альт СП Сервер. Под `metallb-controller` в пространстве имен `metallb-system` не запущен.

Причина: ошибка конфигурации MetalLB.

Решение: внесите правку в конфигурацию MetalLB, выполнив следующую команду на master-узле кластера:

```
ubectl apply -f - <<EOF
apiVersion: metallb.io/v1beta1
```

```
kind: IPAddressPool
metadata:
  name: evo-pool
  namespace: metallb-system
spec:
  addresses:
  - <system_ip>/32
  autoAssign: true
  avoidBuggyIPs: false
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: evo-nlb
  namespace: metallb-system
EOF
```

Здесь:

- `<system_ip>` — IP-адрес для обращения к системе, зарезервированный на этапе подготовки к установке системы.

Устранение проблем при установке или обновлении агентов

В данном разделе приведены возможные проблемы при установке или обновлении агента и способы их решения.

Некорректный статус модуля `nats-server`

Проблема: после установки и запуска агента статус модуля `nats-server` отличается от `running`, например:

```
[{"name":"nats-server","status":"running"},...]
```

Решение: переустановите агент. Для этого:

1. Удалите агент.
2. Перезапустите устройство.
3. Заново установите и запустите агент.

Установленный агент не отображается в менеджере агентов

Проблема: после установки и запуска агент не отображается в менеджере агентов.

Решение:

1. Если система отображает ошибку вида `Error trying to connect as leafnode to remote server`:
   a. Проверьте подключение компьютера к сети. Если подключение отсутствует, восстановите его.
   b. Проверьте доступность шлюза, указанного в конфигурационном файле `rpoint.conf`. Если указаны некорректные настройки подключения (например, заданы закрытые порты), исправьте их.
2. Откройте файл `logs/nats.log` и найдите последнее сообщение.
   - Если сообщение содержит запись вида `10.99.66.43:32422 - lid:29 - JetStream using domains: local "agent-19395b65-6bdd-5a00-9970-47668a6ce92c"`, подключение к NATS прошло успешно. Вероятнее всего, проблема в конфигурации пространства центрального кластера или кластера сателлита.

     Для решения проблемы удостоверьтесь в корректности заданных настроек подключения к пространству:

     - Если настройки некорректны, исправьте их.
     - Если настройки корректны, но ошибка сохраняется, обратитесь в службу поддержки по адресу support@rvision.ru.
   - Если сообщение содержит запись вида `Error trying to connect as leafnode to remote server "ep-worker03.rpoint.test.dev.rvision.local:32422" (attempt 1): dial tcp 10.99.66.43:32422: connect: connection refused`, указанный в сообщении порт недоступен на оборудовании.

     Для решения проблемы обратитесь к системному администратору с запросом на открытие порта.
3. Если на предыдущих шагах проблемы не были выявлены и подключение агента прошло успешно, возможно, IP-адреса, к которым должны подключаться агенты, не указаны в файле `etc/nats-leafnode.conf`.

   Для решения проблемы:

   a. Откройте файл конфигурации агента `rpoint.conf`.
   b. Проверьте, правильно ли указаны в файле IP-адреса, к которым должны подключаться агенты.

      Если IP-адреса указаны неправильно, укажите верные IP-адреса в файле `rpoint.conf`.

      Если IP-адреса указаны правильно, но данные тенантов и шлюзов не отображаются в файле `nats-leafnode.conf`, проверьте корректность этих данных в менеджере агентов:

      - Если данные тенантов и шлюзов некорректны, исправьте их.
      - Если данные корректны, ошибка, вероятнее всего, связана с самим агентом. Обратитесь в службу поддержки по адресу support@rvision.ru.

Агенты на Windows-хостах не отображаются в менеджере агентов

Проблема: агент, установленный на хосте с ОС Windows, не отображается в менеджере агентов. В файле логов `rpoint.log` агента возникают ошибки, содержащие сообщение `maximum payload exceeded`.

Причина: агент отправил в систему сообщение, размер которого превысил максимальный размер сообщения NATS.

Решение: увеличьте значение параметра `max_payload` для агента. Для этого:

1. Добавьте в секцию `nats_server` файла конфигурации агента `rpoint.conf` параметр `max_payload`:

   ```
   "max_payload": <new_max_payload>
   ```

   Здесь:

   - `<new_max_payload>` — новый максимальный размер сообщения NATS в байтах, например, `32000000`.

2. Перезапустите агент.

Старый агент заменяется новым

Проблема: при установке нового агента старый пропадает из списка раздела **Агенты** → **Агенты**.

Причина: старый и новый агент имеют одинаковый идентификатор. При развертывании нового агента произошла перезапись старого.

Решение: измените идентификатор нового агента одним из следующих способов:

- Ручное изменение текущего идентификатора:
  1. Измените идентификатор агента в файле `.agent_id` в подкаталоге `etc` каталога установки агента.

     Пути хранения файла `.agent_id` по умолчанию:

     - для ОС Linux: `/opt/rvision/agent/etc/`;
     - для ОС Windows: `C:\Program Files (x86)\RVision\Agent\etc\`;
     - для macOS: `/Library/R-Vision/agent/etc/`.

  2. Запустите агент.
- Изменение настроек генерации идентификатора:
  1. Остановите агент.
  2. Удалите файлы `.agent_id` и `nats-leafnode.conf` в подкаталоге `etc` каталога установки агента.
  3. Задайте новые настройки генерации идентификатора агента в файле конфигурации `rpoint.conf`.

4. Запустите агент. Для агента будет сгенерирован новый идентификатор в соответствии с заданными настройками.

Старый агент отобразится в списке раздела **Агенты** → **Агенты**.

Сбой при обновлении агента

Проблема: во время обновления агента произошел сбой.

Решение:

1. Удалите агент.
2. Установите агент заново.

Установленный агент не отображается в списке агентов

Проблема: установленный агент не отображается в списке раздела **Агенты** → **Агенты**.

Причина: сообщение о регистрации, отправляемое агентом, превышает максимальный размер сообщения NATS, задаваемый параметром конфигурации `max_payload`. По умолчанию он равен 4 МБ (4 194 304 байт).

Решение: увеличьте значение параметра `max_payload`.



Максимальное допустимое значение `max_payload` — 64 МБ. Однако не рекомендуется увеличивать его более чем до 8 МБ для поддержания оптимальной производительности системы и контролируемого потребления вычислительных ресурсов.



Увеличить значение параметра `max_payload` следует в центральном кластере и во всех кластерах сателлитов.

Чтобы увеличить значение `max_payload`:

1. Добавьте параметр `max_payload` или отредактируйте его значение в следующих объектах ConfigMap:
   - в центральном кластере — `nats-main-config`, `nats-bridge-config` и `nats-space-config`;
   - в кластере сателлита — только `nats-space-config`.

   Для этого выполните следующие действия:

   a. Откройте ConfigMap, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit configmap <configmap>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

- `<configmap>` — имя ConfigMap: `nats-main-config`, `nats-bridge-config` или `nats-space-config`.

b. По пути `data.nats.conf` добавьте ключ `max_payload` или отредактируйте его значение, если он уже существует. В качестве значения этого ключа укажите необходимый максимальный размер события в мегабайтах.

▼ *Пример ConfigMap с ключом* `max_payload`

```
data:
  nats.conf: |
    {
      "http_port": 8222,
      "jetstream": {
        "domain": "main",
        "max_file_store": 30Gi,
        "max_memory_store": 0,
        "store_dir": "/data"
      },
      "lame_duck_duration": "30s",
      "lame_duck_grace_period": "10s",
      "leafnodes": {
        "no_advertise": true,
        "port": 7422
      },
      # Добавьте или отредактируйте данный параметр.
      "max_payload": 8Mb,
      "pid_file": "/var/run/nats/nats.pid",
      "port": 4222,
      "server_name": $SERVER_NAME
    }
```

2. Подождите несколько минут. Когда Kubernetes применит новую конфигурацию, проблема будет устранена.



Если новая конфигурация не применяется в течение длительного времени, выполните следующие действия:

1. Проверьте корректность ConfigMap NATS, выполнив следующую команду для `nats-main`, `nats-bridge` и `nats-space` (в случае центрального кластера) или только `nats-space` (в случае кластера сателлита):

```
kubectl exec -n <namespace> <nats_name>-0 -c nats -- nats-server
--config /etc/nats-config/nats.conf -t
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

- `<nats_name>` — имя экземпляра NATS: `nats-main`, `nats-bridge` или `nats-space`.

Если ConfigMap валидный, отобразится сообщение, содержащее строку `configuration file /etc/nats-config/nats.conf is valid`, например:

```
nats-server: configuration file /etc/nats-config/nats.conf is
valid
(sha256:0f46738ebed370c5c52ee0ad96dec8f459fb901c2ca4e285211eddf9
03bf1598)
```

2. Если ConfigMap валидный, отправьте сигнал на его принудительное обновление, выполнив следующую команду:

```
kubectl exec -n <namespace> <nats_name>-0 -c nats -- nats-server
--config /etc/nats-config/nats.conf --signal
reload=/var/run/nats/nats.pid
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<nats_name>` — имя экземпляра NATS: `nats-main`, `nats-bridge` или `nats-space`.

3. Если ConfigMap не валидный, вернитесь к началу инструкции и проверьте валидность внесенных вами изменений.

Устранение проблем при эксплуатации системы

В данном разделе приведены возможные проблемы при работе системы и способы их решения.

ClickHouse: не выполняются распределенные запросы (`on cluster`)

Проявление: в графическом интерфейсе не создаются новые объекты, такие как хранилища или активные списки.

В консоли кластера ClickHouse не выполняются распределенные запросы, содержащие секцию `on cluster`. Например, не создается распределенная таблица `test`:

```
create table test on cluster '{cluster}' (id UUID) Engine = MergeTree order by id
```

Причина: сбой запроса, который привел к невозможности обработки очереди запросов.

Решение:

1. Получите список подов, на которых расположены шарды ClickHouse:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/name=clickhouse
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

2. Найдите в логах шардов ошибку очереди запросов:

```
kubectl logs -n <namespace> <shard_name> --all-containers=true | grep
"DB::Exception: Cannot create status dirs"
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<shard_name>` — имя шарда из списка, полученного на шаге 1, например, `clickhouse-shard1-0`.

  Пример ошибки очереди запросов:

```
2024.12.12 10:31:51.883015 [ 410 ] {} <Error> DDLWorker: Unexpected error (1559
times in a row), will try to restart main thread: Code: 341. DB::Exception:
Cannot create status dirs for /clickhouse/task_queue/ddl/query-0007458048, most
likely because someone is deleting it concurrently. (UNFINISHED), Stack trace
(when copying this message, always include the lines below)
```

3. Сохраните идентификатор запроса в очереди, вызвавшего ошибку: в примере выше — `/clickhouse/task_queue/ddl/query-0007458048`.
4. Проверьте размер очереди запросов ClickHouse на шарде, на котором возникла ошибка:

```
kubectl exec -n <namespace> <clickhouse_shard_name> -- clickhouse-keeper-client -h
127.0.0.1 -p 2181 -q "ls 'clickhouse/task_queue/ddl'" 2>/dev/null | tr ' ' "\n"| wc
-l
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<clickhouse_shard_name>` — имя шарда ClickHouse, на котором возникла ошибка, например, `clickhouse-shard1-0`.

5. Удалите запрос из очереди ClickHouse:

```
kubectl exec -n <namespace> <clickhouse_shard_name> -it -- clickhouse-keeper-client
```

```
-h 127.0.0.1 -p 2181 -q "rmr '<query_id>'"
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<clickhouse_shard_name>` — имя шарда ClickHouse, на котором возникла ошибка, например, `clickhouse-shard1-0`;
- `<query_id>` — идентификатор запроса из сообщения об ошибке, полученный на шаге 3.

6. Повторно проверьте размер очереди запросов ClickHouse, выполнив команду из шага 4. Убедитесь, что размер очереди уменьшается, что подтверждает возобновление ее обработки.
7. Повторно выполните распределенный запрос.

ClickHouse: ошибка аутентификации при выполнении запросов

Проявление: запросы к БД ClickHouse завершаются с ошибкой `Authentication failed: password is incorrect, or there is no user with such name`. Это может происходить в любом разделе, где осуществляется работа с событиями, например, в разделе **Поиск**.

Причина: не совпадают пароли пользователей `owner`, `reader` или `writer` в объектном хранилище NATS Key/Value и в конфигурации ClickHouse.

Решение: установите единые пароли в NATS Key/Value и в конфигурации ClickHouse. Для этого:

1. Получите пароли пользователей из NATS Key/Value:
   a. Подключитесь к терминалу master-узла кластера.
   b. Подключитесь к терминалу пода `nats-main-box`, выполнив следующую команду:

   ```
   kubectl exec -itn <namespace> deployment/nats-main-box -c nats-box -- sh
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   c. Получите пути к ключам, в которых хранятся пароли ClickHouse, выполнив следующую команду:

   ```
   nats kv ls siem
   ```

   ▼ *Пример путей к ключам, в которых хранятся пароли ClickHouse*

   ```
   space-main/evo-clickhouse1/owner
   space-main/evo-clickhouse1/reader
   space-main/evo-clickhouse1/writer
   ```

   d. Просмотрите пароли пользователей ClickHouse, выполнив для каждого из них команду:

```
nats kv get siem <path_to_key> --raw
```

Здесь:

- `<path_to_key>` — путь к ключу, где хранится пароль.

