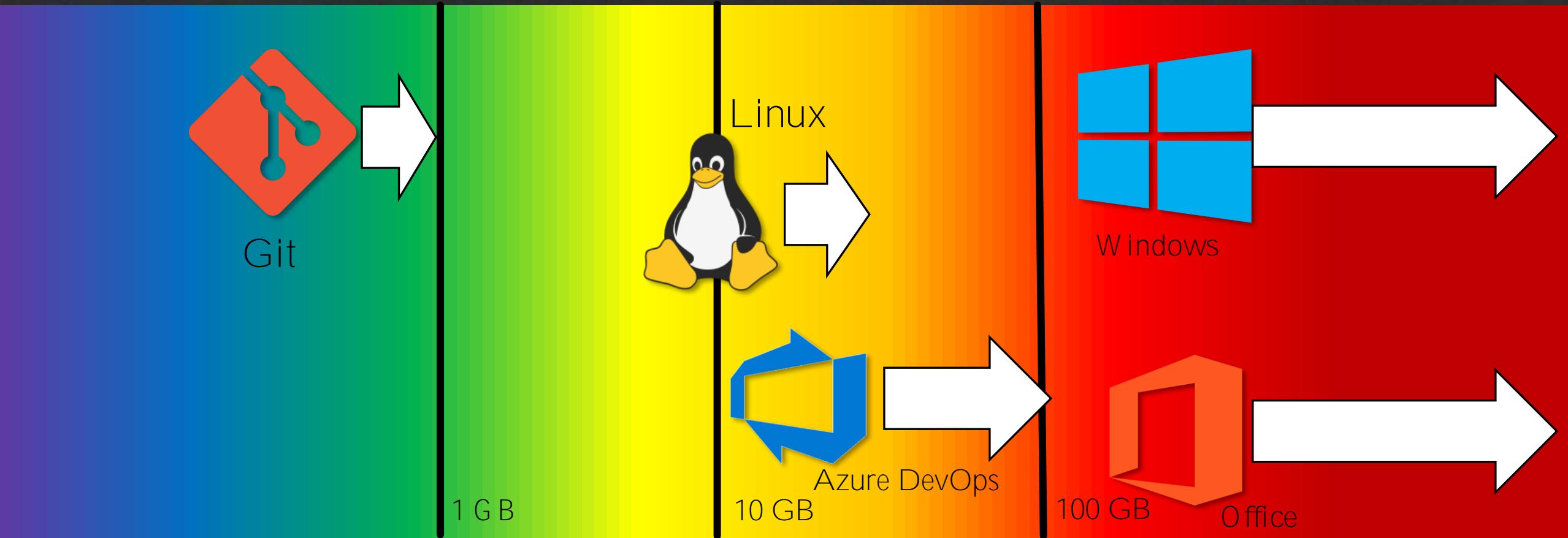
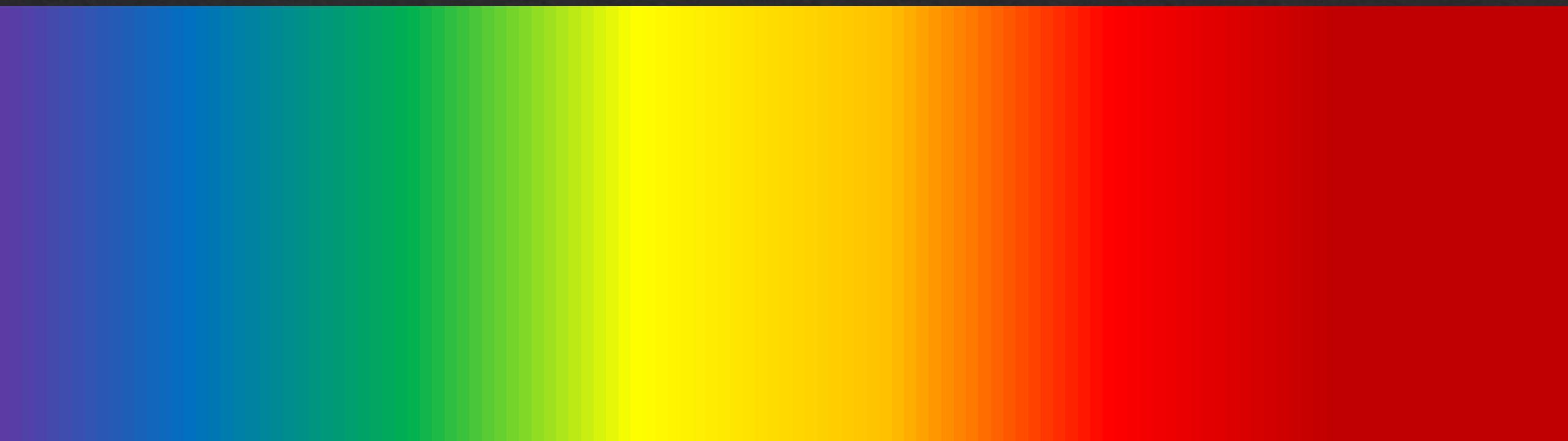


