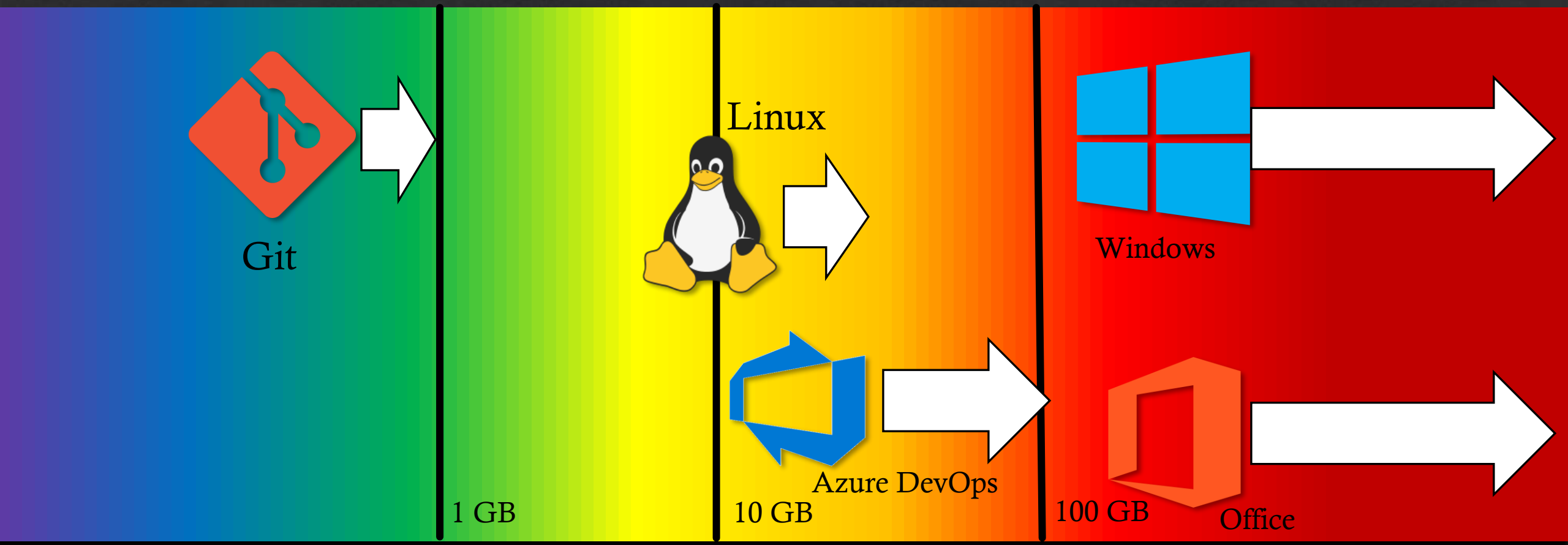
Git at Scale for Everyone

D. Stolee, Git Ecosystem Team, Microsoft

Twitter: @stolee GitHub: @derrickstolee

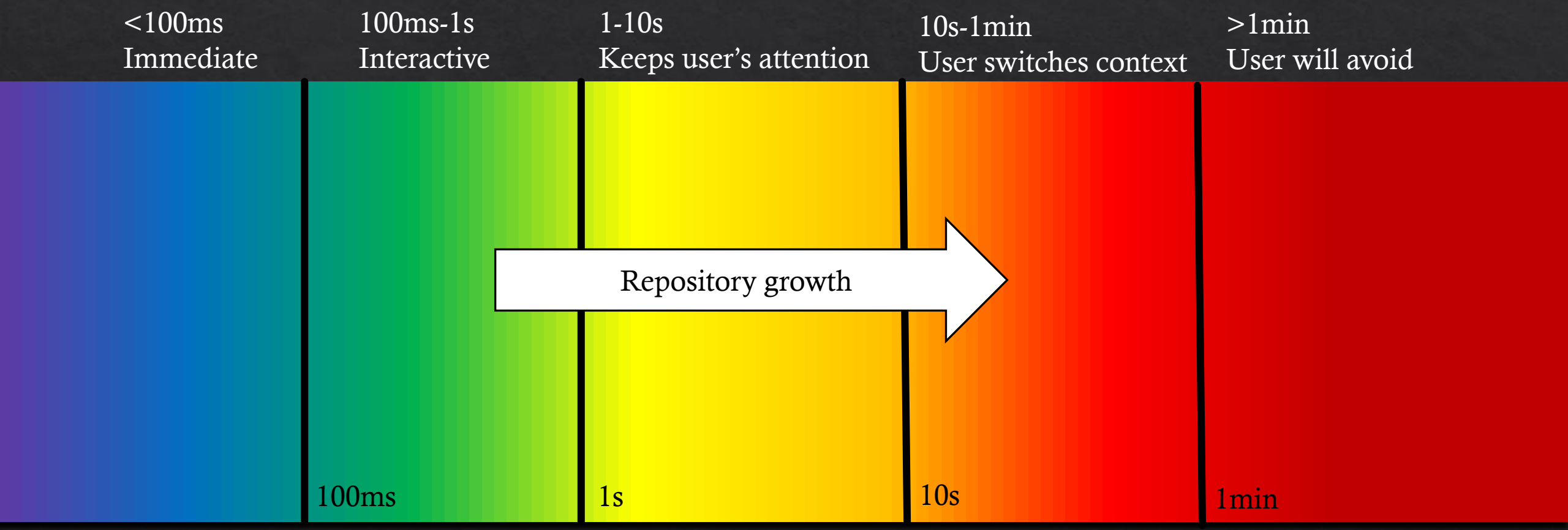
https://stolee.dev/docs/git-merge-2020.pdf

Spectrum of Scale



Pack-file size for initial clone

Spectrum of Perceived Performance

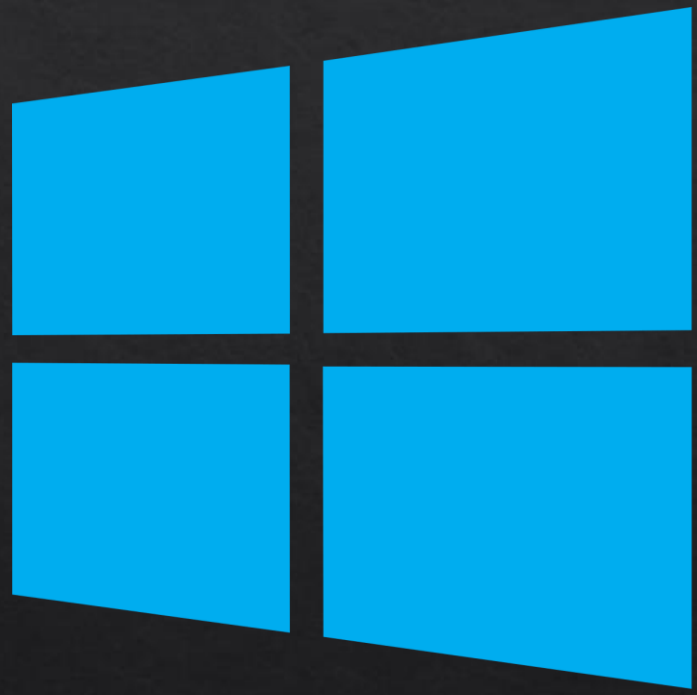


Git command time

http://gunshowcomic.com/648

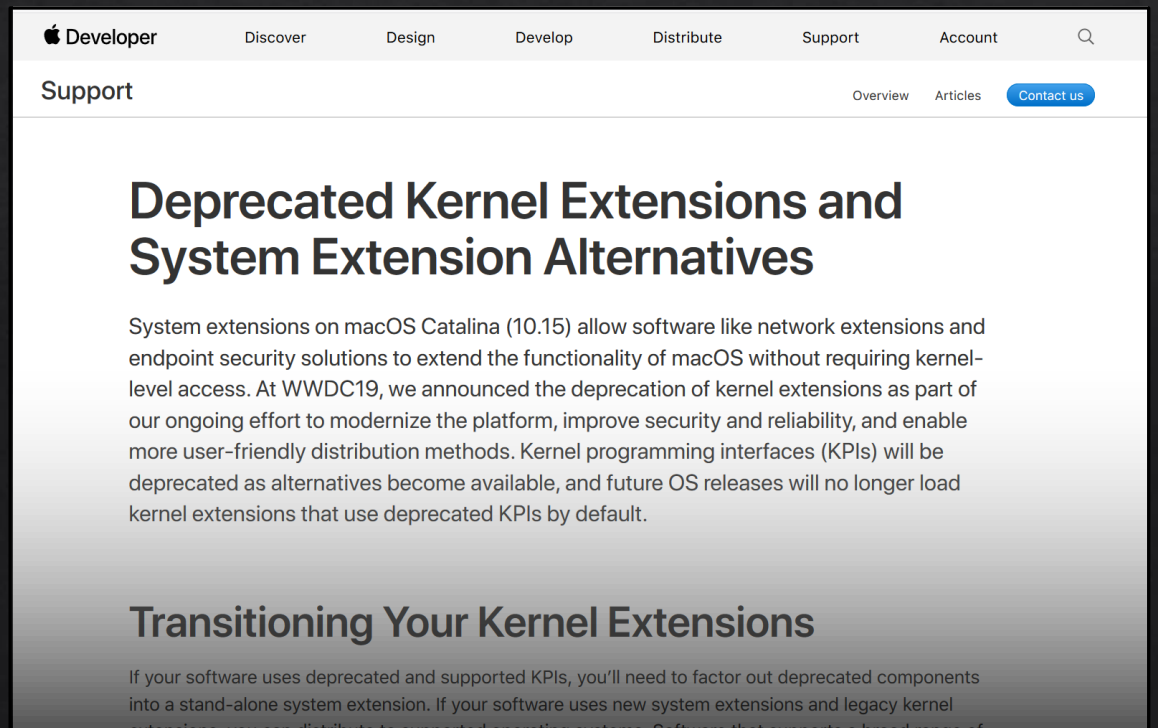
Success Story: Microsoft Windows

- Largest Git repository
- VFS for Git enabled using it *at all*
  - Virtualized filesystem “fakes” working directory updates
- Measuring real user interactions showed need for Git performance improvements
- Delivered most improvements as contributions to core Git client