2. Определите, пароль какого пользователя в конфигурации ClickHouse не совпадает с паролем в хранилище. Для этого:
   a. Осуществите проброс портов ClickHouse, выполнив на хосте с сетевым доступом к master-узлу кластера следующую команду:

   ```
   kubectl port-forward -n <namespace> service/clickhouse <local_click_port>:8123
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<local_click_port>` — порт на хосте, по которому будет осуществляться доступ к ClickHouse.

   b. В браузере откройте URL `localhost:<local_click_port>/play`, где `<local_click_port>` — порт с предыдущего шага. Отобразится графический клиент ClickHouse.
   c. Выполните следующие действия для каждого из пользователей `owner`, `reader` и `writer`:
      i. Введите логин и пароль в поля в правом верхнем углу экрана.
      ii. Выполните любой запрос, например, `SELECT * FROM default.disks`.

      Если возникает ошибка `Authentication failed: password is incorrect, or there is no user with such name. (AUTHENTICATION_FAILED)`, то пароль текущего пользователя необходимо исправить в таблице пользователей ClickHouse.

3. Получите список подов, на которых расположены шарды ClickHouse:

   ```
   kubectl -n <namespace> get pods -l app.kubernetes.io/name=clickhouse
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

4. Получите с любого шарда имя и пароль администратора ClickHouse, чтобы впоследствии выполнить запросы к базе данных через консольный клиент:
   a. Чтобы получить имя администратора ClickHouse, выведите подробную информацию о поде шарда:

```
kubectl -n <namespace> describe pod <shard_pod>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<shard_pod>` — имя пода, на котором расположен шард ClickHouse.

  Имя администратора ClickHouse расположено в разделе `Containers.<clickhouse_container_name>. Environment.CLICKHOUSE_ADMIN_USER`, где `<clickhouse_container_name>` — имя контейнера ClickHouse.

b. Получите пароль администратора ClickHouse из секрета `clickhouse` и декодируйте его из base64:

```
kubectl -n <namespace> get secret clickhouse -o jsonpath='{.data.admin-
password}' | base64 -d
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

5. Для каждого пользователя, для которого возникла ошибка аутентификации, обновите пароль, выполнив следующую команду на каждом шарде:

```
kubectl exec -n <namespace> <shard_pod> -- clickhouse-client --user
<clickhouse_admin> --password <clickhouse_admin_password> "ALTER USER
<user_to_edit> IDENTIFIED BY '<password_from_storage>'"
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<shard_pod>` — имя пода, на котором расположен шард ClickHouse.
- `<clickhouse_admin>` — имя администратора ClickHouse, полученное на шаге 4.
- `<clickhouse_admin_password>` — пароль администратора ClickHouse, полученный на шаге 4.
- `<user_to_edit>` — пользователь, пароль которого нужно изменить: `owner`, `reader` или `writer`.
- `<password_from_storage>` — пароль пользователя из объектного хранилища, полученный на шаге 1.

6. Для пользователя `writer` также следует обновить пароль в ConfigMap `clickhouse`, если он там присутствует:

a. Откройте ConfigMap `clickhouse`, выполнив следующую команду:

```
kubectl -n <namespace> edit configmap clickhouse
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

b. Проверьте, имеет ли ключ `remote_servers.default.shard.replica.user` значение `writer`.

▼ *Пример секции `remote_servers` для пользователя `writer`*

```
<remote_servers>
  <default>
    <shard>
      <replica>
        <host>clickhouse-shard0-0.cluster.local</host>
        <port>9000</port>
        <user>writer</user>
        <password>writer</password>
      </replica>
    </shard>
    <shard>
      <replica>
        <host>clickhouse-shard1-0.cluster.local</host>
        <port>9000</port>
        <user>writer</user>
        <password>writer</password>
      </replica>
    </shard>
  </default>
</remote_servers>
```



Если ключ `remote_servers.default.shard.replica.user` имеет значение, отличное от `writer` (например, `root` или `evo_user`), перейдите к шагу 7. Редактировать ConfigMap в этом случае не нужно.

c. Если ключ `remote_servers.default.shard.replica.user` имеет значение `writer`, укажите пароль пользователя `writer` из объектного хранилища как значение ключа `remote_servers.default.shard.replica.password`.



Ключей `remote_servers.default.shard.replica.password` столько же, сколько шардов. Необходимо установить пароль из объектного хранилища для каждого шарда.

▼ *Пример секции `remote_servers` с измененным паролем*

```
<remote_servers>
  <default>
    <shard>
      <replica>
        <host>clickhouse-shard0-0.cluster.local</host>
        <port>9000</port>
        <user>writer</user>
        <password>new_writer</password>
      </replica>
```

```
    </shard>
    <shard>
      <replica>
        <host>clickhouse-shard1-0.cluster.local</host>
        <port>9000</port>
        <user>writer</user>
        <password>new_writer</password>
      </replica>
    </shard>
  </default>
</remote_servers>
```

d. Сохраните изменения в ConfigMap.

7. Удалите поды ClickHouse. Они будут пересозданы автоматически и начнут использовать новые пароли:

```
kubectl -n <namespace> delete pods -l app.kubernetes.io/name=clickhouse
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

NATS: переполнение стрима аудита

Проявление: внутренние ошибки при работе с различными разделами системы, в том числе **Ресурсы → Коллекторы**, **Ресурсы → Активные списки**, **Поиск**, **Инструменты → Аудит источников**.

Причина: переполнение стрима аудита NATS, сообщения из которого не вычитываются другими сервисами системы.

Проверить размер стрима можно следующим образом:

1. Получите полное имя пода сервиса `nats-main-box`, выполнив следующую команду:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/instance=nats-main | grep
nats-main-box
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

▼ *Пример вывода имени пода* `nats-main-box`

```
nats-main-box-658b9fc759-x9fkb   1/1     Running   2 (61d ago)   118d
```

В примере выше полное имя пода `nats-main-box` — `nats-main-box-658b9fc759-x9fkb`.

2. Просмотрите информацию о стриме, выполнив следующую команду:

```
kubectl exec -n <namespace> <nats_main_box_name> -- nats stream info events-
streams-audit-prepared-audit
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<nats_main_box_name>` — полное имя пода `nats-main-box`, полученное на предыдущем шаге.

▼ *Пример вывода информации о стриме, когда он переполнен*

```
Information for Stream events-streams-audit-prepared-audit created 2025-04-24
09:16:55

             Subjects: events.streams.audit.prepared-audit
             Replicas: 1
              Storage: File

Options:

            Retention: Interest
       Acknowledgments: true
        Discard Policy: Old
      Duplicate Window: 2m0s
     Allows Msg Delete: true
          Allows Purge: true
        Allows Rollups: false

Limits:

      Maximum Messages: unlimited
   Maximum Per Subject: unlimited
         Maximum Bytes: unlimited
           Maximum Age: unlimited
  Maximum Message Size: unlimited
     Maximum Consumers: unlimited

State:

             Messages: 3,818,150
                Bytes: 28 GiB
        First Sequence: 4,237,386 @ 2025-06-16 11:53:57
         Last Sequence: 7,419,535 @ 2025-07-24 10:58:30
      Active Consumers: 1
```

В примере выше стрим переполнен: его размер (`State.Bytes`) достиг 28 ГиБ. Нормальный размер стрима должен находиться около 0 КиБ, поскольку при нормальной работе системы сообщения вычитываются в режиме реального времени.

Решение: удалите контроллеры Deployment коллекторов, которые используют образ коллектора

версии ниже 2.3.0. Для этого:

1. Получите список контроллеров Deployment для коллекторов в системе, выполнив следующую команду:

   ```
   kubectl -n <namespace> get deployments | grep -- -collector-
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   ▼ *Пример вывода списка контроллеров*

   ```
   collector-9c7baa5d-011a-4fdb-84d9-dd577303782e-main     1/1     1            1
   61d
   wl.db66eb9e-6a03-4919-a23a-edf8fc8a9de7-collector-main  1/1     1            1
   61d
   ```

   Контроллеры Deployment из списка, имя которых начинается с префикса `collector`, используют версию коллектора ниже 2.3.0. В примере это `collector-9c7baa5d-011a-4fdb-84d9-dd577303782e-main`.

2. Удалите контроллеры коллекторов, использующих версию коллектора ниже 2.3.0, выполнив для каждого из них следующую команду:

   ```
   kubectl -n <namespace> delete deployment <deployment>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<deployment>` — имя контроллера, полученное на предыдущем шаге.

3. Получите полное имя пода сервиса `nats-main-box`, выполнив следующую команду:

   ```
   kubectl -n <namespace> get pods -l app.kubernetes.io/instance=nats-main | grep
   nats-main-box
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   ▼ *Пример вывода имени пода* `nats-main-box`

   ```
   nats-main-box-658b9fc759-x9fkb   1/1     Running   2 (61d ago)   118d
   ```

   В примере выше полное имя пода `nats-main-box` — `nats-main-box-658b9fc759-x9fkb`.

4. Удалите стрим аудита из NATS, выполнив следующую команду:

   ```
   kubectl exec -n <namespace> <nats_main_box_name> -- nats stream rm events-streams-audit-prepared-audit
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<nats_main_box_name>` — полное имя пода `nats-main-box`, полученное на предыдущем шаге.

Пространства и сервисы: отсутствуют метрики или логи сервиса

Проявление: в карточке сервиса в разделе **Настройки** → **Пространства и сервисы** → **Сервисы** не отображаются метрики CPU и RAM, используемых сервисом, или отсутствуют логи сервиса.

Возможные причины:

- Переполнение хранилища логов `loki-minio`.
- Недостаточная пропускная способность системы сбора телеметрии.
- Превышение лимита количества лейблов логов Loki.
- Сетевая недоступность пространства.

Решение:



При возникновении данной проблемы не следует самостоятельно изменять конфигурацию элементов системы, перечисленных ниже. Изменения, не согласованные со службой поддержки, могут привести к нестабильной работе системы.