Spectrum of Scale



Pack-file size for initial clone

Spectrum of Perceived Performance





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



Scalar

<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

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.

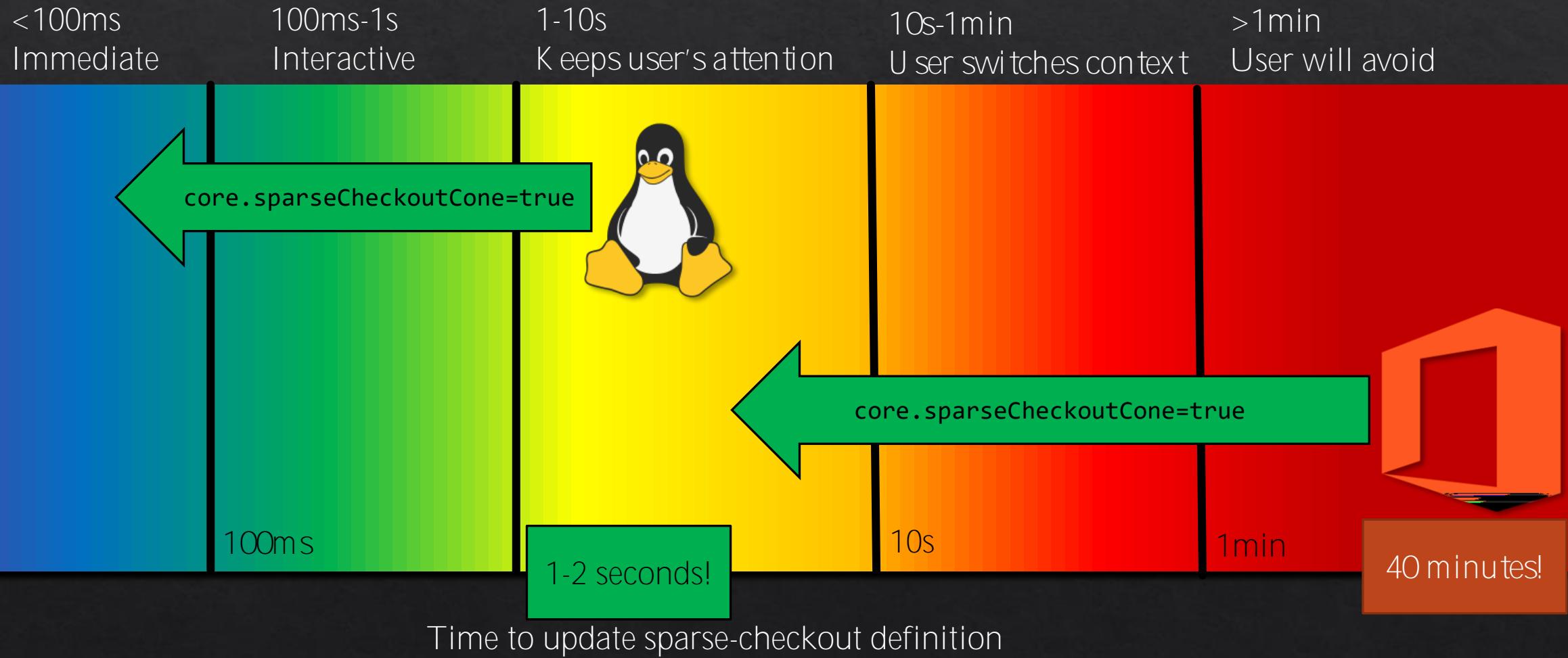
Included paths can be expanded using

```
git sparse-checkout set <dir> <dir> ...
```