Next Milestone: Microsoft Office

- Similar size and shape to Windows OS repo
- Hosted on Azure Repos
- Client **must** work on Windows & macOS



https://github.com/microsoft/scalar

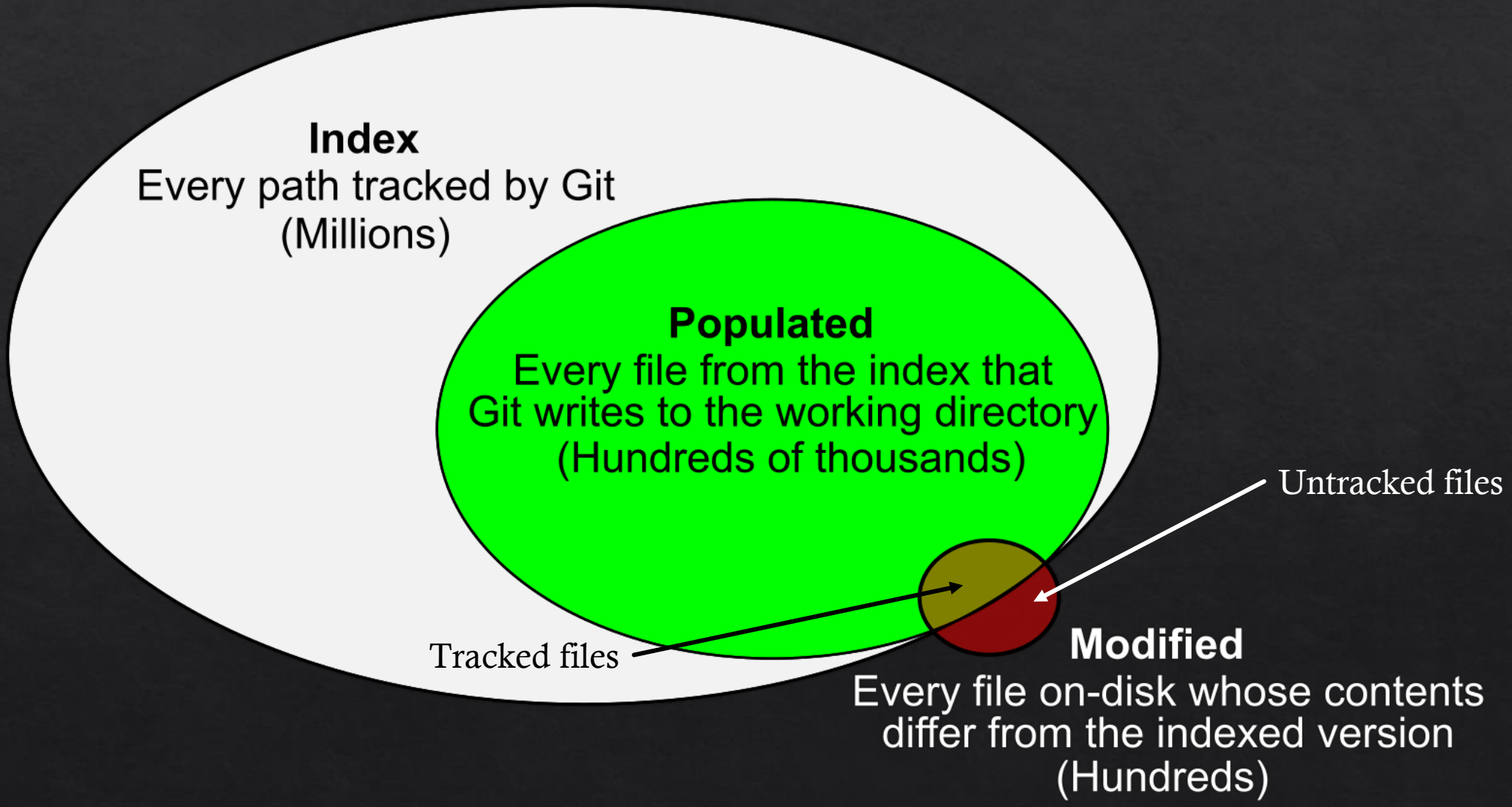
Lessons for Git at Scale

Lesson 1: Focus on the files that matter

Lesson 2: Reduce object transfer

Lesson 3: Don’t wait for expensive operations

Lesson 1: Focus on the files that matter



Reduce Populated Size: Sparse-checkout

To control the number of files in your working directory, run

```
git sparse-checkout init --cone
```

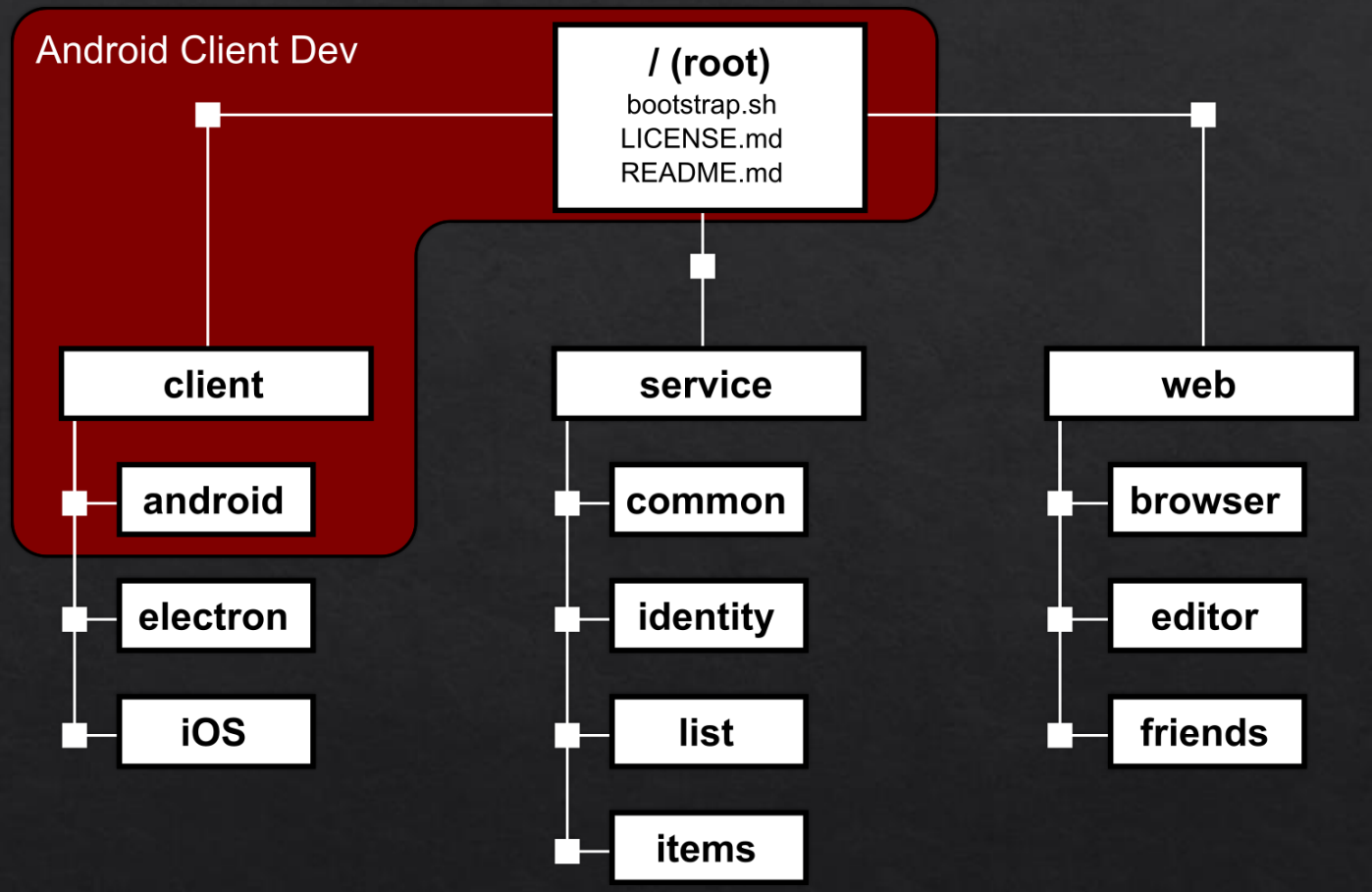
initializes sparse-checkout in “cone mode”. This starts with only the files at root.