1. Соберите информацию о вашем кластере:
   a. Просмотрите статус пространства, в котором запущен сервис, в разделе **Настройки** → **Пространства и сервисы** → **Пространства** веб-интерфейса системы.
   b. Просмотрите статус сервиса в разделе **Настройки** → **Пространства и сервисы** → **Сервисы** веб-интерфейса системы.
   c. Подключитесь к терминалу master-узла кластера.
   d. Выгрузите логи подов системы сбора логов Loki в файл `loki-logs.txt`, выполнив следующую команду:

      ```
      kubectl -n kube-prometheus-stack logs -l app.kubernetes.io/component=write --all-containers=true > loki-logs.txt
      ```

   e. Выгрузите логи подов сервиса `vector-aggregator` в файл `vector-aggregator-logs.txt`, выполнив следующую команду:

```
kubectl -n <namespace> -l app.kubernetes.io/instance=evo.space.vector-aggregator
--all-containers=true > vector-aggregator-logs.txt
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

f. Выгрузите логи подов сервиса `vector-collector` в текстовый файл `vector-collector-logs.txt`, выполнив следующую команду:

```
kubectl -n <namespace> -l app.kubernetes.io/instance=evo.space.vector-collector
--all-containers=true > vector-collector-logs.txt
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

g. Получите информацию о стриме `r-space-telemetry` в основном экземпляре NATS:

i. Получите полное имя пода сервиса `nats-main-box`, выполнив следующую команду:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/instance=nats-main |
grep nats-main-box
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

▼ *Пример вывода имени пода* `nats-main-box`

```
nats-main-box-658b9fc759-x9fkb   1/1     Running   2 (61d ago)   118d
```

В примере выше полное имя пода `nats-main-box` — `nats-main-box-658b9fc759-x9fkb`.

ii. Получите таблицу стримов NATS, выполнив следующую команду:

```
kubectl exec -n <namespace> <nats_main_box_name> -- nats s ls
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<nats_main_box_name>` — полное имя пода `nats-main-box`, полученное на предыдущем шаге.

iii. Сохраните количество сообщений (`Messages`) и размер (`Size`) стрима `r-space-telemetry`.

h. Получите информацию о стриме `r-space-telemetry` в экземпляре NATS для управления пространствами:

i. Получите полное имя пода сервиса `nats-space-box`, выполнив следующую команду:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/instance=nats-space |
grep nats-space-box
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

▼ *Пример вывода имени пода* `nats-space-box`

```
nats-space-box-646fb89d4b-st2bd   1/1     Running   1 (7d1h ago)   16d
```

В примере выше полное имя пода `nats-space-box` — `nats-space-box-646fb89d4b-st2bd`.

ii. Получите таблицу стримов NATS, выполнив следующую команду:

```
kubectl exec -n <namespace> <nats_space_box_name> -- nats s ls
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<nats_space_box_name>` — полное имя пода `nats-space-box`, полученное на предыдущем шаге.

iii. Сохраните количество сообщений (`Messages`) и размер (`Size`) стрима `r-space-telemetry`.

i. Выгрузите ConfigMap `vector-aggregator-extravars` в файл `vector-aggregator-extravars.yml`, выполнив следующую команду:

```
kubectl -n <namespace> describe configmap vector-aggregator-extravars > vector-
aggregator-extravars.yml
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

j. Выгрузите ConfigMap `vector-collector-extravars` в файл `vector-collector-extravars.yml`, выполнив следующую команду:

```
kubectl -n <namespace> describe configmap vector-collector-extravars > vector-
collector-extravars.yml
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

2. Отправьте в службу поддержки по адресу support@rvision.ru следующую информацию:

- Технические характеристики вашего кластера:
  - Среднее число событий в секунду (EPS).
  - Количество master- и worker-узлов.
  - Технические характеристики узлов: количество ядер процессора, объем ОЗУ и хранилища, пропускная способность сетевого соединения.
- Статус сервиса, для которого наблюдается проблема, и пространства, где он развернут.
- Файлы логов:
  - `loki-logs.txt`.
  - `vector-aggregator-logs.txt`.
  - `vector-collector-logs.txt`.
- Метрики стримов `r-space-telemetry` в `nats-main` и `nats-space`.
- Файлы объектов ConfigMap:
  - `vector-aggregator-extravars.yml`.
  - `vector-collector-extravars.yml`.

3. Следуйте инструкциям, полученным в ответ на ваше обращение.

Работа в интерфейсе системы: внутренние ошибки при создании или запуске сущностей

Проблема: при попытке создать или запустить сущность в системе (например, коллектор) возникают сообщения о внутренних ошибках. В списке подов пространства имен, в котором установлена система, присутствуют поды со статусом `ImagePullBackOff`.

Причина: сборщик мусора удалил образы подов с одного или нескольких worker-узлов кластера, поскольку эти образы длительное время не использовались, а на узлах занято более 80% дискового пространства.

Решение: необходимо восстановить образы контейнеров для кластеров основного экземпляра системы и каждого из сателлитов.

Чтобы восстановить образы контейнеров:

1. Подключитесь к терминалу master-узла кластера.
2. Повторно загрузите образы подов на worker-узлы, выполнив следующую команду:

```
/opt/r-vision/common/bin/evoctl instance upload
```



Вы также можете выборочно загрузить образы подов отдельных модулей. Для этого:

1. Получите список образов модулей, выполнив следующую команду:

```
/opt/r-vision/common/bin/evoctl distrib images
```

2. Загрузите образы подов выбранных модулей, выполнив следующую команду:

   ```
   /opt/r-vision/common/bin/evoctl instance upload -m <module_list>
   ```

   Здесь:

   - `<module_list>` — список идентификаторов модулей через запятую.

   Например, для загрузки образов подов R-Space, Streams и Endpoint вы можете выполнить следующую команду:

   ```
   /opt/r-vision/common/bin/evoctl instance upload -m
   evo.space,evo.streams,evo.ep
   ```



Во избежание повторного срабатывания сборщика мусора избегайте заполнения дискового пространства на worker-узлах более чем на 80%.



Вы можете перенести образы контейнеров на отдельный диск, чтобы обезопасить целостность файлов в будущем. Подробности приведены в разделе Настройка отдельного диска под образы контейнеров.

Работа в интерфейсе системы: не отображаются метрики

Проявление: метрики отсутствуют в разделах интерфейса системы, где должны отображаться, таких как **Ресурсы → Коллекторы** или **Инструменты → Диспетчер запросов**.

В списке подов пространства имен `kube-prometheus-stack` присутствуют поды со статусом `OOMKilled`. В описании их контейнеров, полученном с помощью команды `kubectl describe`, отображается статус `OOMKilled` с кодом ошибки `137`, например:

```
Containers:
  grafana-sc-dashboard:
    State:          Running
      Started:      Mon, 05 May 2026 10:00:00 +0000
    Last State:     Terminated
      Reason:       OOMKilled
      Exit Code:    137
      Started:      Mon, 05 May 2026 09:30:00 +0000
      Finished:     Mon, 05 May 2026 10:00:00 +0000
    Ready:          True
    Restart Count:  1
```

Причина: высокая нагрузка на инсталляцию, для обработки метрик которой недостаточно выделенных лимитов для подов Prometheus.

Решение: увеличьте лимиты для подов Prometheus:

1. Откройте конфигурацию Prometheus, выполнив следующую команду:

   ```
   kubectl -n kube-prometheus-stack edit prometheus/kube-prometheus-stack-prometheus
   ```

2. Увеличьте лимиты ресурсов процессора и ОЗУ в разделе `spec.resources.limits`, например:

   ```
   spec:
     resources:
       limits:
         cpu: "2"
         memory: "4Gi"
   ```

Работа с событиями: ошибки в логах сервисов

Проявление: при работе системы с потоком событий в логах сервисов возникает ошибка, содержащая строку `EXT4-fs error (device sdu)`. Установлен провайдер хранилища Longhorn. Инсталляция ClickHouse находится внутри кластера.

Причина: конфигурация оборудования не соответствует техническим требованиям системы. Например, выделено мало ядер CPU.

Решение: приведите конфигурацию оборудования в соответствие с техническими требованиями системы.

Работа с событиями: не работает поиск, не строятся графики на дашбордах

Проявление: не работает поиск, не строятся графики на дашбордах.

При выполнении поискового запроса в консоли браузера возникает ошибка вида:

```
search::search::endpoints::simplified_search: error: Invalid state: Unable get
connection info: Unable to get connection info for event storage: grpc status:
code=13, message=error trying to connect: tcp open error: Address family not supported
by protocol (os error 97)
```

Причина:

- либо gRPC-порт инсталляции ClickHouse недоступен или не совпадает с указанным при установке системы, поэтому сервис поиска `search` не может подключиться к ClickHouse;
- либо сервис поиска не может подключиться к сервису `event-storage-manager`.

Решение:

1. Получите номер gRPC-порта из значения ключа `grpc_port` конфигурации ClickHouse.
2. Попробуйте выполнить поиск или построить график несколько раз, чтобы в логах пода накопились сообщения об ошибках.
3. Просмотрите логи пода `event-storage-manager`:
   a. Получите полное имя пода сервиса `event-storage-manager`, выполнив следующую команду:

      ```
      kubectl -n <namespace> get pods -l app.kubernetes.io/name=evo.streams.event-storage-manager
      ```

      Здесь:

      - `<namespace>` — название пространства имен, в котором установлена система.

        ▼ *Пример вывода имени пода* `event-storage-manager`

        ```
        NAME                                                READY   STATUS
        RESTARTS   AGE
        evo.streams.event-storage-manager-7db4bc4c99-wlqlk   1/1     Running   0
        21m
        ```

      В примере выше полное имя пода `event-storage-manager` — `evo.streams.event-storage-manager-7db4bc4c99-wlqlk`.

   b. Для просмотра логов пода `event-storage-manager` используйте руководство **Просмотр логов пода**. В качестве имени пода (`<pod_name>`) укажите полное имя пода `event-storage-manager`, полученное на предыдущем шаге.
4. В логах пода может присутствовать номер gRPC-порта ClickHouse, используемого сервисом `event-storage-manager`, как значение ключа `clickHouseConnectionInfo.port`. В примере ниже — `9100`:

   ```
   {"level":10,"time":1738668412977,"pid":1,"hostname":"evo.streams.event-storage-manager-66667479f5-74tn2","context":"EventStorageService","clickHouseConnectionInfo":{"host":"clickhouse","password":"REDACTED","port":9100,"user":"owner"},"requestId":"[missing]","msg":"connection info"}
   ```

   a. Если ключ `clickHouseConnectionInfo.port` присутствует в логах и номер порта отличается от номера в конфигурации ClickHouse:
      i. Укажите номер порта из `clickHouseConnectionInfo.port` как значение ключа `grpc_port` конфигурации ClickHouse.
      ii. Удалите поды ClickHouse. Они будут пересозданы автоматически и будут использовать новый gRPC-порт:

      ```
      kubectl -n <namespace> delete pods -l app.kubernetes.io/name=clickhouse
      ```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

b. Если ключ `clickHouseConnectionInfo.port` отсутствует или номера портов в конфигурации ClickHouse и логах совпадают, удалите поды сервисов `search` и `bff`:

```
kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.streams.search
kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.core.bff
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Удаленные поды будут автоматически пересозданы в течение нескольких минут. Когда поды будут иметь статус `Running`, можно будет повторить поиск или построение графика.

Если после пересоздания подов поиск и графики на дашбордах не работают, обратитесь в службу поддержки.

Диспетчер запросов: отсутствуют метрики запросов

Проявление: в разделе **Инструменты → Диспетчер запросов** не отображаются метрики RQL-запросов, такие как время выполнения и выделенная память. Для установки ClickHouse был использован существующий кластер.

Причина: при настройке кластера ClickHouse не были созданы служебные распределенные таблицы.

Решение: создайте таблицы в кластере ClickHouse. Для этого:

1. Получите список подов, на которых расположены шарды ClickHouse:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/name=clickhouse
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.



Вы можете получить название пространства имен, в котором установлена система, с помощью утилиты evoctl.

2. На любом шарде выполните следующие действия:

a. Чтобы получить имя администратора ClickHouse, выведите подробную информацию о поде шарда:

```
kubectl -n <namespace> describe pod <shard_pod>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<shard_pod>` — имя пода, на котором расположен шард ClickHouse.

  Имя администратора ClickHouse расположено в разделе `Containers.<clickhouse_container_name>.Environment.CLICKHOUSE_ADMIN_USER`, где `<clickhouse_container_name>` — имя контейнера ClickHouse.

b. Получите пароль администратора ClickHouse из секрета `clickhouse` и декодируйте его из base64:

```
kubectl -n <namespace> get secret clickhouse -o jsonpath='{.data.admin-
password}' | base64 -d
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

c. Подключитесь к консольному клиенту ClickHouse с учетными данными, полученными на предыдущих шагах:

```
kubectl -n <namespace> exec -it <shard_pod> -- clickhouse-client --user
<clickhouse_user> --password <clickhouse_password>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<shard_pod>` — имя пода, на котором расположен шард ClickHouse.
- `<clickhouse_user>` — имя пользователя ClickHouse.
- `<clickhouse_password>` — пароль пользователя ClickHouse.

d. Получите имя кластера ClickHouse, выполнив следующий запрос:

```
SELECT distinct cluster FROM system.clusters
```

e. Выполните следующие запросы для создания служебных таблиц:

```
CREATE TABLE IF NOT EXISTS default.disks ON CLUSTER '<cluster>' AS system.disks
ENGINE = Distributed('<cluster>', system, disks);
CREATE TABLE IF NOT EXISTS default.storage_policies ON CLUSTER '<cluster>' AS
system.storage_policies ENGINE = Distributed('<cluster>', system,
storage_policies);
CREATE TABLE IF NOT EXISTS default.parts ON CLUSTER '<cluster>' AS system.parts
ENGINE = Distributed('<cluster>', system, parts);
CREATE TABLE IF NOT EXISTS default.query_log ON CLUSTER '<cluster>' AS
system.query_log ENGINE = Distributed('<cluster>', system, query_log);
CREATE TABLE IF NOT EXISTS default.columns ON CLUSTER '<cluster>' AS
system.columns ENGINE = Distributed('<cluster>', system, columns);
```

```
CREATE TABLE IF NOT EXISTS default.parts_columns ON CLUSTER '<cluster>' AS
system.parts_columns ENGINE = Distributed('<cluster>', system, parts_columns);
CREATE TABLE IF NOT EXISTS default.tables ON CLUSTER '<cluster>' AS
system.tables ENGINE = Distributed('<cluster>', system, tables);
CREATE TABLE IF NOT EXISTS default.clusters ON CLUSTER '<cluster>' AS
system.clusters ENGINE = Distributed('<cluster>', system, clusters);
CREATE TABLE IF NOT EXISTS default.asynchronous_metrics ON CLUSTER '<cluster>'
AS system.asynchronous_metrics ENGINE = Distributed('{cluster}', system,
asynchronous_metrics);
```

Здесь:

- `<cluster>` — имя кластера, полученное на предыдущем шаге.