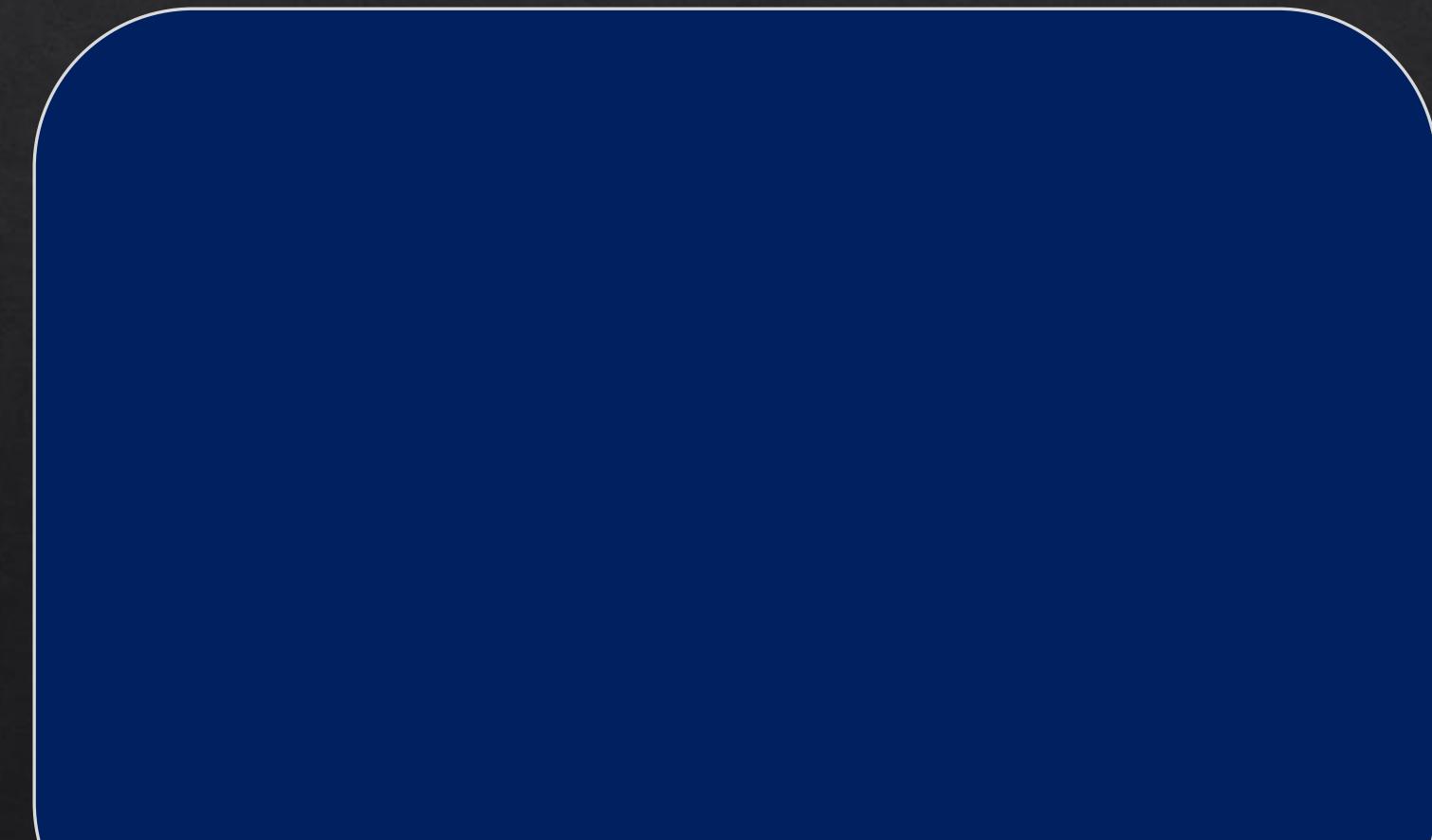
In this example, we use:

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

Spectrum of Perceived Performance



Finding Modified Files with Filesystem Monitor



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 command execution

in the following sequence:

1. Get file list

2. Check modification

3. Add modified files

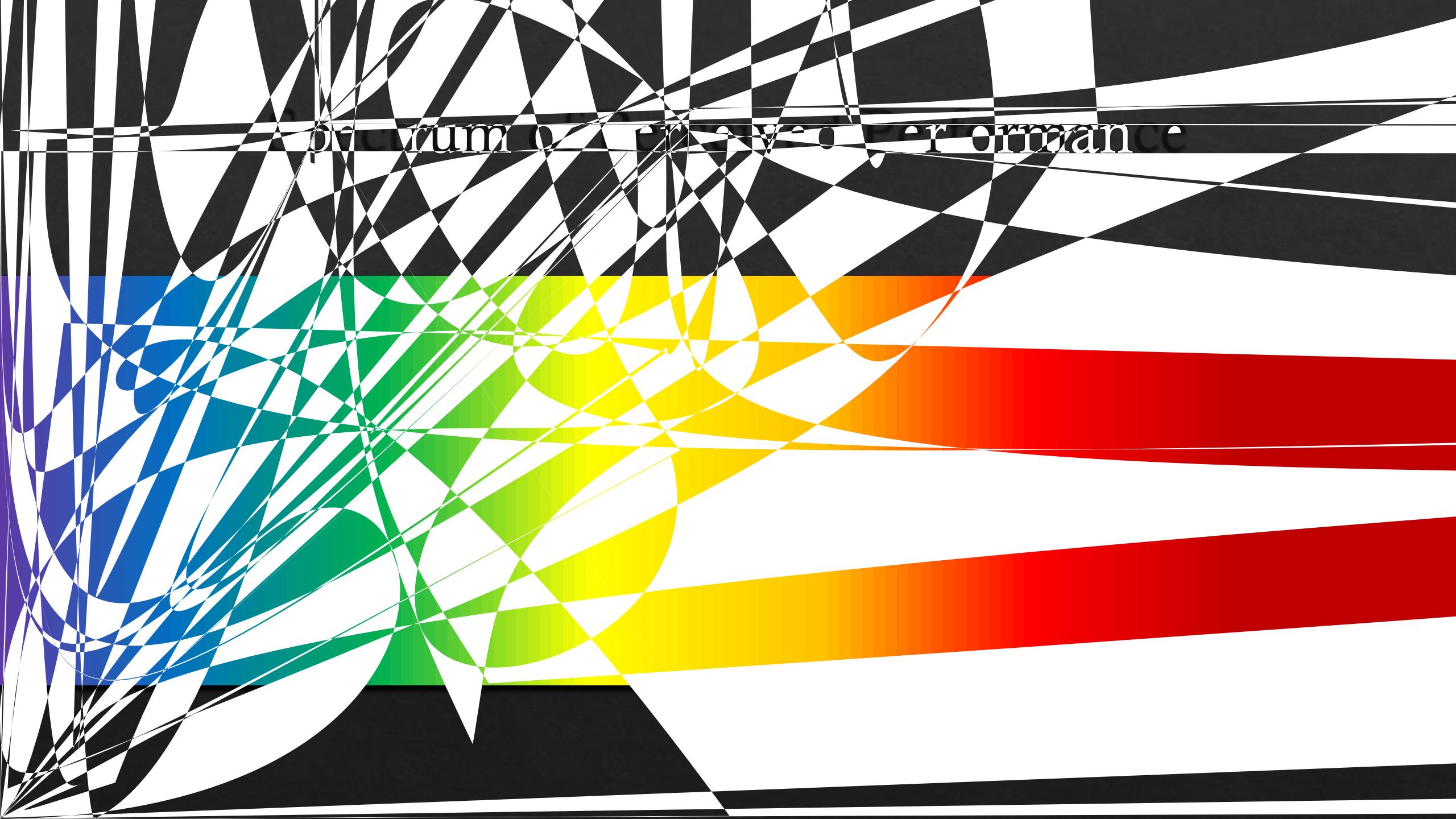
4. Commit changes

5. Push changes

6. Pull changes

7. Merge changes

8. Repeat

The background features a complex pattern of black and white triangles that overlap and intersect, creating a sense of depth and motion. A horizontal band of color runs across the middle of the image, transitioning from purple on the left to red on the right. This color band is composed of many small, semi-transparent squares.