Included paths can be expanded using

```
git sparse-checkout set <dir1> <dir2> ...
```

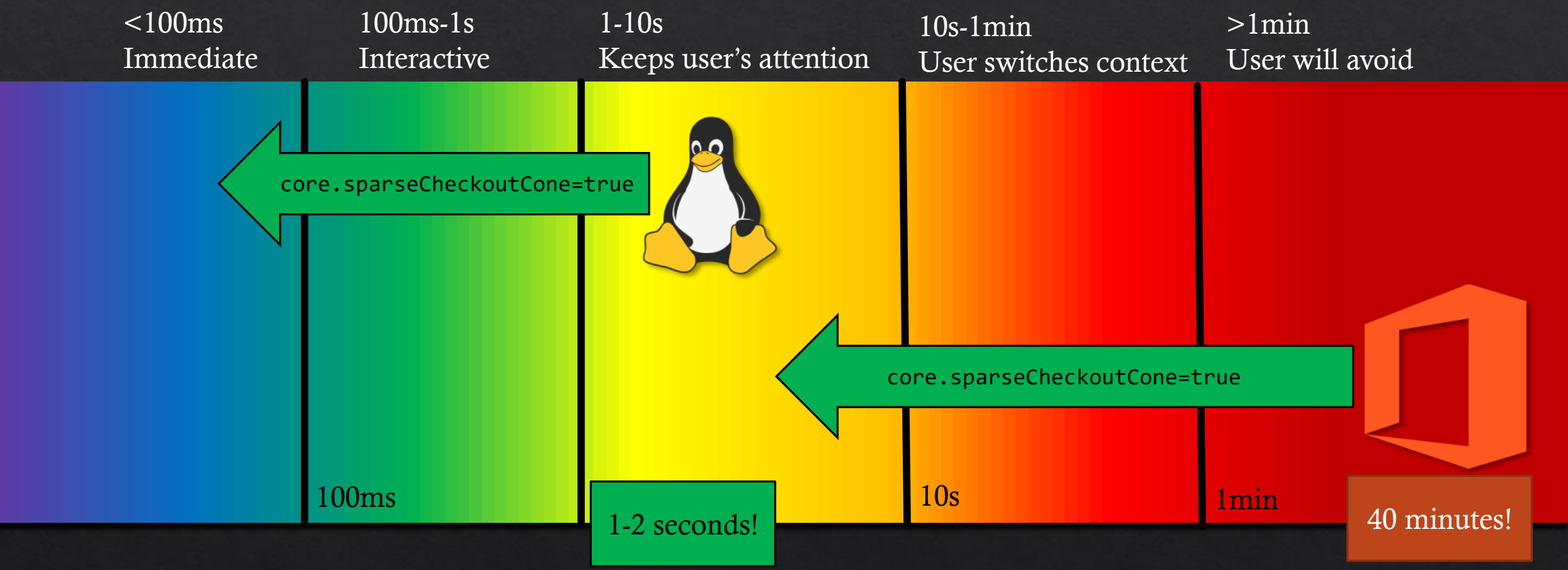
In this example, we use:

```
git sparse-checkout set client/android
```



https://github.blog/2020-01-17-bring-your-monorepo-down-to-size-with-sparse-checkout/

Spectrum of Perceived Performance

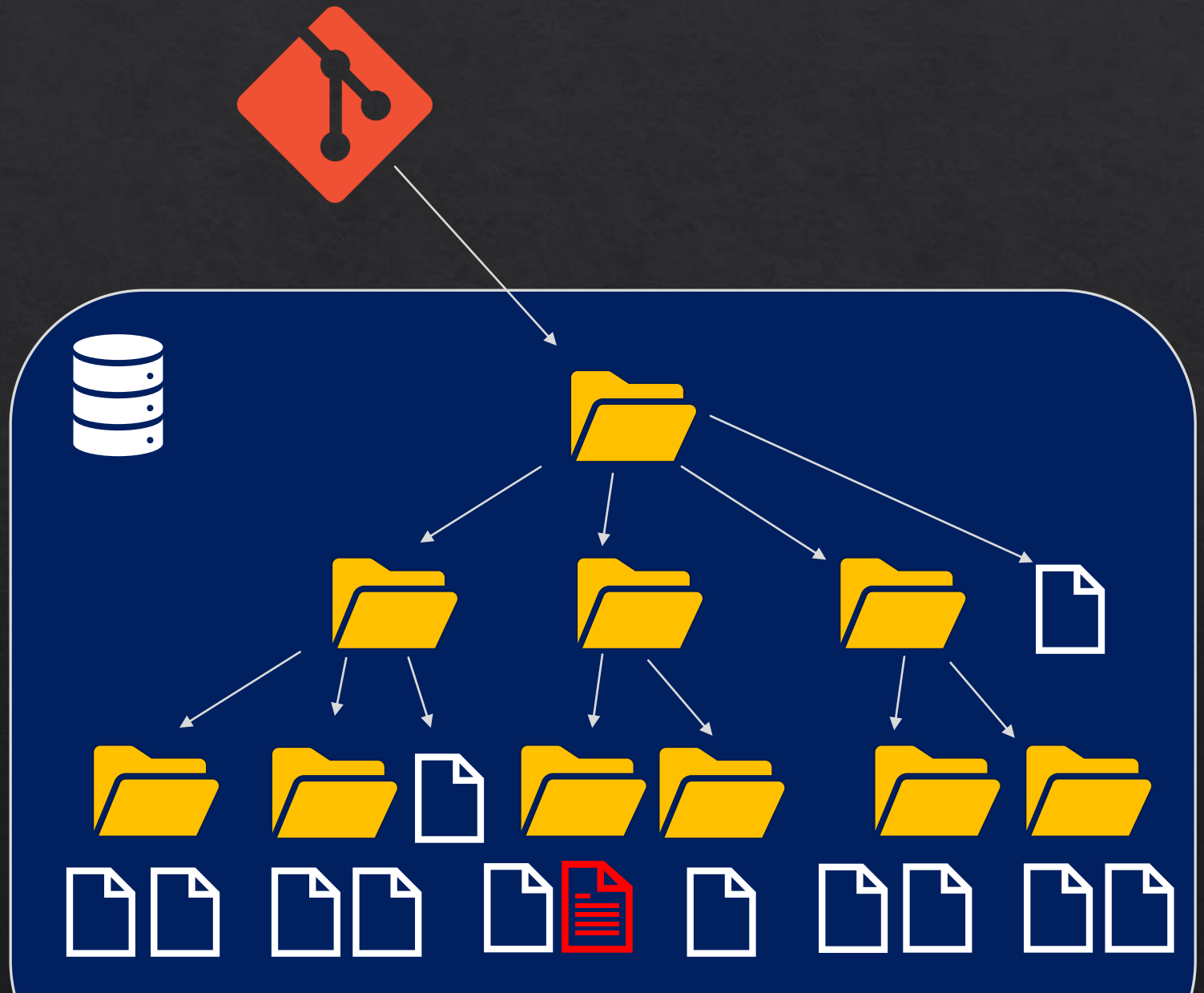


Time to update sparse-checkout definition

Finding Modified Files with Filesystem Monitor

Commands like `git status` or `git add` need to know which files were modified since the last checkout.

This usually results in scanning directories.