WebSocket: ошибки запросов

Проявление: не работают функции, использующие протокол WebSocket: например, поиск, импорт элементов экспертизы или скачивание отчетов. В консоли браузера запросы по протоколу WebSocket завершаются с ошибками.

Причина: для сервиса API `evo.streams.bff` не включены подписки WebSocket, или есть проблема с соединениями WebSocket на уровне инфраструктуры.

Решение:

1. Откройте ConfigMap `evo.streams-bff-env`, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.streams-bff-env
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

2. Если переменная `EVO_STREAMS_BFF_GRAPHQL_ENABLE_PLAYGROUND` имеет значение `"false"`, включите подписки WebSocket:

   a. Установите для переменной `EVO_STREAMS_BFF_GRAPHQL_SUBSCRIPTION_TRANSPORT_WS_ENABLED` значение `"true"`.

   b. Закройте ConfigMap.

   c. Удалите под сервиса `evo.streams.bff`, выполнив следующую команду:

   ```
   kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.streams.bff
   ```

   Под будет пересоздан со включенными подписками WebSocket.

3. Если способ выше не решил проблему или переменная `EVO_STREAMS_BFF_GRAPHQL_ENABLE_PLAYGROUND` уже имеет значение `"true"`, выполните запрос с помощью любой утилиты wss, например, wscat, к следующему URL:

```
wss://<system_url>/api/graphql
```

Здесь:

- `<system_url>` — URL, по которому осуществляется доступ к системе.

Если после выполнения вышеперечисленных действий попытка установить WSS-соединение завершается с ошибкой, то вероятная причина проблемы — нарушения в инфраструктуре, в которой установлена система: например, сбой сети, некорректные настройки прокси или ограничение файрвола.

Коллекторы: падение или отсутствие EPS

Проявление: значительное падение EPS (количества событий в секунду) или отсутствие событий на коллекторе.

Ошибки в логе коллектора:

```
"line": "2024-11-28T11:39:52.739224Z ERROR source{component_kind=\"source\"
component_id=4b5ae224-78fc-4c00-9614-5ecb41871f2b_eJ__wBp4vtuHe9YVoOxCm_syslog
component_type=socket}: vector::internal_events::codecs: Failed framing bytes.
error=Unable to decode input as UTF8 error_code=\"decoder_frame\"
error_type=\"parser_failed\" stage=\"processing\" internal_log_rate_limit=true",
"line": "2024-11-28T11:39:52.753215Z ERROR source{component_kind=\"source\"
component_id=4b5ae224-78fc-4c00-9614-5ecb41871f2b_eJ__wBp4vtuHe9YVoOxCm_syslog
component_type=socket}: vector::topology: An error occurred that Vector couldn't
handle: the task panicked and was aborted.",
```

Причина: выбран некорректный тип точки входа, не соответствующий формату поступающих событий; например, в точку входа типа Syslog поступают события в формате JSON.

Решение: необходимо проверить конфигурацию точек входа конвейеров, установленных в коллекторе, и привести их типы в соответствие с форматом поступающих событий.

Коллекторы: не отображаются метрики коллекторов, конвейеров и элементов конвейеров

Проявление: в карточках коллекторов и конвейеров, а также на элементах конвейеров в режиме активной версии не отображаются метрики.

Причина: поду сервиса сбора метрик Prometheus не хватает размера тома, определяемого ресурсом PVC (Persistent Volume Claim), для хранения данных.

Решение: нужно увеличить PVC для Prometheus. Для этого:

1. Получите имя PVC Prometheus, выполнив следующую команду:

```
kubectl -n kube-prometheus-stack describe pod prometheus-kube-prometheus-stack-
```

```
prometheus-0
```

Имя PVC Prometheus является значением ключа `Volumes.<prometheus_volume_name>.ClaimName`.

▼ *Пример имени PVC Prometheus*

```
# ...
Volumes:
  prometheus-kube-prometheus-stack-prometheus-db:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in
the same namespace)
    ClaimName:  prometheus-kube-prometheus-stack-prometheus-db-prometheus-kube-
prometheus-stack-prometheus-0
    ReadOnly:   false
# ...
```

В этом случае имя PVC Prometheus — `prometheus-kube-prometheus-stack-prometheus-db-prometheus-kube-prometheus-stack-prometheus-0`.

2. Откройте конфигурацию PVC для Prometheus, выполнив следующую команду:

```
kubectl -n kube-prometheus-stack edit pvc <prometheus_pvc_name>
```

Здесь:

- `<prometheus_pvc_name>` — имя PVC Prometheus, полученное на предыдущем шаге.

3. Увеличьте размер PVC, отредактировав значение ключа `spec.resources.requests.storage`. Формат и единицы измерения размеров PVC приведены в документации Kubernetes.

```
# ...
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      # Отредактируйте значение этого ключа.
      storage: 15Gi
# ...
```

4. Закройте файл. Kubernetes перераспределит дисковое пространство в соответствии с обновленным PVC, и размер тома изменится через некоторое время.

Коллекторы: при установке нового конвейера или перезапуске коллектора не поступают события

Проявление: при установке нового конвейера в коллектор не поступают события на конвейер, или при перезапуске коллектора перестают поступать события в весь коллектор.

Причина: занят порт, который используют конвейеры коллектора.

Решение:

1. Просмотрите конфигурации каждого из включенных конвейеров, установленных в коллекторе, и сохраните список портов, используемых точками входа этих конвейеров.
2. Скопируйте идентификатор коллектора из его карточки.
3. Получите список портов узла кластера, используемых коллектором. Для этого на узле кластера выполните следующую команду:

   ```
   kubectl -n <namespace> get svc | grep <collector_id>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<collector_id>` — идентификатор коллектора, полученный на предыдущем шаге.

   ▼ *Пример вывода команды для коллектора с идентификатором* `a6985d07-6400-49ef-a47c-e5c1a411f8f3`

   ```
   wl-a6985d07-6400-49ef-a47c-e5c1a411f8f3-collector-main         ClusterIP
   10.233.37.47    <none>        31537/TCP,8686/TCP
   wl-a6985d07-6400-49ef-a47c-e5c1a411f8f3-collector-main-ports   NodePort
   10.233.10.154   <none>        31537:31537/TCP
   ```

   Один из конвейеров, запущенных в коллекторе, использует TCP-порт 31537.

4. Обратите внимание на сервис `main-ports` с типом `NodePort`:

   a. Если в выводе команды есть все порты, используемые конвейерами, но события все равно не поступают в систему, обратитесь в службу поддержки по адресу support@rvision.ru.

   b. Если сервис `main-ports` отсутствует в выводе команды, или в списке его портов нет какого-либо из портов, который используют конвейеры, выполните следующие действия:

      i. Получите полное имя пода коллектора, выполнив команду ниже.

      Каждому поду присваивается случайный идентификатор, добавляемый к имени сервиса, для которого создается под.

      ```
      kubectl -n <namespace> get pods -l app.kubernetes.io/name=wl-<collector_id>
      ```

      Здесь:

      - `<namespace>` — название пространства имен, в котором установлена система.
      - `<collector_id>` — идентификатор коллектора, полученный на предыдущем шаге.

      ▼ *Пример вывода имени пода для коллектора с идентификатором* `d712394d-90ae-41fb-b998-2dd5b580d77d`

```
NAME                                                              READY
STATUS    RESTARTS   AGE
wl.d712394d-90ae-41fb-b998-2dd5b580d77d-collector-main-59fzz62v   2/2
Running   0          3h10m
```

ii. Получите список событий пода коллектора, выполнив следующую команду:

```
kubectl -n <namespace> events --for pod/<collector_pod_name>
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<collector_pod_name>` — полное имя пода коллектора, полученное на предыдущем шаге.

iii. Если в списке событий обнаружились события, указывающие на то, что требуемый конвейером порт занят, освободите порт, используя информацию из событий, и перезапустите коллектор.

iv. Если в списке событий нет событий, указывающих на то, что порты заняты, изучите правила `iptables`, настроенные в системе. Возможно, они запрещают использование необходимого порта. В таком случае следует изменить или выключить эти правила и перезапустить коллектор.

v. Если в результате вышеперечисленных действий события все равно не поступают в систему, обратитесь в службу поддержки по адресу support@rvision.ru.

Точка входа типа Database: загрузка драйвера прекращается по таймауту

Проявление: загрузка файла драйвера в точку входа типа Database происходит слишком долго и прекращается по таймауту.

Причина: недостаточно ресурсов сервиса `evo.streams.driver-manager`.

Решение: увеличьте ресурсы сервиса `evo.streams.driver-manager`. Для этого:

Через командную строку

1. Подключитесь к кластеру.
2. Откройте конфигурацию контроллера Deployment `evo.streams.driver-manager`, выполнив команду:

```
kubectl edit -n <namespace> deployment evo.streams.driver-manager
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

3. Увеличьте значения лимитов процессора (`cpu`) и оперативной памяти (`memory`) в секции

`spec.template.spec.containers.resources`.

Пример секции `resources` в конфигурации контроллера

```
resources:
  limits:
    cpu: 500m
    memory: 300Mi
  requests:
    cpu: 25m
    memory: 100Mi
```

4. Сохраните обновленную конфигурацию. Под контроллера Deployment `evo.streams.driver-manager` будет автоматически перезапущен.

Через Lens

1. Подключитесь к кластеру.
2. В выпадающем меню свойств кластера выберите раздел **Workloads** → **Deployments**.
3. Выберите контроллер Deployment `evo.streams.driver-manager`. В правой части экрана отобразится панель настроек контроллера.
4. Нажмите на кнопку **Edit** в верхней части правой панели. Отобразится конфигурация контроллера Deployment.
5. Увеличьте значения лимитов процессора (`cpu`) и оперативной памяти (`memory`) в секции `spec.template.spec.containers.resources`.

   Пример секции `resources` в конфигурации контроллера

   ```
   resources:
     limits:
       cpu: 500m
       memory: 300Mi
     requests:
       cpu: 25m
       memory: 100Mi
   ```

6. Сохраните обновленную конфигурацию. Под контроллера Deployment `evo.streams.driver-manager` будет автоматически перезапущен.

Точка входа типа Database (MS SQL): не поступают события

Проявление: не поступают события в точку входа типа Database с драйвером MS SQL.

Ошибки в логе коллектора:

```
ERROR vector::sources::database: JDBC Server unexpected stopped error=Unable to open
JDBC Connection
PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException:
```

```
unable to find valid certification path to requested target.
```

Причина: ошибка при разборе сертификата сервера БД, что приводит к невозможности установить защищенное SSL-соединение.

Решение: отключите SSL для соединения с БД. Для этого измените настройки точки входа, добавив в **Адрес подключения** параметры `trustServerCertificate=false` и `encrypt=false`.

Пример строки подключения:

```
jdbc:sqlserver://localhost;user=user;password=password;trustServerCertificate=false;en
crypt=false
```



Если подобная ошибка возникает при использовании другой СУБД, кроме MS SQL, получите информацию о способах отключения SSL в официальной документации JDBC-драйвера соответствующей СУБД.

Базы данных событий: не удается зарегистрировать новую базу данных

Проявление: в интерфейсе системы не удается зарегистрировать новую базу данных событий. ClickHouse установлен в выделенном кластере (вынесенная установка). В логах сервера ClickHouse возникает ошибка вида:

```
Error when deleting table of event storage: All connection tries failed while
connecting to ZooKeeper. nodes:
[fe80::20c:29ff:fe89:fa80%ens33]:9181\nPoco::Exception. Code: 1000, e.code() = 111,
Connection refused (version 24.3.2.23 (official build)),
[fe80::20c:29ff:fe89:fa80%ens33]:9181\n
```

Причина: если во время установки системы выбрана вынесенная установка ClickHouse и указано полное доменное имя (FQDN) сервера ClickHouse, то при выполнении запроса доменное имя по умолчанию преобразуется в адрес IPv6. Это приводит к ошибке, если у сервера ClickHouse нет IPv6-адреса.

Решение: следует отключить использование IPv6 на сервере, на котором установлен ClickHouse. Для этого выполните следующие действия на сервере, на котором установлен ClickHouse:

1. В каталоге `/etc/sysctl.d` создайте файл `99-disable-ipv6.conf` со следующим содержимым:

   ```
   net.ipv6.conf.all.disable_ipv6 = 1
   net.ipv6.conf.default.disable_ipv6 = 1
   net.ipv6.conf.lo.disable_ipv6 = 1
   ```

2. Сохраните файл.
3. Примените изменения с помощью следующей команды:

```
sudo sysctl --system
```

4. Перезагрузите службу ClickHouse:

```
sudo service clickhouse-server restart
```

Хранилище событий аудита: не поступают новые события

Проявление: в хранилище событий аудита не поступают новые события.

Причина: сбой чтения событий аудита из стрима NATS.

Решение: проверьте, совпадает ли номер последней переданной последовательности в стриме и consumer (потребителе) событий аудита. Для этого:

1. Получите полное имя пода сервиса `nats-main-box`, выполнив следующую команду:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/instance=nats-main | grep
nats-main-box
```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

   ▼ *Пример вывода имени пода* `nats-main-box`

```
nats-main-box-658b9fc759-x9fkb   1/1     Running   2 (61d ago)   118d
```

   В примере выше полное имя пода `nats-main-box` — `nats-main-box-658b9fc759-x9fkb`.