Optimum video performance

How can Git better focus on files that matter?

Sparse-Checkout

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

Lesson 2: Reduce Object Transfer

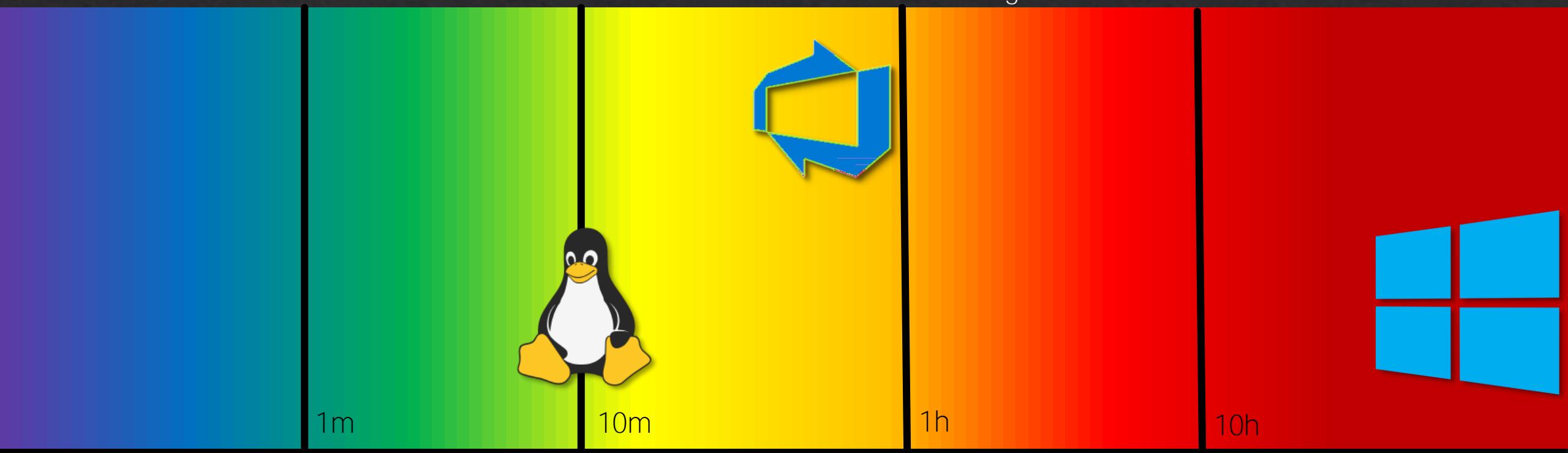
```
dstolee@dstolee-book MINGW64 /c/_git/t
$ git clone --single-branch https://dev.azure.com/mseng/_git/AzureDevOps
Cloning into 'AzureDevOps'...
remote: Azure Repos
remote: Found 6938156 objects to send. (1090 ms)
Receiving objects:  0% (18433/6938156), 3.13 MiB | 1.17 MiB/s
```

Spectrum of Perceived Performance



Spectrum of Perceived Performance

<1m Feels fast 1m-10m Feels slow 10m-1h Over lunch break 1h-10h Overnight >10h User will avoid



`git clone` time

GVFS Protocol → Partial Clone

GVFS protocol (Created 2015-16)