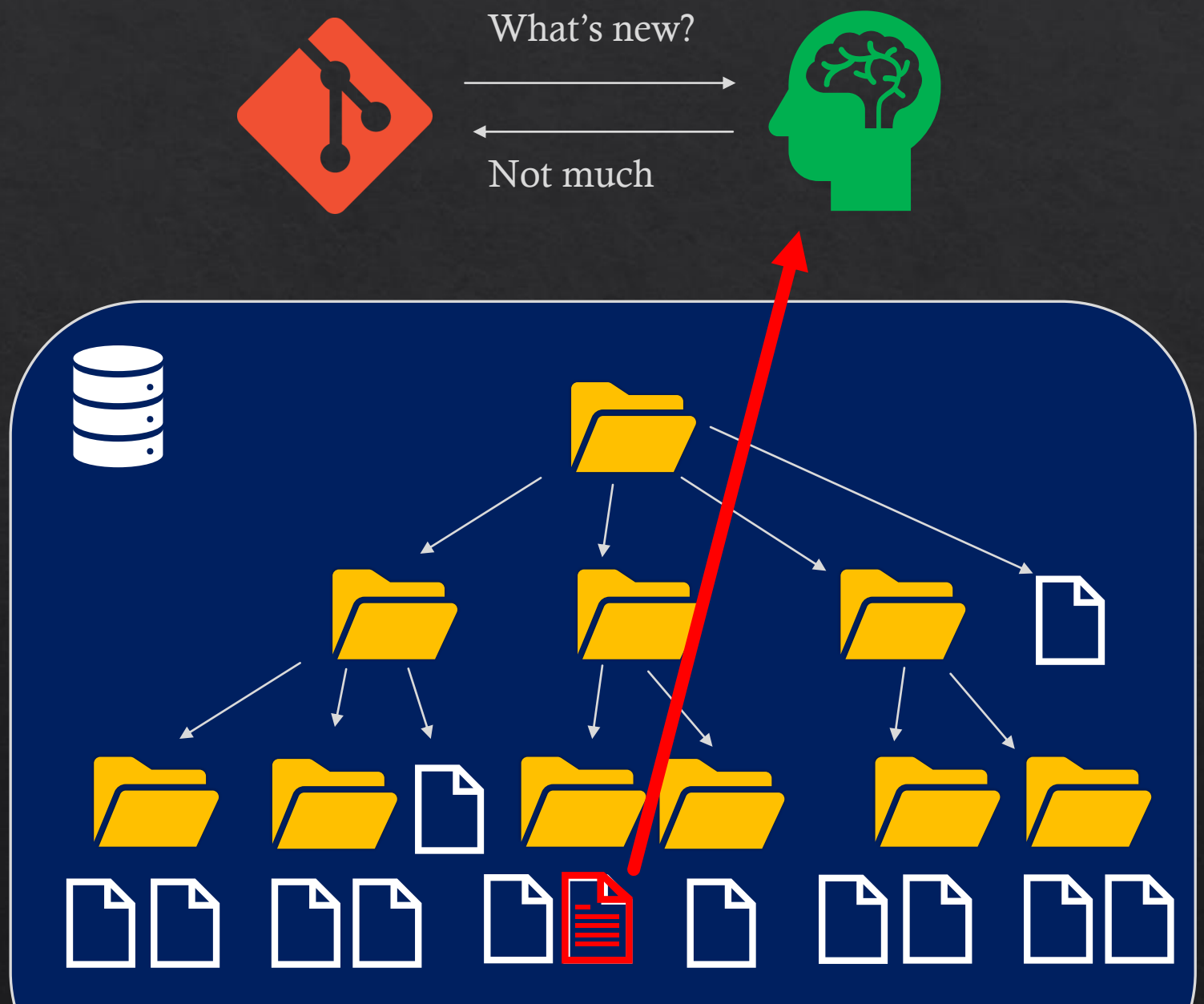
Finding Modified Files with Filesystem Monitor

Commands like `git status` or `git add` need to know which files were modified since the last checkout.

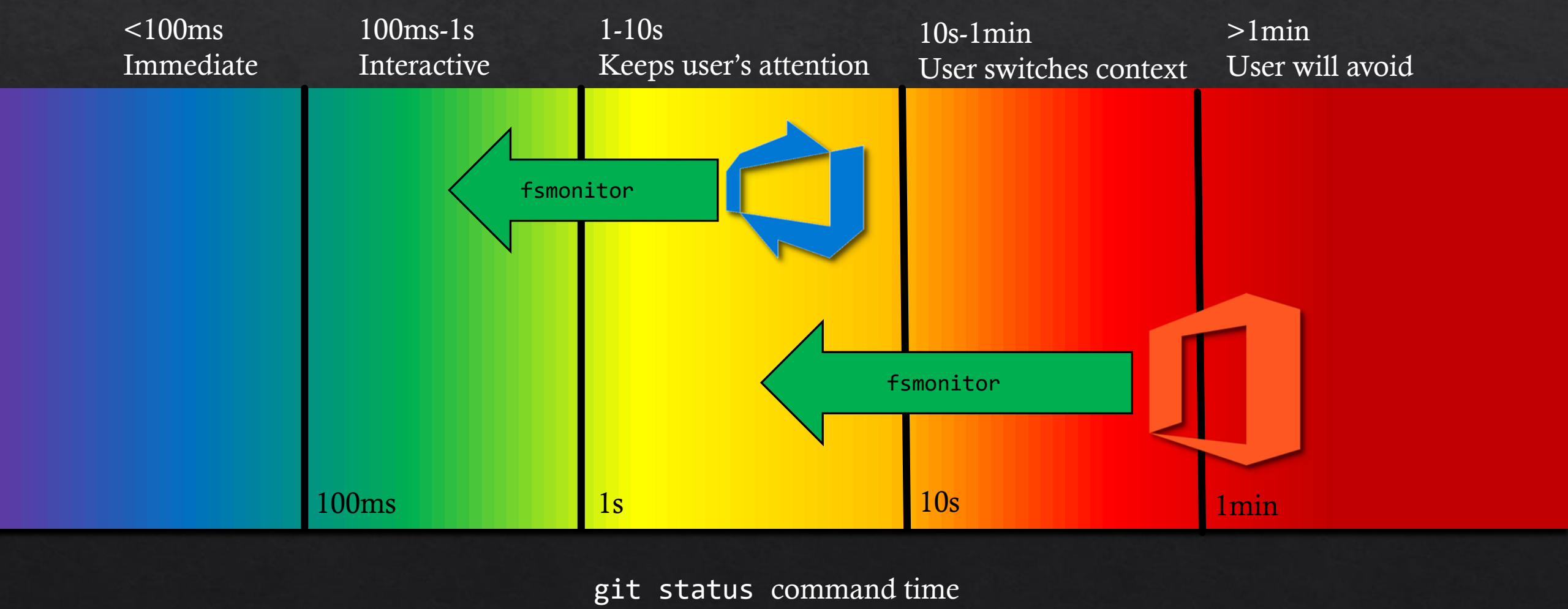
This usually results in scanning directories.

With the `fsmonitor` hook, Git can get a list from a specialize filesystem watcher, such as

https://github.com/facebook/watchman



Spectrum of Perceived Performance



`git status` command time

How can Git better focus on files that matter?

Sparse-Checkout

- Continued UX improvements
  - `git sparse-checkout add <dir>`
  - `git sparse-checkout remove <dir>`
  - `git sparse-checkout stats`
  - Update with non-empty `git status`

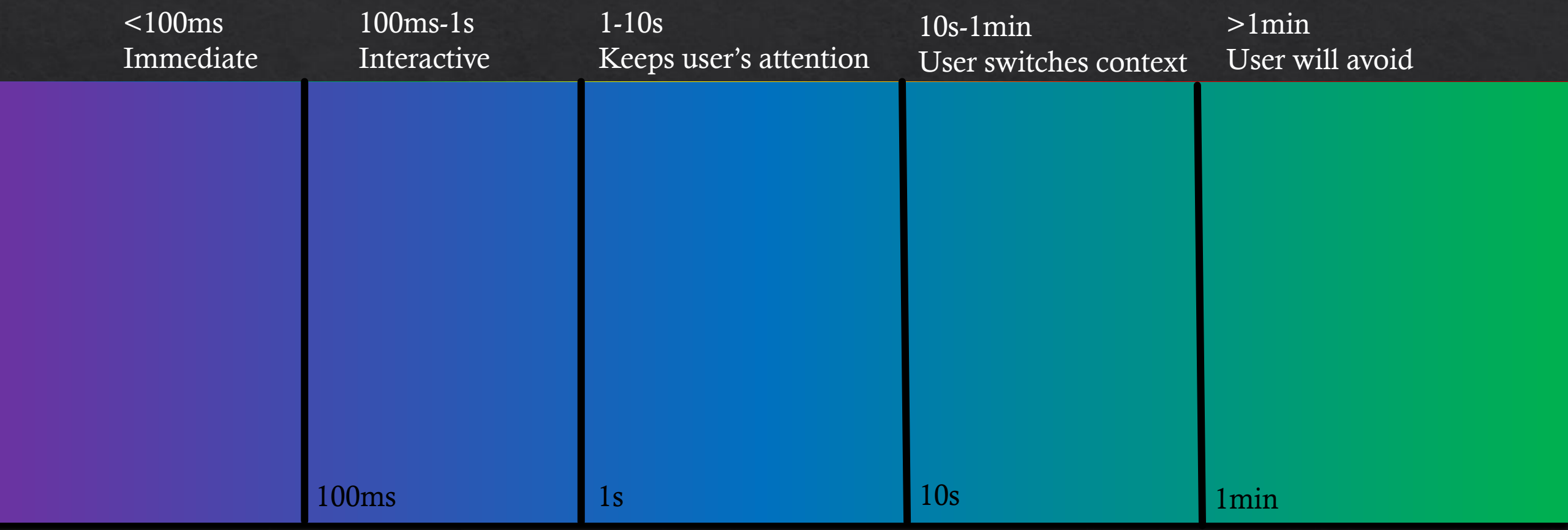
Filesystem Monitor

- Make the hook more robust, faster
- We are preparing a **Git-aware** filesystem monitor.