2. Получите номер последней переданной последовательности стрима аудита `events-streams-audit-system-audit`:

   a. Получите информацию о стриме `events-streams-audit-system-audit`, выполнив следующую команду:

```
kubectl exec -n <namespace> <nats_main_box_name> -- nats stream info events-
streams-audit-system-audit
```

      Здесь:

      - `<namespace>` — название пространства имен, в котором установлена система.
      - `<nats_main_box_name>` — полное имя пода `nats-main-box`, полученное ранее.

      ▼ *Пример вывода информации о стриме* `events-streams-audit-system-audit`

```
Information for Stream events-streams-audit-system-audit created 2025-06-27
```

```
01:40:48

                Subjects: events.streams.audit.v1.SystemAuditEvent
                Replicas: 1
                 Storage: File

Options:

               Retention: WorkQueue
         Acknowledgments: true
          Discard Policy: Old
        Duplicate Window: 2m0s
       Allows Msg Delete: true
            Allows Purge: true
  Allows Per-Message TTL: false
          Allows Rollups: false

Limits:

        Maximum Messages: unlimited
     Maximum Per Subject: unlimited
           Maximum Bytes: unlimited
             Maximum Age: unlimited
    Maximum Message Size: unlimited
       Maximum Consumers: unlimited

State:

             Host Version: 2.11.2
      Required API Level: 0 hosted at level 1
                Messages: 40
                   Bytes: 19 KiB
          First Sequence: 131,385 @ 2025-11-01 13:55:26
           Last Sequence: 131,424 @ 2025-11-01 14:29:15
         Active Consumers: 1
      Number of Subjects: 1
```

b. Сохраните номер последней переданной последовательности, указанный в значении ключа `Last Sequence`. В примере выше — `131,424`.

3. Получите номер последней полученной последовательности из информации о consumer `nats_system_audit_events`:

   a. Получите информацию о consumer `nats_system_audit_events`, выполнив следующую команду:

   ```
   kubectl exec -n <namespace> <nats_main_box_name> -- nats consumer info events-streams-audit-system-audit nats_system_audit_events
   ```

   Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<nats_main_box_name>` — полное имя пода `nats-main-box`, полученное ранее.

▼ *Пример вывода информации о consumer* `nats_system_audit_events`

```
Information for Consumer events-streams-audit-system-audit >
nats_system_audit_events created 2025-10-10T11:06:44Z

Configuration:

                    Name: nats_system_audit_events
               Pull Mode: true
         Filter Subjects: events.streams.audit.v1.SystemAuditEvent
          Deliver Policy: All
              Ack Policy: Explicit
                Ack Wait: 30.00s
           Replay Policy: Instant
         Max Ack Pending: 1,000
       Max Waiting Pulls: 512

State:

            Host Version: 2.11.2
      Required API Level: 0 hosted at level 1
  Last Delivered Message: Consumer sequence: 307,282 Stream sequence: 205,215
Last delivery: 268s ago
    Acknowledgment Floor: Consumer sequence: 307,184 Stream sequence: 205,175
Last Ack: 1h32m6s ago
        Outstanding Acks: 40 out of maximum 1,000
    Redelivered Messages: 40
    Unprocessed Messages: 0
           Waiting Pulls: 2 of maximum 512
```

b. Сохраните номер последнего полученного сообщения, указанный в поле `Last Delivered Message`, в значении ключа `Stream sequence`. В примере выше — `205,215`.

4. Сравните номер последней переданной последовательности, полученный из информации о стриме, с номером последнего полученного сообщения. В примере выше — `205,215`. Этот номер значительно больше, чем `131,424`. В таком случае переданные последовательности не читаются из стрима до тех пор, пока их номер не достигнет `Last Delivered Message`, что и приводит к отсутствию новых событий в хранилище.
5. Если номер последнего полученного сообщения значительно больше номера последней переданной последовательности, выполните следующие действия:

   a. Удалите consumer, выполнив следующую команду:

```
kubectl exec -n <namespace> <nats_main_box_name> -- nats consumer rm events-
streams-audit-system-audit nats_system_audit_events
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<nats_main_box_name>` — полное имя пода `nats-main-box`, полученное ранее.

b. Удалите под `audit-manager`, выполнив следующую команду:

```
kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.streams.audit-manager
```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

Consumer и под будут пересозданы автоматически, и события начнут поступать в хранилище аудита.



Если вышеперечисленные действия не решили проблему, обратитесь к следующим разделам, описывающим схожие проблемы:

- Хранилища событий: не отображаются события за последние несколько минут
- Хранилища событий: не поступают новые события из-за превышения максимального размера события
- Хранилище событий: не выполняется поиск или не поступают события

Хранилища событий: не отображаются события за последние несколько минут

Проявление: при просмотре хранилища событий или поиске по нему не отображаются события, поступившие в систему за последние несколько минут.

Причина: отставание времени на узлах от текущего. Получить время на узлах можно, выполнив в их консоли команду `date`.

Решение: настройте синхронизацию времени на узлах согласно документации операционной системы, которая установлена на этих узлах. Например, документация по синхронизации времени в Astra Linux расположена на справочном портале разработчика ОС.

Хранилища событий: не поступают новые события из-за превышения максимального размера события

Проявление: в хранилище событий не поступают события. В логах пода `nats-space` возникают ошибки, содержащие сообщение `maximum payload exceeded`, например:

```
[74] 2025/07/23 09:03:12.276656 [ERR] 10.233.112.88:45352 - cid:401631 - maximum payload exceeded: 13466645 vs 4194304
```

Причина: в систему поступило событие, размер которого превышает максимальный размер сообщения NATS, задаваемый параметром конфигурации `max_payload`. По умолчанию он равен 4 МБ

(4 194 304 байт).

Решение: увеличьте значение параметра `max_payload`.



Максимальное допустимое значение `max_payload` — 64 МБ. Однако не рекомендуется увеличивать его более чем до 8 МБ для поддержания оптимальной производительности системы и контролируемого потребления вычислительных ресурсов.



Увеличить значение параметра `max_payload` следует в центральном кластере и во всех кластерах сателлитов.

Чтобы увеличить значение `max_payload`:

1. Добавьте параметр `max_payload` или отредактируйте его значение в следующих объектах ConfigMap:
   - в центральном кластере — `nats-main-config`, `nats-bridge-config` и `nats-space-config`;
   - в кластере сателлита — только `nats-space-config`.

   Для этого выполните следующие действия:

   a. Откройте ConfigMap, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit configmap <configmap>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<configmap>` — имя ConfigMap: `nats-main-config`, `nats-bridge-config` или `nats-space-config`.

   b. По пути `data.nats.conf` добавьте ключ `max_payload` или отредактируйте его значение, если он уже существует. В качестве значения этого ключа укажите необходимый максимальный размер события в мегабайтах.

   ▼ *Пример ConfigMap с ключом* `max_payload`

   ```
   data:
     nats.conf: |
       {
         "http_port": 8222,
         "jetstream": {
           "domain": "main",
           "max_file_store": 30Gi,
           "max_memory_store": 0,
           "store_dir": "/data"
         },
         "lame_duck_duration": "30s",
         "lame_duck_grace_period": "10s",
   ```

```
    "leafnodes": {
      "no_advertise": true,
      "port": 7422
    },
    # Добавьте или отредактируйте данный параметр.
    "max_payload": 8Mb,
    "pid_file": "/var/run/nats/nats.pid",
    "port": 4222,
    "server_name": $SERVER_NAME
  }
```

2. Подождите несколько минут. Когда Kubernetes применит новую конфигурацию, проблема будет устранена.



Если новая конфигурация не применяется в течение длительного времени, выполните следующие действия:

1. Проверьте корректность ConfigMap NATS, выполнив следующую команду для `nats-main`, `nats-bridge` и `nats-space` (в случае центрального кластера) или только `nats-space` (в случае кластера сателлита):

   ```
   kubectl exec -n <namespace> <nats_name>-0 -c nats -- nats-server
   --config /etc/nats-config/nats.conf -t
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<nats_name>` — имя экземпляра NATS: `nats-main`, `nats-bridge` или `nats-space`.

   Если ConfigMap валидный, отобразится сообщение, содержащее строку `configuration file /etc/nats-config/nats.conf is valid`, например:

   ```
   nats-server: configuration file /etc/nats-config/nats.conf is
   valid
   (sha256:0f46738ebed370c5c52ee0ad96dec8f459fb901c2ca4e285211eddf9
   03bf1598)
   ```

2. Если ConfigMap валидный, отправьте сигнал на его принудительное обновление, выполнив следующую команду:

   ```
   kubectl exec -n <namespace> <nats_name>-0 -c nats -- nats-server
   --config /etc/nats-config/nats.conf --signal
   reload=/var/run/nats/nats.pid
   ```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<nats_name>` — имя экземпляра NATS: `nats-main`, `nats-bridge` или `nats-space`.

3. Если ConfigMap не валидный, вернитесь к началу инструкции и проверьте валидность внесенных вами изменений.

После изменения максимального размера сообщения вы можете получить и проанализировать выборку событий большого размера, из-за которых могла возникнуть данная проблема. Для этого:

1. Перейдите в раздел **Поиск**.
2. В списке хранилищ событий выберите хранилище, для которого наблюдалась проблема.
3. Выполните следующий запрос:

```
select * where length(toString(data)) > <event_size>
```



Здесь:

- `<event_size>` — размер события в байтах.

Возможные причины появления событий большого размера:

- Если проблема возникает в хранилище аудита:
  - Некорректная настройка параметров системы.
  - Ошибка в логике работы системы.
- Если проблема возникает не в хранилище событий аудита: ошибка настройки конвейера, отправляющего события в глобальную шину, которая использует NATS.

Если вы считаете, что события большого размера появились из-за ошибки в логике работы системы, обратитесь в службу поддержки по адресу support@rvision.ru.

Хранилище событий: не выполняется поиск или не поступают события

Проявление: возникает внутренняя ошибка сервера при поиске в хранилище событий или в хранилище не поступают новые события.

Причина: на некоторых шардах ClickHouse отсутствует таблица, соответствующая хранилищу событий.

Решение: проведите автоматическое восстановление таблиц хранилищ событий. Для этого выполните следующие действия:

1. Откройте ConfigMap сервиса `evo.streams.event-storage-manager`. Для этого выполните следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.streams-event-storage-manager-env
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

2. Измените значение переменной `EVO_STREAMS_EVENT_STORAGE_MANAGER_RECREATE_EVENT_STORAGES` на `"true"`.

   Восстановление таблиц хранилищ событий занимает некоторое время. В процессе восстановления liveness-пробы сервиса `event-storage-manager` могут завершиться с ошибкой, что приведет к циклическому перезапуску этого сервиса. Во избежание такого поведения рекомендуется увеличить интервал между liveness-пробами, выполнив следующие действия:

   1. Откройте конфигурацию контроллера Deployment `evo.streams.event-storage-manager`, выполнив следующую команду:

      ```
      kubectl -n <namespace> edit deployment evo.streams.event-storage-manager
      ```

      Здесь:

      - `<namespace>` — название пространства имен, в котором установлена система.

   2. Скопируйте и сохраните текущие значения параметров в секции `livenessProbe`.