Lesson 2: Reduce Object Transfer

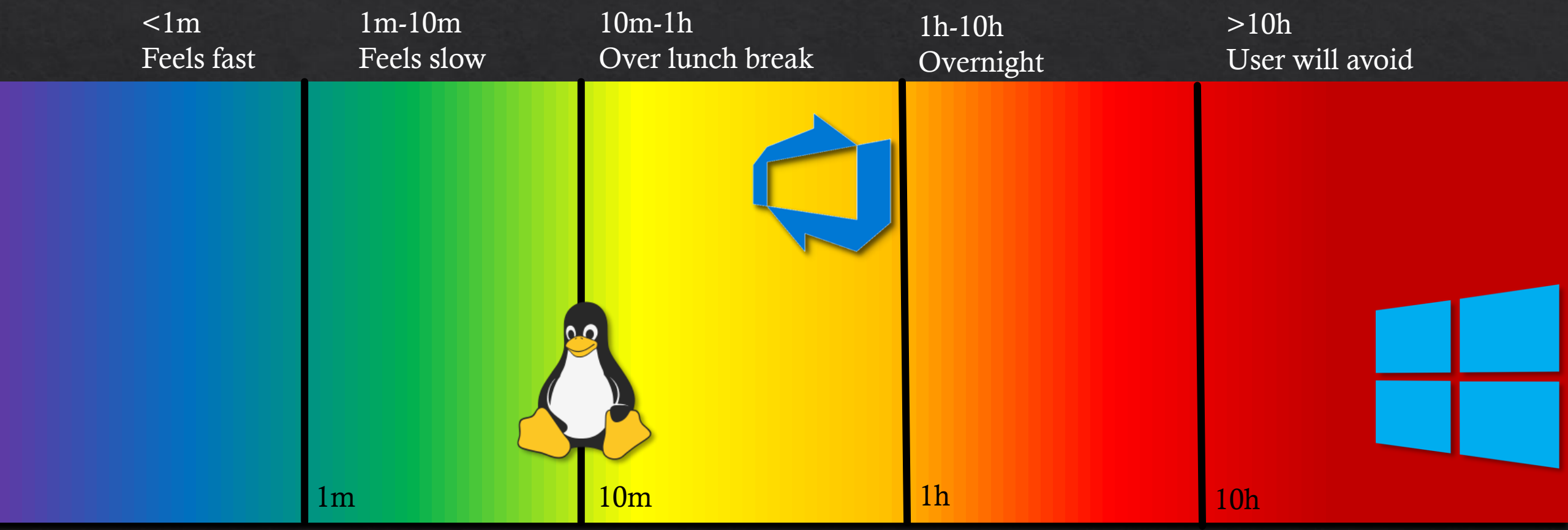
```
dstolee@stolee-book MINGW64 /c/_git/t
$ git clone --single-branch https://dev.azure.com/mseng/_git/AzureDevOps
Cloning into 'AzureDevOps'...
remote: Azure Repos
remote: Found 6938156 objects to send. (1090 ms)
Receiving objects:   0% (18433/6938156), 3.13 MiB | 1.17 MiB/s
```

Spectrum of Perceived Performance



`git clone` time

Spectrum of Perceived Performance



`git clone` time

GVFS Protocol → Partial Clone

GVFS protocol (Created 2015-16)

- Uses these REST API endpoints:
  - `GET <url>/gvfs/config`
  - `GET <url>/gvfs/objects/{objectid}`
  - `POST <url>/gvfs/objects`
  - `GET <url>/gvfs/prefetch`
  - `POST <url>/gvfs/sizes`

Git Partial Clone (Created 2018)

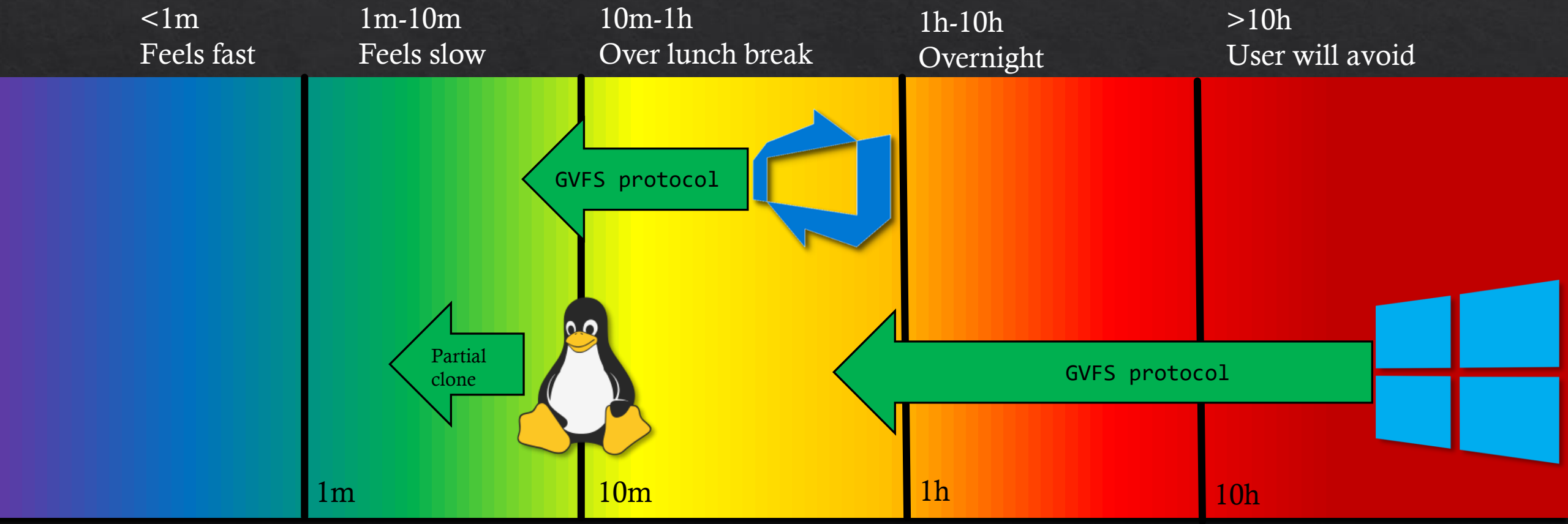
- `git clone --filter=blob:none <url>`
- Fetches only commits and trees
- Blobs are fetched in a batch request during `git checkout` and similar requests

Now available on all GitHub.com repositories!

Reduced Object Transfer + Sparse-Checkout = Success!

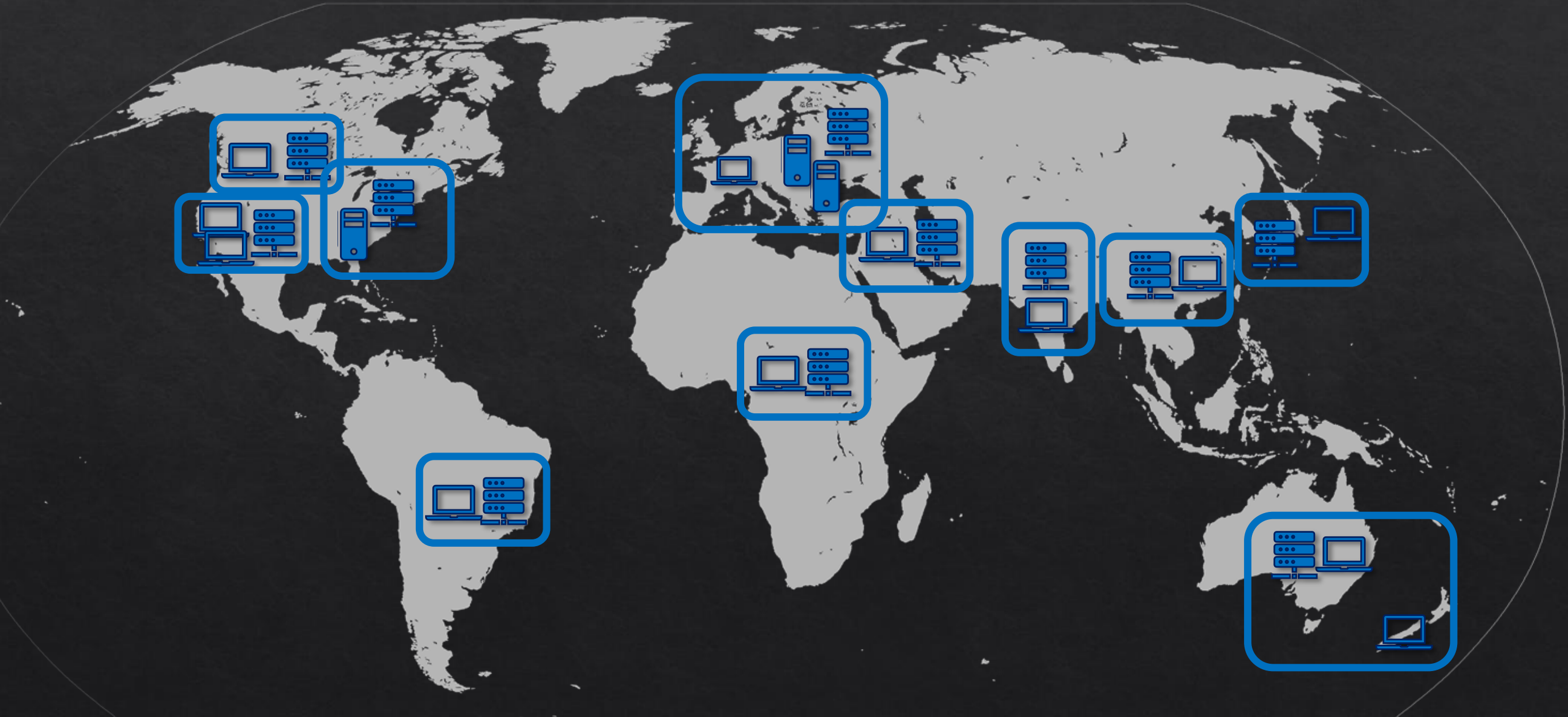
`https://git-scm.com/docs/partial-clone`

Spectrum of Perceived Performance



Time for git clone vs partial clone or GVFS protocol

GVFS Cache Servers and Git Promisor Remotes

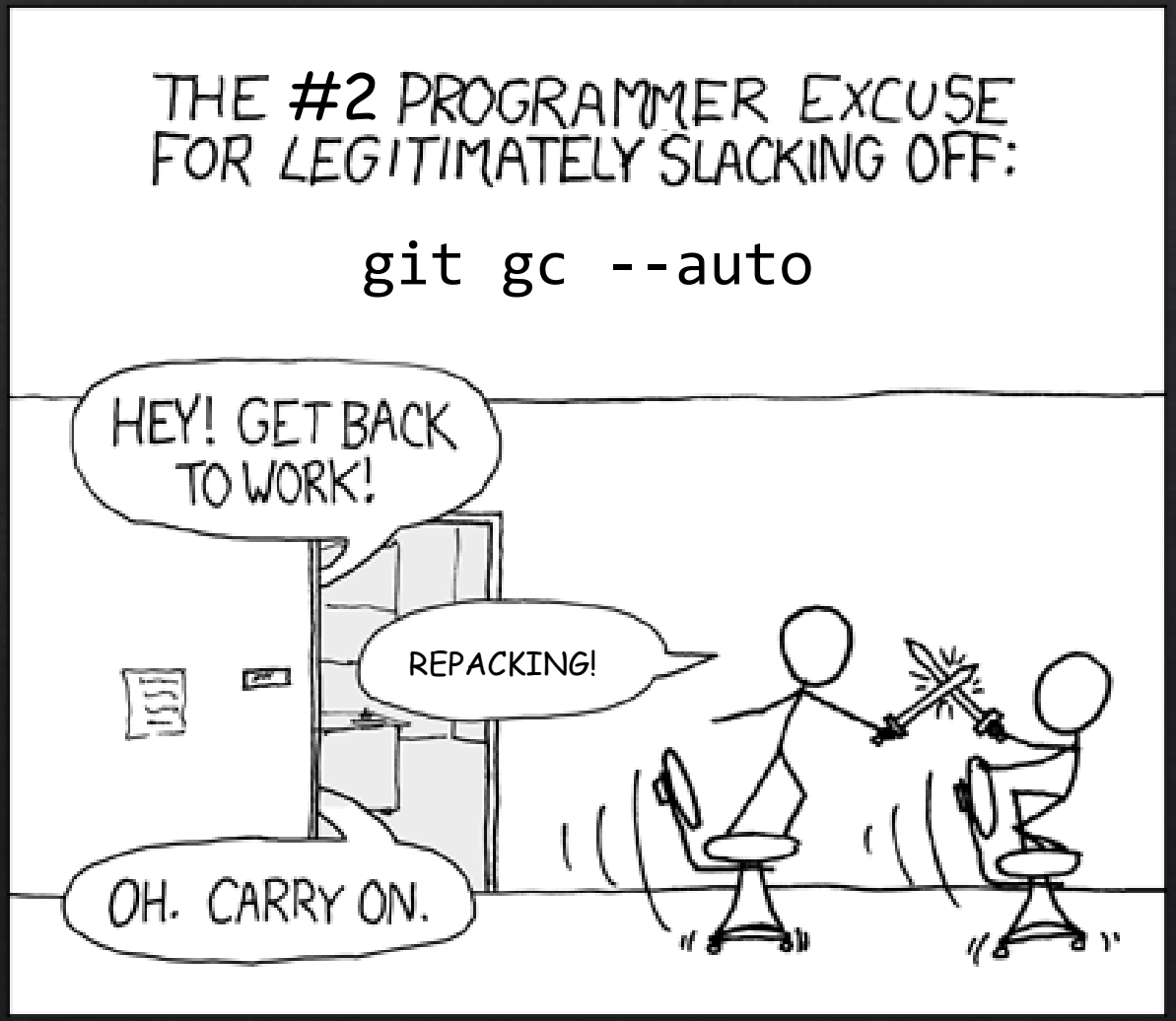


Recommended Updates to Partial Clone

1. Extend multiple promisor remotes to do commit and tree fetches.

2. Extend Git protocol to assist auto-discovery of nearby promisor remotes

Lesson 3: Don’t wait for expensive operations



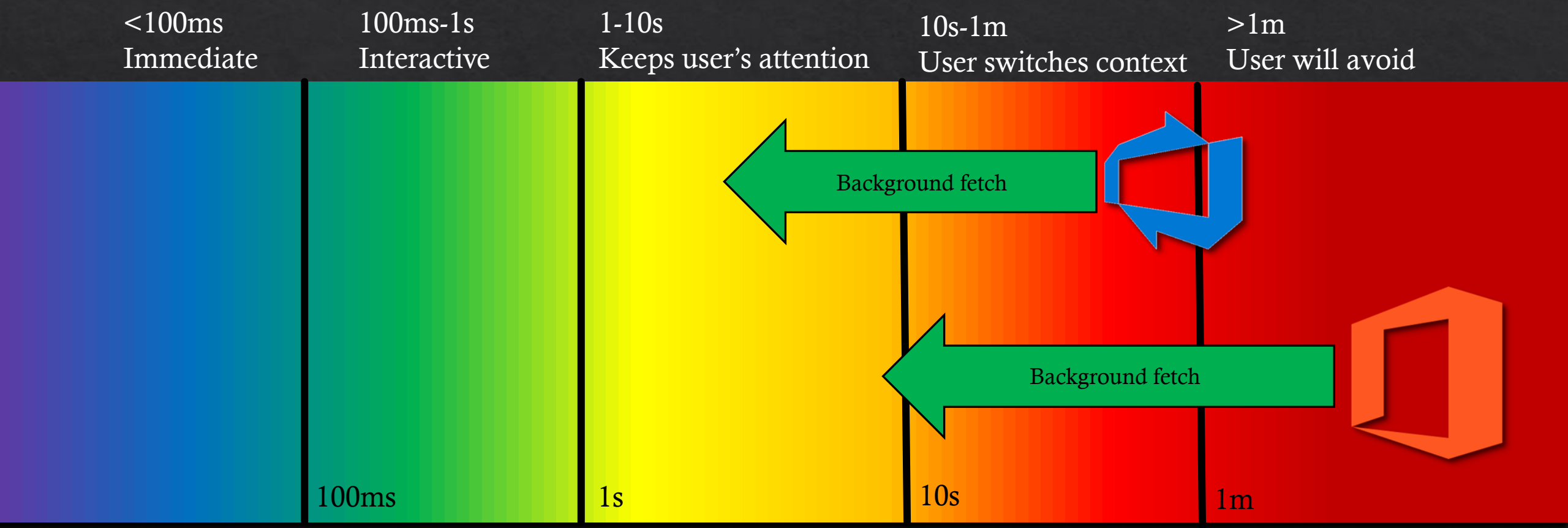
https://xkcd.com/303/

Background Maintenance

The following can be done in the background, reducing user-blocking time:

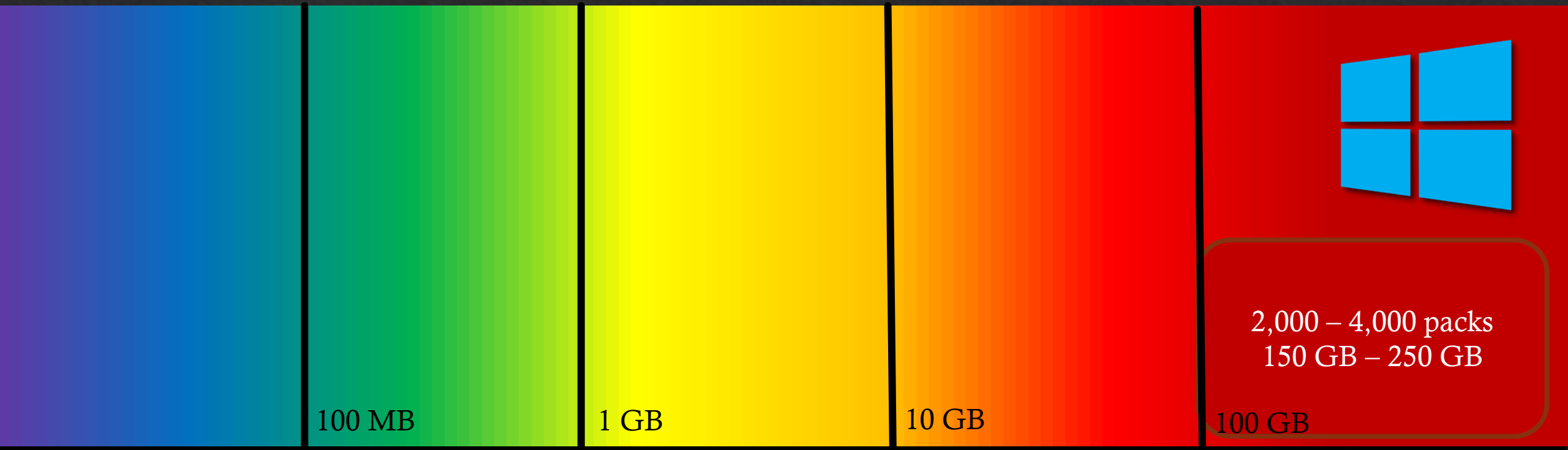
- **Background fetch:** get latest objects from remotes
- **Loose Objects:** Clean up loose objects safely
- **Pack-files:** Index and repack pack-files incrementally