      ▼ *Пример содержимого секции* `livenessProbe`

      ```
      livenessProbe:
        grpc:
          port: 3000
          service: ''
        initialDelaySeconds: 10
        timeoutSeconds: 3
        periodSeconds: 10
        successThreshold: 1
        failureThreshold: 3
      ```

   3. Увеличьте значения параметров `timeoutSeconds`, `periodSeconds` и `failureThreshold`.

   4. Сохраните изменения.

> После перезапуска сервиса `evo.streams.event-storage-manager` рекомендуется восстановить настройки liveness-проб по умолчанию.

3. Удалите под сервиса `evo.streams.event-storage-manager`, выполнив следующую команду:

   ```
   kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.streams.event-storage-manager
   ```

   Под будет пересоздан автоматически.

4. Дождитесь, когда под `evo.streams.event-storage-manager` перейдет в статус `Running`. Инструкция по проверке статуса пода приведена в разделе Просмотр состояния подов кластера.
5. Когда под перейдет в статус `Running`, перейдите в веб-интерфейс системы и убедитесь, что поиск по хранилищам событий выполняется успешно и события поступают в хранилища.
6. Откройте ConfigMap сервиса `evo.streams.event-storage-manager`. Для этого выполните следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.streams-event-storage-manager-env
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

7. Измените значение переменной `EVO_STREAMS_EVENT_STORAGE_MANAGER_RECREATE_EVENT_STORAGES` обратно на `"false"`.
8. Если вы меняли интервал liveness-проб, восстановите интервал по умолчанию в конфигурации `evo.streams.event-storage-manager`. Ее можно открыть, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit deployment evo.streams.event-storage-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

Хранилища событий и активные списки: ошибка размера таблицы при удалении

Проявление: используется вынесенная инсталляция ClickHouse. Не удается удалить хранилище с большим количеством событий или активный список с большим количеством записей. Возникает ошибка с ключевыми словами `Size is greater than max_[table/partition]_size_to_drop`, например:

```
Delete event storage error: Error when dropping table of event storage: Table or
Partition in default.EventStorage_31b091be_8da6_4979_96aa_d8f5ec975d69_local was not
dropped.
Reason:
1. Size (76.38 GB) is greater than max_[table/partition]_size_to_drop (50.00 GB)
```

```
2. File '/var/lib/clickhouse/flags/force_drop_table' intended to force DROP doesn't
exist
```

Причина: в настройках ClickHouse не заданы параметры `max_table_size_to_drop` и/или `max_partition_size_to_drop` или для них используется слишком низкое значение.

Решение: измените значения параметров `max_table_size_to_drop` и `max_partition_size_to_drop`. Для этого выполните следующие действия на каждом из узлов кластера ClickHouse:

1. Перейдите в поддиректорию `config.d` директории установки сервера ClickHouse.
2. Откройте для редактирования файл `config.xml`.
3. Добавьте или измените значения параметров `max_table_size_to_drop` и `max_partition_size_to_drop` внутри тега `<merge_tree>`:

   ```
   <merge_tree>
       <max_table_size_to_drop>0</max_table_size_to_drop>
       <max_partition_size_to_drop>0</max_partition_size_to_drop>
   </merge_tree>
   ```

   Рекомендуется установить для обоих параметров значение `0`, чтобы снять ограничение на размер удаляемых партиций и таблиц. Однако при необходимости вы можете установить любое значение больше 50 ГБ в байтах: например, `100000000000` (100 ГБ).

Активный список: не выполняется поиск или не обновляются записи

Проявление: возникает внутренняя ошибка сервера при поиске в активном списке или записи в нем не обновляются.

Причина: на некоторых шардах ClickHouse отсутствует таблица, соответствующая активному списку.

Решение: проведите автоматическое восстановление таблиц активных списков. Для этого выполните следующие действия:

1. Откройте ConfigMap сервиса `evo.streams.active-list-manager`. Для этого выполните следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.streams-active-list-manager-env
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

2. Измените значение переменной `EVO_STREAMS_ACTIVE_LIST_MANAGER_RECREATE_ACTIVE_LISTS` на `"true"`.

   Восстановление таблиц активных списков занимает некоторое время. В

процессе восстановления liveness-пробы сервиса `active-list-manager` могут завершиться с ошибкой, что приведет к циклическому перезапуску этого сервиса. Во избежание такого поведения рекомендуется увеличить интервал между liveness-пробами, выполнив следующие действия:

1. Откройте конфигурацию контроллера Deployment `evo.streams.active-list-manager`, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit deployment evo.streams.active-list-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

2. Скопируйте и сохраните текущие значения параметров в секции `livenessProbe`.

   ▼ *Пример содержимого секции* `livenessProbe`

   ```
   livenessProbe:
     grpc:
       port: 3000
       service: ''
     initialDelaySeconds: 10
     timeoutSeconds: 3
     periodSeconds: 10
     successThreshold: 1
     failureThreshold: 3
   ```

3. Увеличьте значения параметров `timeoutSeconds`, `periodSeconds` и `failureThreshold`.
4. Сохраните изменения.

После перезапуска сервиса `evo.streams.active-list-manager` рекомендуется восстановить настройки liveness-проб по умолчанию.

3. Удалите под сервиса `evo.streams.active-list-manager`, выполнив следующую команду:

   ```
   kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.streams.active-list-manager
   ```

   Под будет пересоздан автоматически.

4. Дождитесь, когда под `evo.streams.active-list-manager` перейдет в статус `Running`. Инструкция по проверке статуса пода приведена в разделе Просмотр состояния подов кластера.
5. Когда под перейдет в статус `Running`, перейдите в веб-интерфейс системы и убедитесь, что поиск

по активным спискам выполняется успешно и записи в них обновляются.

6. Откройте ConfigMap сервиса `evo.streams.active-list-manager`. Для этого выполните следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.streams-active-list-manager-env
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

7. Измените значение переменной `EVO_STREAMS_ACTIVE_LIST_MANAGER_RECREATE_ACTIVE_LISTS` обратно на `"false"`.

8. Если вы меняли интервал liveness-проб, восстановите интервал по умолчанию в конфигурации `evo.streams.active-list-manager`. Ее можно открыть, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit deployment evo.streams.active-list-manager
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

Активные списки: таблица активных списков не отображается

Проявление: в разделе **Ресурсы** → **Активные списки** не отображается таблица активных списков. Вместо нее отображается сообщение о внутренней ошибке.

В логах пода ClickHouse возникает ошибка, содержащая сообщение `MEMORY_LIMIT_EXCEEDED`, например:

```
2025.07.24 12:52:17.464721 [ 234 ] {} <Error> MergeTreeBackgroundExecutor: Exception
while executing background task {08280789-8eee-4778-8445-
009a607256e1::202507_536822_541220_179}: Code: 241. DB::Exception: Memory limit
(total) exceeded: would use 1.13 GiB (attempt to allocate chunk of 5047923 bytes),
maximum: 1.08 GiB. OvercommitTracker decision: Memory overcommit isn't used. Waiting
time or overcommit denominator are set to zero. (MEMORY_LIMIT_EXCEEDED), Stack trace
(when copying this message, always include the lines below):
```

Причина: лимиты памяти для подов ClickHouse недостаточны.

Решение: увеличьте лимиты оперативной памяти для подов ClickHouse. Для этого:

1. Получите список контроллеров StatefulSet, управляющих ресурсами, которые выделяются для подов ClickHouse:

   ```
   kubectl -n <namespace> get sts -l app.kubernetes.io/name=clickhouse
   ```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

2. Для каждого шарда увеличьте лимиты памяти:
   a. Откройте конфигурацию контроллера StatefulSet, соответствующего поду:

   ```
   kubectl -n <namespace> edit sts <shard_stateful_set>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<shard_stateful_set>` — имя Stateful Set.

   b. Увеличьте лимит оперативной памяти в значении ключа `resources.limits.memory`:

   Пример содержимого секции `resources.limits`

   ```
   resources:
     limits:
       cpu: 300m
       memory: 1288490188800m
   ```

   Примеры возможных значений лимита памяти:

   - В мегабайтах: `2560Mi`.
   - В гигабайтах: `3Gi`.

   Список доступных единиц измерения памяти приведен в документации Kubernetes.

   c. Сохраните изменения.

Поды шардов ClickHouse будут перезагружены для применения изменений, и доступ к разделу **Ресурсы → Активные списки** восстановится. Если этого не произошло, обратитесь в службу поддержки по адресу support@rvision.ru.

Активные списки: ошибка при импорте записей

Проявление: импорт записей активного списка завершается уведомлением с ошибкой.

Причина: под `evo.streams.active-list-manager` не успел провести преобразование и валидацию CSV-файла с записями активного списка и не ответил на проверку состояния сервиса (healthcheck). Из-за этого под был перезапущен системой Kubernetes, прежде чем отправил импортируемые записи в NATS.



Ошибка происходит при импорте активных списков с большим количеством записей.

Решение: проверьте логи пода `evo.streams.active-list-manager`. При успешной отправке импортируемых записей в NATS отобразится следующая запись в логах:

```
import active list records for active list with id: "<active_list_id>": return
```

Здесь:

- `<active_list_id>` — ID процесса импорта записей активного списка.

Если данная запись отсутствует в логах предыдущего запуска контейнера пода, это значит, что сервис не успел отправить данные в NATS. В таком случае увеличьте потребляемые ресурсы и интервал liveness-проб пода `evo.streams.active-list-manager`.

Активные списки: не отображаются новые записи после импорта

Проявление: импорт записей активного списка завершается уведомлением об успешном завершении, но при этом новые записи не отображаются в таблице.



Новые записи могут не сразу отобразиться после импорта. Время загрузки новых записей может занимать до 1 минуты.

Причина: произошел сбой при вставке записей активного списка в NATS после проверки на поде `evo.streams.active-list-manager`.

Решение: проверьте логи пода `evo.streams.active-list-manager`. Если при вставке записей NATS произошла ошибка, то в логах отобразится следующая запись:

```
import active list records for active list with id: "<active_list_id>": error upsert
records batch with keys "${promises.map(({ key }) => key)}"
```

Здесь:

- `<active_list_id>` — ID процесса импорта записей активного списка.

При возникновении данной ошибки получите логи пода `nats-main` и отправьте их в службу поддержки. Для этого:

1. Откройте логи контейнера `nats` на поде `nats-main`, выполнив команду:

   ```
   kubectl logs -n <namespace> nats-main-0 -c nats
   ```



У дополнительных реплик пода `nats-main` будет повышаться индекс в названии. Например: `nats-main-1`, `nats-main-2`.

Чтобы посмотреть их логи, замените `nats-main-0` в команде выше на необходимое название. Например:

```
kubectl logs -n <namespace> nats-main-1 -c nats
```



При необходимости вы можете добавить следующие операторы к команде:

- `-c <container_name>` — позволяет показать логи конкретного контейнера на поде.
- `--all-containers` — позволяет посмотреть логи всех контейнеров пода.
- `--previous` — позволяет показать логи предыдущего запуска контейнера.
- `--timestamps` — позволяет добавить временные метки к каждой строке логов.

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<container_name>` — название контейнера, запущенного на поде `nats-main`.

2. Отправьте полученные логи в службу поддержки по адресу support@rvision.ru.

Активные списки: отсутствует черный или белый список

Проявление: отсутствует системный активный список, используемый для исключений в коррелляторе.

Причина: системный активный список не был создан при установке системы.

Решение: создайте активный список на основе соответствующей схемы:

- Для черного списка — system-blacklist.
- Для белого списка — system-whitelist.

Активные списки: системные активные списки работают некорректно

Проявление: наблюдаются следующие проблемы с системными активными списками:

- Записи не удаляются после истечения срока жизни из поля `dueDate`.
- Не работает валидация VRL-кода для поля `values`.

Причина: система была обновлена с версии 2.7.0 без пересоздания системных активных списков. Начиная с версии 2.7.1 в активных списках изменена логика срока жизни записей, а также добавлен специальный тип данных для хранения VRL в записях.

Решение: пересоздайте системные активные списки. Для этого:

1. Удостоверьтесь, что системные активные списки не используются ни в одном из конвейеров. В противном случае удалите их из конвейеров.
2. Сбросьте тип системных активных списков на **Пользовательский**, запустив в СУБД PostgreSQL следующий скрипт:

```
-- Сброс типа для белого и черного списков.
UPDATE active_list.active_list
SET is_system = false
WHERE active_list_name IN ('system-whitelist', 'system-blacklist');

-- Сброс типа для схем белого и черного списков.
UPDATE expertise.expertise
SET object_type = 'custom'
WHERE name IN ('system-whitelist', 'system-blacklist');

-- Удаление скрипта добавления исключений.
DELETE FROM evo_streams_script.executed_script WHERE name =
'ScriptAddExceptionActiveList';
```

3. Удалите системные активные списки:
   a. Перейдите в раздел **Ресурсы → Активные списки**.
   b. Найдите активные списки system-whitelist и system-blacklist.
   c. Удостоверьтесь, что найденные активные списки имеют тип **Пользовательский**. В противном случае повторите шаг 2 текущей инструкции.
   d. Удалите активные списки system-whitelist и system-blacklist.
4. Удалите схемы системных активных списков:
   a. Перейдите в раздел **Экспертиза**.
   b. Найдите схемы активных списков system-whitelist и system-blacklist.
   c. Удостоверьтесь, что найденные схемы активных списков имеют тип создания **Пользовательский**. В противном случае повторите шаг 2 текущей инструкции.
   d. Удалите схемы активных списков system-whitelist и system-blacklist.
5. Подключитесь к терминалу master-узла кластера.
6. Перезапустите сервис `evo.streams.add-exception-active-lists-add-exception` с помощью следующей команды:

   ```
   /opt/r-vision/common/bin/evoctl instance update -n <namespace> -m evo.streams -g
   "Streams applications" -s evo.streams.add-exception-active-lists --no-ansible
   --verbose
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.

Активные списки: ошибка при импорте записей в системный активный список

Проявление: импорт записей в системный активный список завершается уведомлением с ошибкой.

Причина: в импортируемом CSV-файле срок жизни задан через устаревшее поле `dueDate`. Начиная с версии 2.7.1 срок жизни записей в активных списках задается с помощью поля `ttl` типа `integer`

вместо поля `dueDate`.

Решение:

1. Отредактируйте импортируемый CSV-файл, внеся в него следующие изменения:

   a. Замените поле `dueDate` на `ttl` и укажите актуальные сроки жизни для каждой записи в секундах. Поле `ttl` может принимать только целочисленные значения.

      Если требуется сделать записи бессрочными, задайте значение `0`.

   b. Удалите поле `createdAt`.

2. Повторите попытку импорта с измененным CSV-файлом.

Экспертиза: не удается импортировать элементы экспертизы

Проявление: импорт некоторых элементов экспертизы завершается с ошибкой `Worker terminated`.

Причина: сервис `evo.streams.pipeline-verification` обладает недостаточными ресурсами для полной обработки импорта.

Решение: увеличьте ресурсы и таймауты для сервиса `evo.streams.pipeline-verification`. Для этого:

Через командную строку

1. Подключитесь к кластеру.
2. Увеличьте значения таймаутов в ConfigMap `evo.streams-pipeline-verification-env`, выполнив команду:

   ```
   kubectl patch configmap -n <namespace> evo.streams-pipeline-verification-env
   -p '{"data": {"EVO_STREAMS_PIPELINE_VERIFICATION_ROBJECT_RUN_TEST_TIMEOUT":
   "<test_timeout>",
   "EVO_STREAMS_PIPELINE_VERIFICATION_ROBJECT_CHECK_SCHEMA_TIMEOUT":
   "<config_check_timeout>"}}'
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<test_timeout>` — таймаут для выполнения тестов элемента экспертизы в миллисекундах. Значение по умолчанию: `60000`.
   - `<config_check_timeout>` — таймаут для проверки конфигурации элемента экспертизы в миллисекундах. Значение по умолчанию: `60000`.

3. Откройте конфигурацию контроллера Deployment `evo.streams.pipeline-verification`, выполнив команду:

   ```
   kubectl edit -n <namespace> deployment evo.streams.pipeline-verification
   ```

Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.

4. Увеличьте значения лимитов процессора (`cpu`) и оперативной памяти (`memory`) в секции `spec.template.spec.containers.resources`.

   Пример секции `resources` в конфигурации контроллера

   ```
   resources:
     limits:
       cpu: 500m
       memory: 300Mi
     requests:
       cpu: 25m
       memory: 100Mi
   ```

5. Сохраните обновленную конфигурацию. Под контроллера Deployment `evo.streams.pipeline-verification` будет автоматически перезапущен.

Через Lens

1. Подключитесь к кластеру.
2. Увеличьте значения таймаутов для сервиса `evo.streams.pipeline-verification`. Для этого:
   a. В выпадающем меню свойств кластера выберите раздел **Config** → **Config Maps**.
   b. Выберите ConfigMap `evo.streams-pipeline-verification-env`. Откроется панель конфигурации ConfigMap.
   c. Увеличьте значения следующих переменных среды:
      - `EVO_STREAMS_PIPELINE_VERIFICATION_ROBJECT_RUN_TEST_TIMEOUT` — таймаут для выполнения тестов элемента экспертизы в миллисекундах. Значение по умолчанию: `60000`.
      - `EVO_STREAMS_PIPELINE_VERIFICATION_ROBJECT_CHECK_SCHEMA_TIMEOUT` — таймаут для проверки конфигурации элемента экспертизы в миллисекундах. Значение по умолчанию: `60000`.
   d. Сохраните внесенные изменения.
3. Увеличьте лимиты ресурсов контроллера Deployment `evo.streams.pipeline-verification`. Для этого:
   a. В выпадающем меню свойств кластера выберите раздел **Workloads** → **Deployments**.
   b. Выберите контроллер Deployment `evo.streams.pipeline-verification` в пространстве имен, в котором установлена система. Откроется панель конфигурации контроллера.
   c. Нажмите на кнопку **Edit** в верхней части правой панели. Откроется вкладка терминала с конфигурацией контроллера.
   d. Увеличьте значения лимитов процессора (`cpu`) и оперативной памяти (`memory`) в секции `spec.template.spec.containers.resources`.

Пример секции `resources` в конфигурации контроллера

```
resources:
  limits:
    cpu: 500m
    memory: 300Mi
  requests:
    cpu: 25m
    memory: 100Mi
```

e. Сохраните обновленную конфигурацию. Под контроллера Deployment `evo.streams.pipeline-verification` будет автоматически перезапущен.

Шлюзы: шлюз регулярно перезапускается

Проявление: шлюз в системе регулярно отключается и включается заново.

Причина: не хватает оперативной памяти для работы шлюза.

Решение: увеличьте максимальный размер выделенной памяти для шлюза. Для этого:

1. Получите идентификатор шлюза из его карточки.
2. Подключитесь к терминалу master-узла кластера.
3. Получите идентификатор контроллера Deployment нужного шлюза, выполнив следующую команду:

   ```
   kubectl -n <namespace> get deployment | grep <gateway_id>
   ```

   Здесь:

   - `<namespace>` — название пространства имен, в котором установлена система.
   - `<gateway_id>` — идентификатор шлюза, полученный из его карточки.

   ▼ *Пример вывода имени контроллера шлюза*

   ```
   wl.5db08c79-b5f3-4f07-a7b7-49e3b2ac6202-gateway          1/1     1            1
   26h
   ```

   В примере выше имя контроллера шлюза — `wl.5db08c79-b5f3-4f07-a7b7-49e3b2ac6202-gateway`.

4. Откройте конфигурацию контроллера шлюза, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit deployment <deployment_id>
   ```

   Здесь:

- `<namespace>` — название пространства имен, в котором установлена система.
- `<deployment_id>` — идентификатор контроллера, полученный на предыдущем шаге.

5. В отобразившейся конфигурации увеличьте лимит оперативной памяти в переменной окружения `GOMEMLIMIT`:

```
- name: GOMEMLIMIT
  value: 1536MiB
```



Рекомендуется устанавливать лимит в 75% от общего объема оперативной памяти, выделенной для кластера Kubernetes.

6. Сохраните изменения в конфигурации контроллера.



При отключении и повторном включении шлюза настройки его ресурсов и переменных окружения сбрасываются до значений по умолчанию. После перезапуска шлюза необходимо повторить вышеприведенные действия.

Агенты: приложение агента в macOS неожиданно завершает работу

Проявление: система на хосте с macOS выдает ошибку о том, что приложение rpoint-endpoint-security неожиданно завершило работу.

Причина: приложение rpoint-endpoint-security не обладает разрешением на полный доступ к диску, необходимым для его работы.

Решение: выдайте приложению rpoint-endpoint-security полный доступ к диску. Для этого:

1. Откройте меню Apple и выберите опцию **Настройки системы**.
2. Перейдите в раздел **Конфиденциальность и безопасность** → **Полный доступ к диску**.
3. Выполните одно из следующих действий:
   - Если приложение rpoint-endpoint-security уже есть в списке, то установите переключатель справа от него в активное положение.
   - Если приложения rpoint-endpoint-security нет в списке:
     a. Нажмите на кнопку ＋ и выберите **Программы** на боковой панели.
     b. Найдите приложение rpoint-endpoint-security.app, выберите его и нажмите **Открыть**.
     c. Убедитесь, что приложение появилось в списке и переключатель справа от него активен.

Nginx: не загружаются файлы больших размеров

Проявление: не удается загрузить файл размером больше 1 МБ через веб-интерфейс системы. В инструментах разработчика в браузере отображается ошибка запроса с кодом HTTP 413.

Причина: превышен лимит на размер загружаемых файлов (по умолчанию 1 МБ).

Решение: увеличьте максимальный размер загружаемых файлов. Инструкция приведена в разделе

Настройка ограничений размера загружаемых файлов.

Grafana: нет доступа к сервису

Проявление: не удается получить доступ к сервису Grafana по пути `/grafana`.

Причина: форма авторизации в сервисе Grafana по умолчанию скрыта из веб-интерфейса системы.

Решение: подключитесь к сервису Grafana локально или предоставьте доступ к сервису для всех пользователей. Инструкция приведена в разделе Вход в веб-интерфейс Grafana.



Предоставление доступа к Grafana для всех пользователей повышает риск взаимодействия потенциального нарушителя с системой. В целях безопасности рекомендуется подключаться к сервису Grafana локально.

Grafana: не поступают системные метрики в сервис

Проявление: не поступают системные метрики с worker-узлов в сервис Grafana. Некоторые дашборды не отображают данные с кластера.

Причина: Grafana не может получить доступ к сервису Node Exporter по стандартному порту 9100, так как порт занят другим сервисом.

Решение: необходимо изменить порт для сервиса Node Exporter. Вы можете сделать это следующими способами:

- Через обновление helm-чарта.
- Через редактирование существующих ресурсов кластера.



При обновлении системы порт сервиса Node Exporter сбросится на значение по умолчанию — 9100. Чтобы при каждом обновлении системы не менять порт сервиса Node Exporter, рекомендуется перенести на другой свободный порт сервис, который занимает порт 9100.

Обновление helm-чарта

1. Подключитесь к серверу, на котором производилась установка системы.
2. Откройте для редактирования файл `custom_values.yaml`, выполнив команду:

   ```
   sudo nano /opt/r-vision/infra/helm/kube-prometheus-stack/custom_values.yaml
   ```

3. Добавьте в файл `custom_values.yaml` следующие строки:

   ```
   prometheus-node-exporter:
     service:
       port: 9101
       targetPort: 9101
   ```

4. Сохраните изменения.
5. Обновите helm-чарт, выполнив команду:

```
helm upgrade -i kube-prometheus-stack oci://harbor.rvision.pro/charts/kube-prometheus-stack --version 61.7.3 -f helm/kube-prometheus-stack/custom_values.yaml --create-namespace -n kube-prometheus-stack --set grafana.ingress.hosts="{$INGRESS_HOSTS}"
```

Редактирование существующих ресурсов кластера

1. Подключитесь к кластеру.
2. Откройте конфигурацию контроллера DaemonSet `kube-prometheus-stack`, выполнив команду:

```
kubectl edit daemonsets.apps -n kube-prometheus-stack kube-prometheus-stack-prometheus-node-exporter
```

3. Замените все порты 9100 на 9101.
4. Замените порты Service `kube-prometheus-stack-prometheus-node-exporter` с 9100 на 9101, выполнив команду:

```
kubectl patch service -n kube-prometheus-stack kube-prometheus-stack-prometheus-node-exporter --type='merge' -p '{"spec":{"ports":[{"name":"http-metrics","port":9101,"targetPort":9101}]}}'
```

Глоссарий

Управление системой

Модуль

Минимальная функциональная единица системы в рамках платформы R-Vision EVO.

Основной экземпляр системы

Экземпляр системы, который содержит все компоненты продукта, включая графический интерфейс. Устанавливается в центральном кластере.

Подробнее: *Архитектура системы*

Сателлит

Удаленный экземпляр системы, который может работать только с запущенными в нем компонентами. Сателлит не обладает полным спектром функциональности продукта, в том числе для него не предусмотрен графический интерфейс. Устанавливается в кластере сателлита.