Spectrum of Perceived Performance

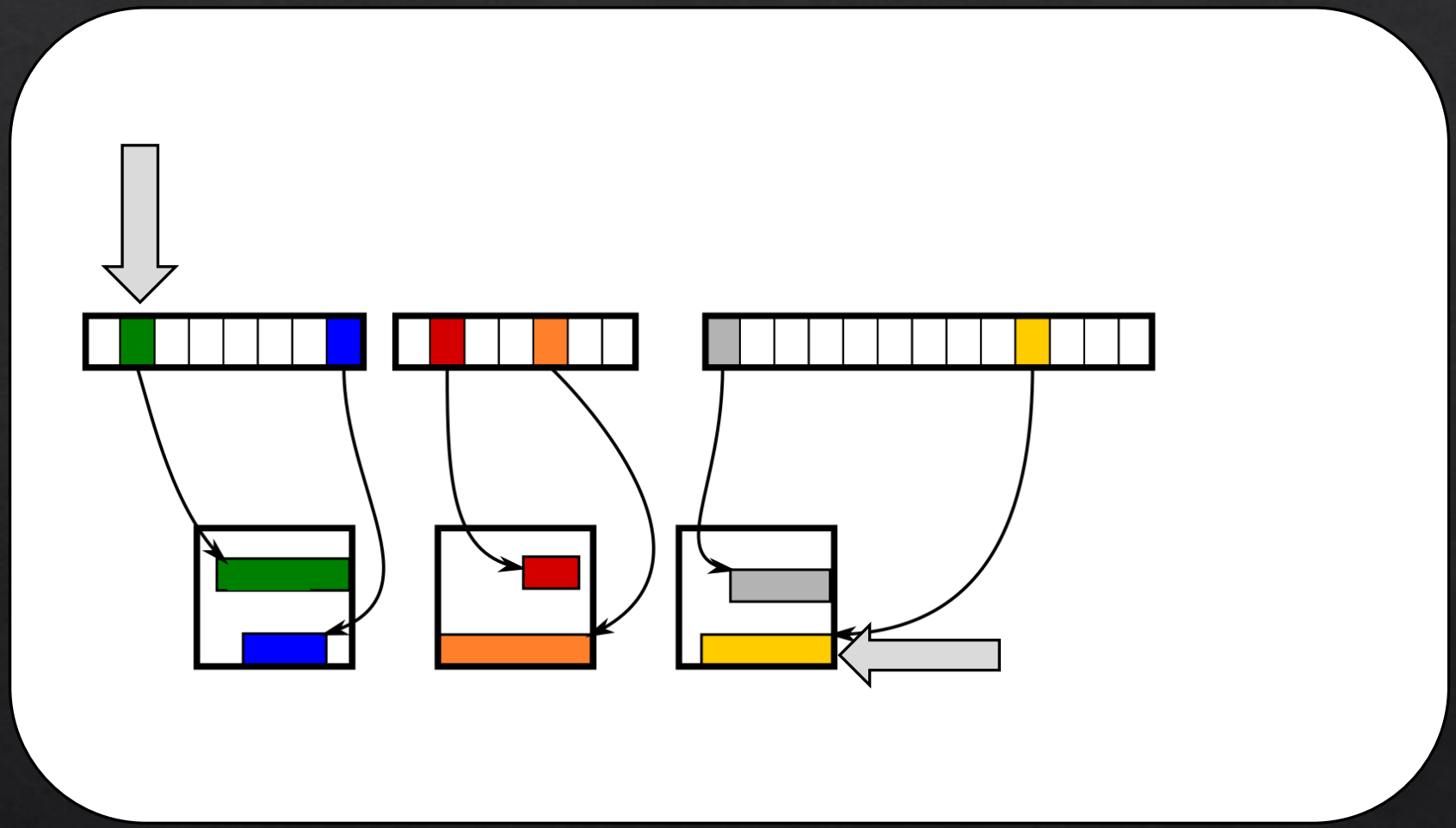


Foreground `git fetch` time

Too Many Packs?



Too Many Packs?

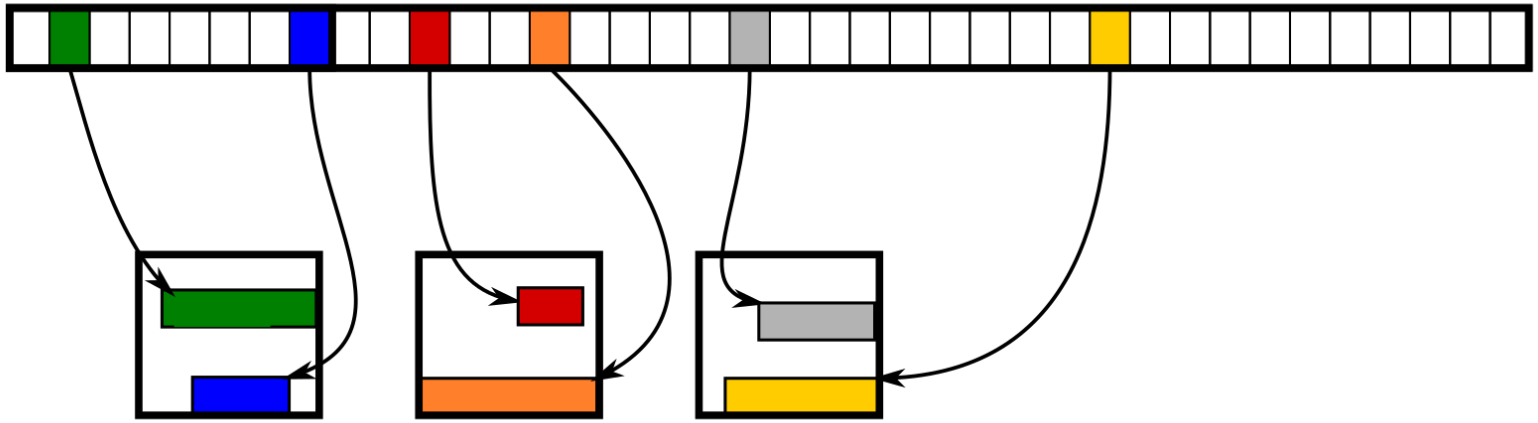


Too Many Packs?

```
git multi-pack-index write
```

multi-pack-index:

.pack files:

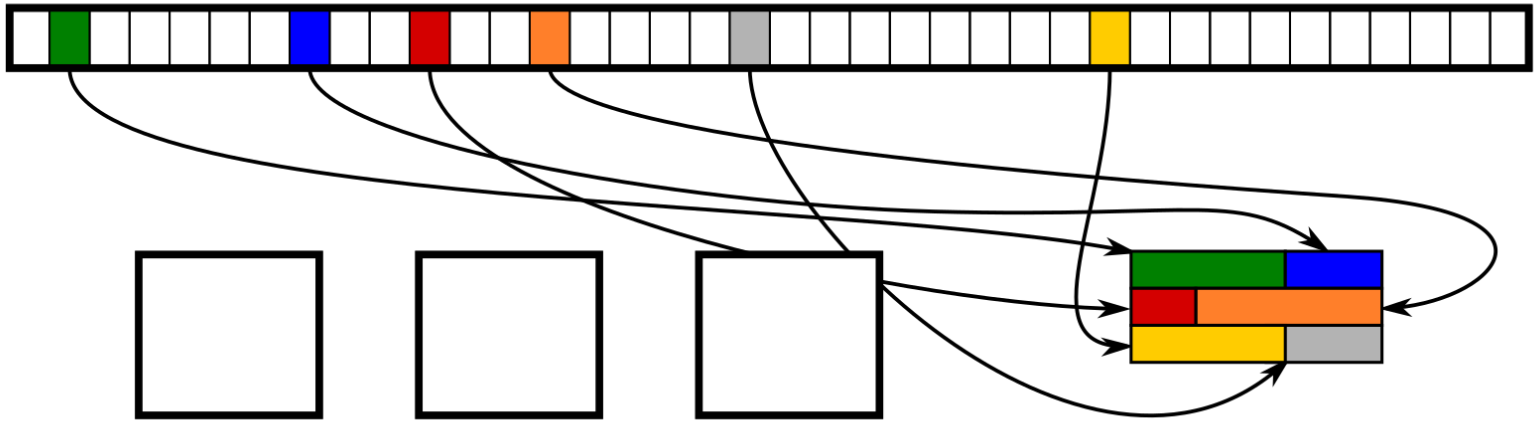


Incremental Repack

```
git multi-pack-index repack
```

multi-pack-index:

.pack files:

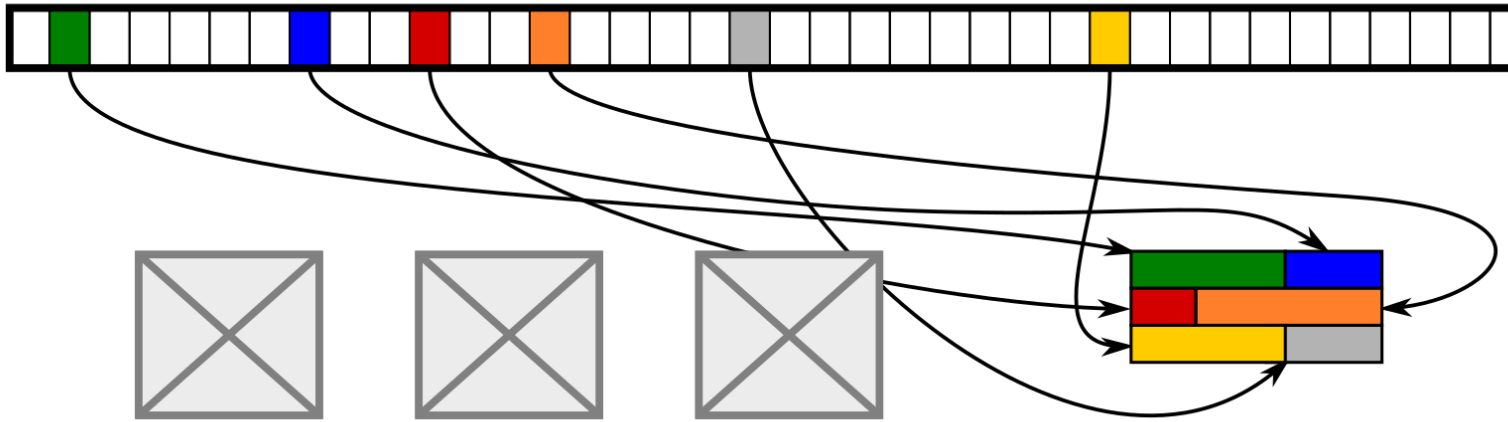


Incremental Repack

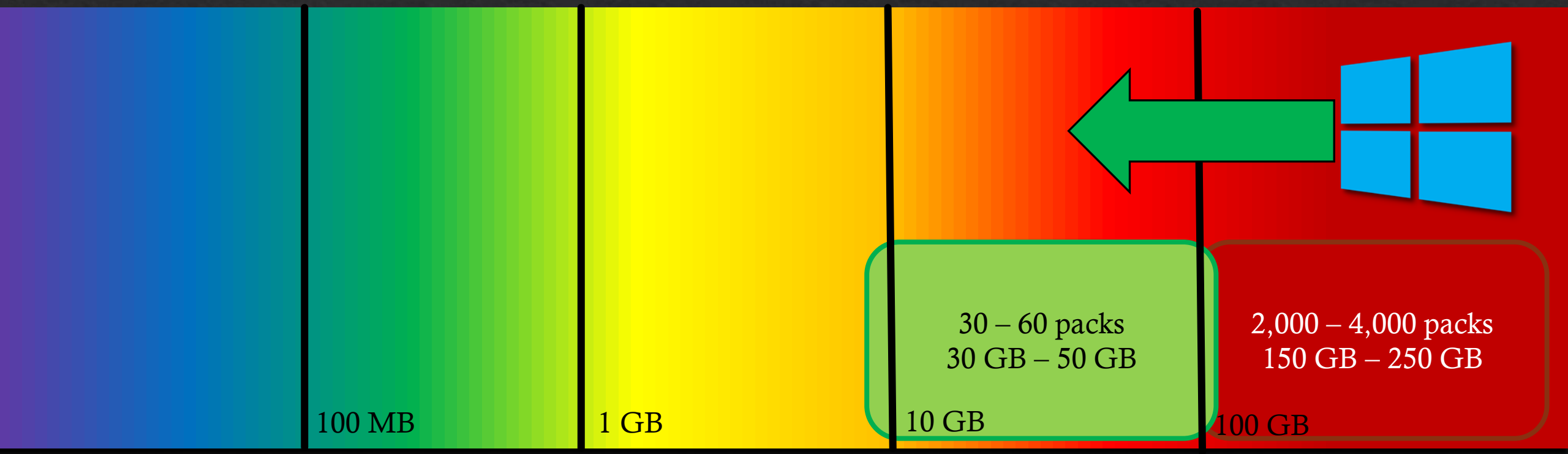
```
git multi-pack-index expire
```

multi-pack-index:

.pack files:



Spectrum of Scale



Background Maintenance in Git?

- *Should* Git do background maintenance?
- What of these background jobs make sense for most users?
- How might expert users want to customize these jobs? (Frequency, batch sizes, etc.)

Scalar

https://github.com/microsoft/scalar

Installers available for Windows and macOS

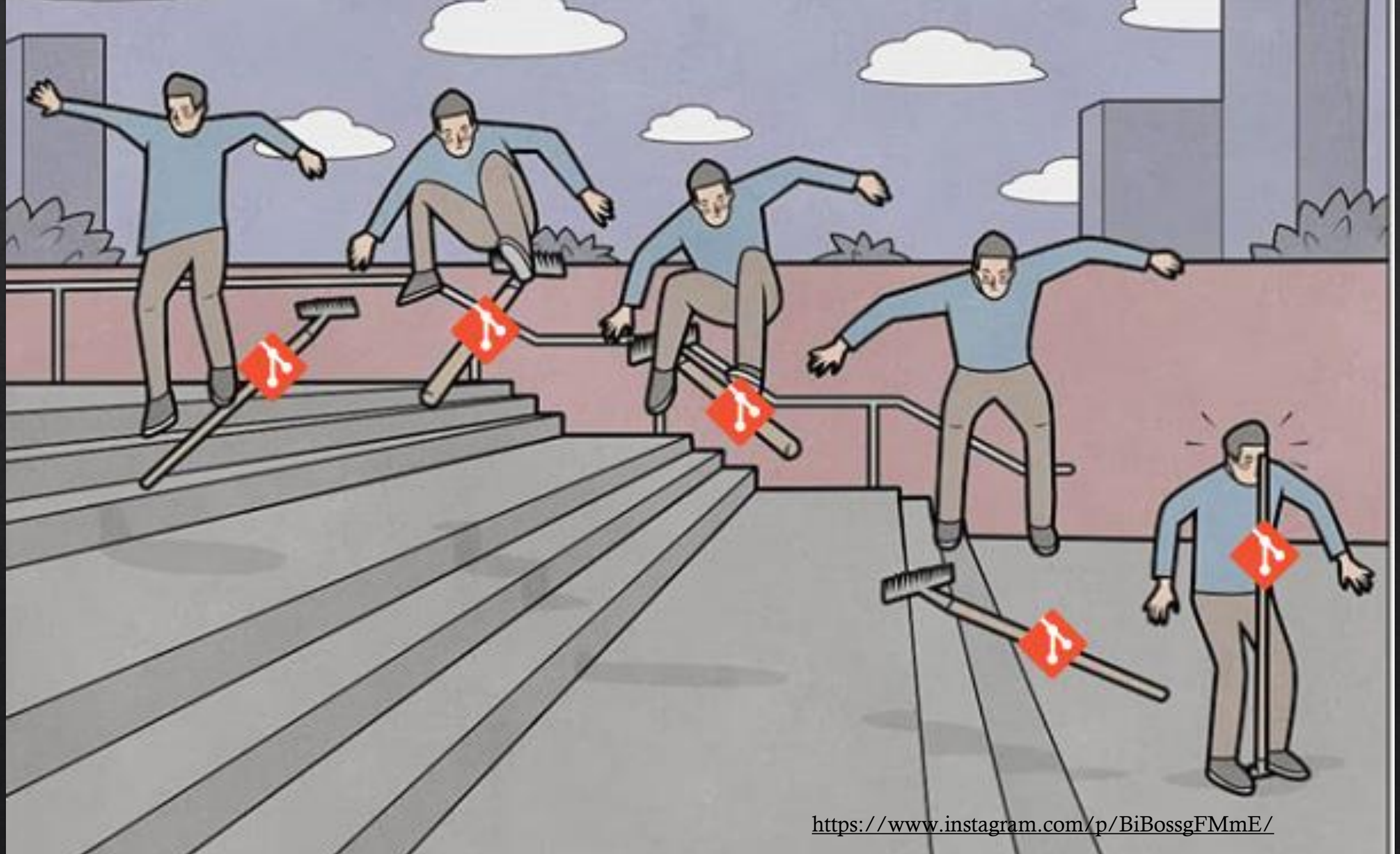
Scalar Quick Start

```
$ git version
git version 2.25.1.vfs.1.2

$ scalar version
scalar 20.03.167.1

$ scalar register
Successfully registered repo at ‘/Users/stolee/_git/vscode’
```

What does `scalar register` do?

1. Sets advanced Git config settings for optimal performance
2. Initializes filesystem monitor hook, if Watchman is installed

   https://github.com/facebook/watchman
3. Starts background maintenance
   1. Background fetch
   2. Write commit-graph
   3. Clean up loose objects
   4. Clean up pack-files



https://www.instagram.com/p/BiBossgFMmE/

What does `scalar clone` do?

1. Creates new repository with working directory `<name>/src`
2. If remote supports **GVFS protocol**, then configure to use it.
3. Otherwise, configures Git to use **partial clone**.
4. Downloads all commits and trees.
5. `git sparse-checkout init --cone`
6. Everything from `scalar register`

Scalar bridges the gap *for now*

Hopefully, one day Scalar will set recommended config *and that's it*.

Scalar

https://github.com/microsoft/scalar

Installers available for Windows and macOS