Подробнее: *Архитектура системы*

Событие аудита

Внутрисистемное событие, которое создается при выполнении определенных действий, связанных с безопасностью системы. Используется для обеспечения целостности системы.

Подробнее: Аудит системных событий

Управление доступом

LDAP-соединение

Настройка для подключения системы к службам каталогов, таким как Microsoft Active Directory, ALD Pro, FreeIPA, OpenLDAP.

Подробнее: LDAP-соединения

Администратор

Стандартная роль, предназначенная для настройки и поддержки инфраструктуры системы. Обладает всеми возможными разрешениями в системе, имеет полный контроль над всеми тенантами системы.

Подробнее: Стандартные роли

Бот

Учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы.

Подробнее: Пользователи

Группа пользователей

Объединение учетных записей для назначения общих прав управления системой.

Подробнее: Группы пользователей

Доменная группа

Группа пользователей, связанная с каталогом LDAP. Ее состав и данные учетных записей автоматически синхронизируются с указанным LDAP-каталогом.

Подробнее: Группы пользователей

Дочерний тенант

Изолированная часть основной инфраструктуры системы, которая содержит ресурсы и данные, выделенные из основного тенанта. Каждый дочерний тенант работает независимо, обеспечивая доступ и управление только собственными ресурсами и данными.

Подробнее: Тенанты

Инженер

Стандартная роль, предназначенная для настройки операционных процессов системы, разработки решений для оптимизации этих процессов и их внедрения. Действия инженера ограничены рамками определенных тенантов.

Подробнее: Стандартные роли

Локальная группа

Группа пользователей, куда могут быть включены любые учетные записи в системе. Ее состав регулируется вручную через веб-интерфейс системы.

Подробнее: Группы пользователей

Локальный пользователь

Учетная запись для доступа и работы пользователей в системе.

Подробнее: Пользователи

Основной тенант

Тенант, устанавливаемый в системе по умолчанию. Имеет доступ к данным всех дочерних тенантов. Основной тенант нельзя удалить или изменить.

Подробнее: Тенанты

Пользователь

1. Средство идентификации и аутентификации лица в системе, включающее в себя уникальные учетные данные, такие как логин и пароль, а также набор присвоенных ролей.

   Подробнее: Пользователи

2. Стандартная роль, предназначенная для работы с данными системы в рамках определенных тенантов. Имеет ограниченный доступ к ресурсам системы.

Подробнее: Стандартные роли

Пользователь из домена

Учетная запись, созданная через интеграцию с LDAP-сервером и предназначенная для управления пользователями из внешнего каталога.

Подробнее: Пользователи

Разрешение

Право пользователя на выполнение определенного действия или наличие у него доступа к определенной функциональности системы.

Подробнее: Роли

Роль

Набор разрешений пользователя в системе.

Подробнее: Роли

Стандартная роль

Роль с предопределенным набором разрешений. Стандартную роль нельзя изменить или удалить.

Подробнее: Стандартные роли

Суперадминистратор

Статус пользователя, предоставляющий ему высший уровень доступа к функциональности системы. Суперадминистратор может выполнять любые операции в системе, включая создание и управление другими ролями и пользователями.

Подробнее: Роли

Тенант

Изолированное виртуальное пространство в многопользовательской системе, где данные, настройки и доступы одних частей системы отделены от других.

Подробнее: Тенанты

Управление пространствами

Кластер сателлита

Дополнительный удаленный кластер Kubernetes, в котором разворачивается сателлит.

Подробнее: Управление пространствами

Пространство

Логическая среда в рамках кластера Kubernetes, предназначенная для развертывания и запуска продуктовых сервисов.

Подробнее: Работа с пространствами

Сервис

Программный компонент, который может разворачиваться в продукте. Сервис создается и запускается в рамках пространства.

Подробнее: Работа с сервисами

Центральный кластер

Кластер Kubernetes, в котором разворачивается основной экземпляр системы.

Подробнее: Управление пространствами

Управление событиями ИБ

VRL-трансформация

Элемент конвейера, который позволяет обрабатывать и преобразовывать события с применением языка VRL.

Подробнее: Работа с VRL-трансформациями

Агрегатор

Элемент конвейера, который использует правила агрегации для объединения и анализа событий.

Подробнее: Работа с агрегаторами

Активный список

Двумерный массив данных для накопления информации, полученной из определенных полей событий, поступающих на коррелятор.

Подробнее: Работа с активными списками

База данных событий

База данных для хранения событий в системе.

Подробнее: Работа с базами данных событий

Вынесенный коллектор

Коллектор, расположенный в сателлите.

Подробнее: Сбор событий из вынесенного коллектора

Дашборд

Специальная панель, предназначенная для отображения информации об основных метриках системы, оповещениях, событиях и активных списках в графическом виде. Панель дашборда состоит из виджетов.

Подробнее: Работа с дашбордами

Виджет

Элемент дашборда, предназначенный для представления информации об основных метриках системы, оповещениях, событиях и активных списках в графическом виде.

Подробнее: Работа с виджетами

Драйвер БД

Компонент системы, который используется для подключения внешних баз данных и дальнейшей работы с ними.

Подробнее: Работа с драйверами БД

Интеграция

Механизм взаимодействия системы с внешними системами для отправки оповещений.

Подробнее: Работа с интеграциями

Коллектор

Компонент системы, который обеспечивает сбор событий из источников, их обработку и дальнейшую отправку в хранилище и/или во внешние системы.

Подробнее: Работа с коллекторами

Конвейер

Компонент коллектора, представляющий собой упорядоченную последовательность взаимосвязанных элементов, выполняющих различные этапы обработки событий.

Подробнее: Работа с конвейерами

Конечная точка

Элемент конвейера, который отвечает за отправку событий.

Подробнее: Работа с конечными точками

Коррелятор

Элемент конвейера, который представляет собой комплекс правил корреляции.

Подробнее: Работа с корреляторами

Маршрутизатор

Элемент конвейера, который позволяет на основе заданных фильтров (маршрутов) направлять события на разные этапы обработки в зависимости от заданных условий.

Подробнее: Работа с маршрутизаторами

Модель события

Компонент системы, определяющий формат хранилища событий: имена, типы полей и типы данных в полях.

Подробнее: Работа с моделями событий

Нормализатор

Элемент конвейера, который представляет собой комплекс правил нормализации.

Подробнее: Работа с нормализаторами

Обогащение

Компонент коллектора, который позволяет использовать таблицы обогащения в корреляторах и/или нормализаторах конвейеров.

Подробнее: Работа с обогащениями

Оповещение

Системное сообщение, генерируемое на основе логики правил сегментации.

Подробнее: Работа с оповещениями

Отчет

Компонент системы, предназначенный для сбора, анализа и представления данных.

Подробнее: Работа с отчетами

Ресурс

Компонент системы, который обеспечивает настройку ее конкретных функций и управление ими.

Подробнее: *Ресурсы*

Сбор метрик

Элемент конвейера, который собирает метрики различных элементов конвейеров для их дальнейшей обработки и анализа.

Подробнее: Работа со сбором метрик

Секрет

Компонент системы, который обеспечивает безопасное хранение чувствительных данных, таких как учетные данные, токены, строки подключения, сертификаты, приватные ключи.

Подробнее: Работа с секретами

Сервис оповещений

Элемент конвейера, который позволяет настраивать связи правил сегментации и правил корреляции для распределения корреляционных событий по оповещениям.

Подробнее: Работа с сервисами оповещений

Событие

Идентифицированный переход системы в состояние, которое указывает на возможное нарушение информационной безопасности.

Подробнее: Работа с разделом Поиск

Точка входа

Элемент конвейера, который отвечает за доставку в систему исходных событий.

Подробнее: Работа с точками входа

Универсальная модель события

Системная модель события, предназначенная для хранения информации о событиях в едином, универсальном, виде.

Подробнее: Универсальная модель события

Универсальная модель события 2.0

Системная модель события, представляющая собой развитие предыдущей модели события.

Подробнее: Универсальная модель события 2.0

Фильтр

Элемент конвейера, который позволяет сократить объем обрабатываемых событий путем их отсеивания на основе условий на языке VRL.

Подробнее: Работа с фильтрами

Хранилище событий

Таблица в базе данных событий. Используется для хранения событий, поступивших в систему и обработанных коллектором.

Подробнее: Работа с хранилищами событий

Шаблон отчета

Инструмент для определения структуры и периодичности публикации отчета. Позволяет пользователю выбирать аналитические данные для включения в отчет и поддерживает функции автоматического создания и отправки отчета по расписанию.

Подробнее: Работа с отчетами

Шина

Компонент коллектора, который позволяет направлять потоки событий между конвейерами.

Подробнее: Работа с шинами

Управление агентами

WatchDog

Параметр конфигурации группы агентов, который запускает сервис для отслеживания процессов на агенте.

Подробнее: Настройка конфигурации группы агентов

Агент

Бинарный исполняемый файл на конечных устройствах. Агент позволяет производить сбор информации об уязвимостях и угрозах.

Подробнее: Работа с агентами

Группа агентов

Объединение агентов, упрощающее выполнение массовых операций над ними. Для группы доступна настройка конфигурации, которая позволяет задавать политики отслеживания различных действий на ее агентах.

Подробнее: Работа с группами агентов

Инвентарная информация

Агрегированные данные о хосте, собираемые агентом.

Подробнее: Сбор инвентарной информации

Конфигурация группы агентов

Набор правил и настроек, которые регламентируют поведение группы агентов.

Подробнее: Настройка конфигурации группы агентов

Менеджер агентов

Компонент, который разворачивается в центральном кластере и предназначен для управления агентами, группами агентов и шлюзами.

Подробнее: *Агенты*

Шлюз

Промежуточный уровень в сети между агентами, установленными на конечных устройствах, и центральным кластером. Шлюзы предназначены для управления потоками данных между агентами и центральным кластером.

Подробнее: Работа со шлюзами

Управление экспертизой

Глобальная функция

Элемент экспертизы, задающий переиспользуемый блок кода на языке VRL, который можно применять при создании правил корреляции и нормализации.

Подробнее: Глобальные функции

Конструктор правил корреляции

Компонент системы, позволяющий визуально строить правила корреляции в интерактивном режиме без необходимости ручной настройки его RObject-конфигурации.

Подробнее: Построение правила корреляции в конструкторе

Правило агрегации

Элемент экспертизы, который описывает логику объединения и анализа групп событий безопасности.

Подробнее: Правила агрегации

Правило корреляции

Элемент экспертизы, который содержит конфигурацию для автоматизированного распознания определенных последовательностей обрабатываемых событий ИБ.

Подробнее: Правила корреляции

Правило нормализации

Элемент экспертизы, который обрабатывает и преобразовывает поступающие события согласно условиям и коду нормализации на языке VRL.

Подробнее: Правила нормализации

Правило сегментации

Элемент экспертизы, который определяет логику для классификации и обработки корреляционных событий для генерации оповещений.

Подробнее: Правила сегментации

Сабтехника

Вариант действий злоумышленника в рамках техники.

Подробнее: Оценка покрытия техник MITRE ATT&CK правилами экспертизы

Схема активного списка

Элемент экспертизы, который задает конфигурацию и параметры для создания активного списка и управления им.

Подробнее: Схемы активных списков

Схема дашборда

Элемент экспертизы, который задает конфигурацию и параметры для создания дашборда.

Подробнее: Схемы дашбордов

Схема шаблона отчета

Элемент экспертизы, который задает конфигурацию и параметры для создания шаблона отчета.

Подробнее: Схемы шаблонов отчетов

Таблица обогащения

Элемент экспертизы, который используется для дополнения и уточнения информации по событиям безопасности.

Подробнее: Таблицы обогащения

Тактика

Категория действий злоумышленника, описывающая его цель на определенном этапе атаки.

Подробнее: Оценка покрытия техник MITRE ATT&CK правилами экспертизы

Техника

Вид приемов, которые использует злоумышленник для достижения тактической цели.

Подробнее: Оценка покрытия техник MITRE ATT&CK правилами экспертизы

Шаблон поиска

Элемент экспертизы, который определяет параметры и условия для выполнения запросов к данным в системе в разделе **Поиск** и виджетах дашбордов.

Подробнее: Шаблоны поиска

Экспертиза

Модуль системы, позволяющий создавать объекты на языке RObject для специализированных задач.

Подробнее: Работа с элементами экспертизы

Элемент экспертизы

Объект, создаваемый в разделе **Экспертиза**.

Подробнее: Работа с элементами экспертизы