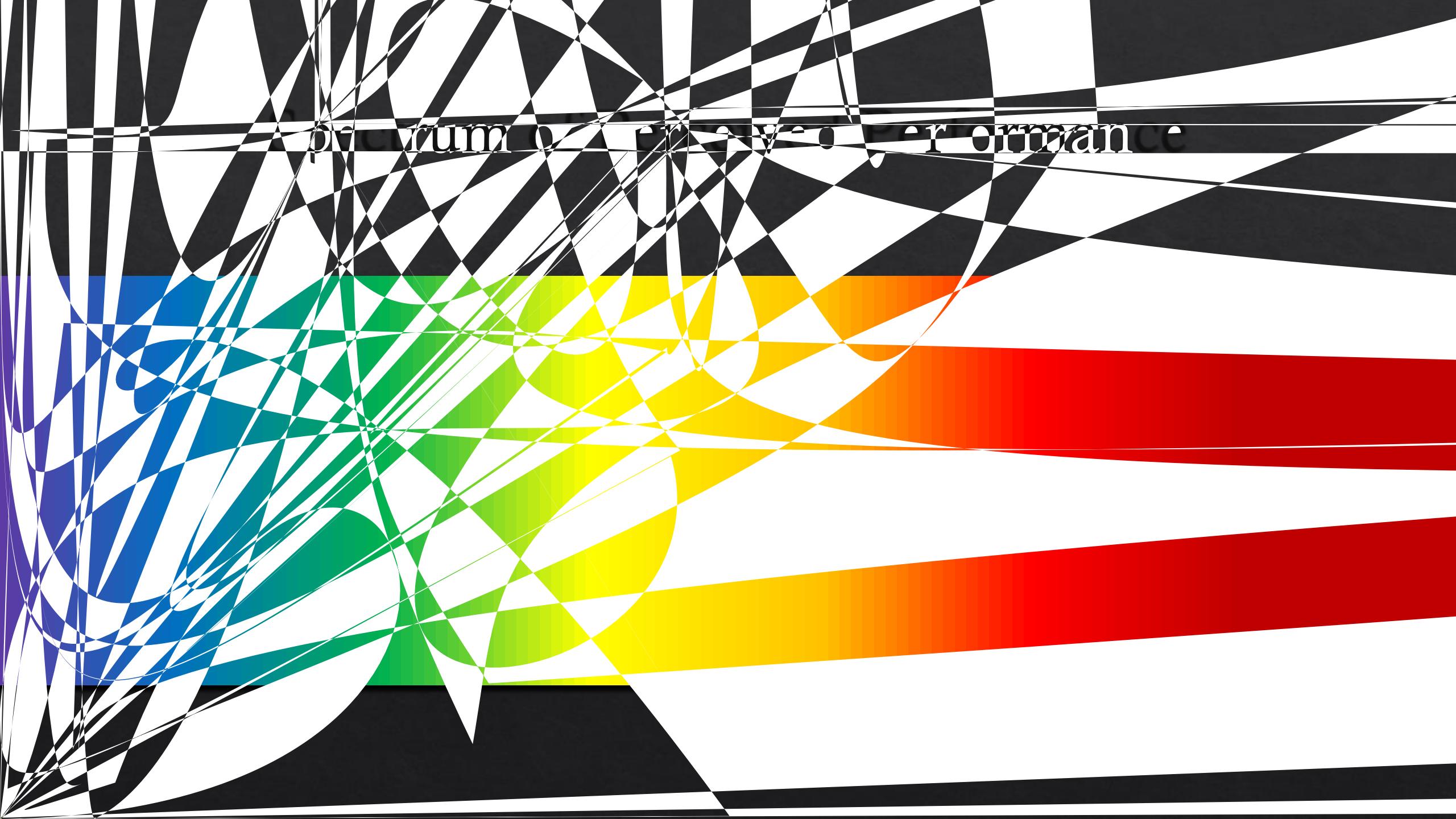
- ❖ Uses these REST API endpoints:
 - ❖ GET <url>/gvfs/config
 - ❖ GET <url>/gvfs/objects/{objectid}
 - ❖ POST <url>/gvfs/objects
 - ❖ GET <url>/gvfs/prefetch
 - ❖ POST <url>/gvfs/sizes

Git Partial Clone (Created 2018)

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

<https://git->

Optimum video performance



GVFS Cache Servers and Git Promisor Remotes



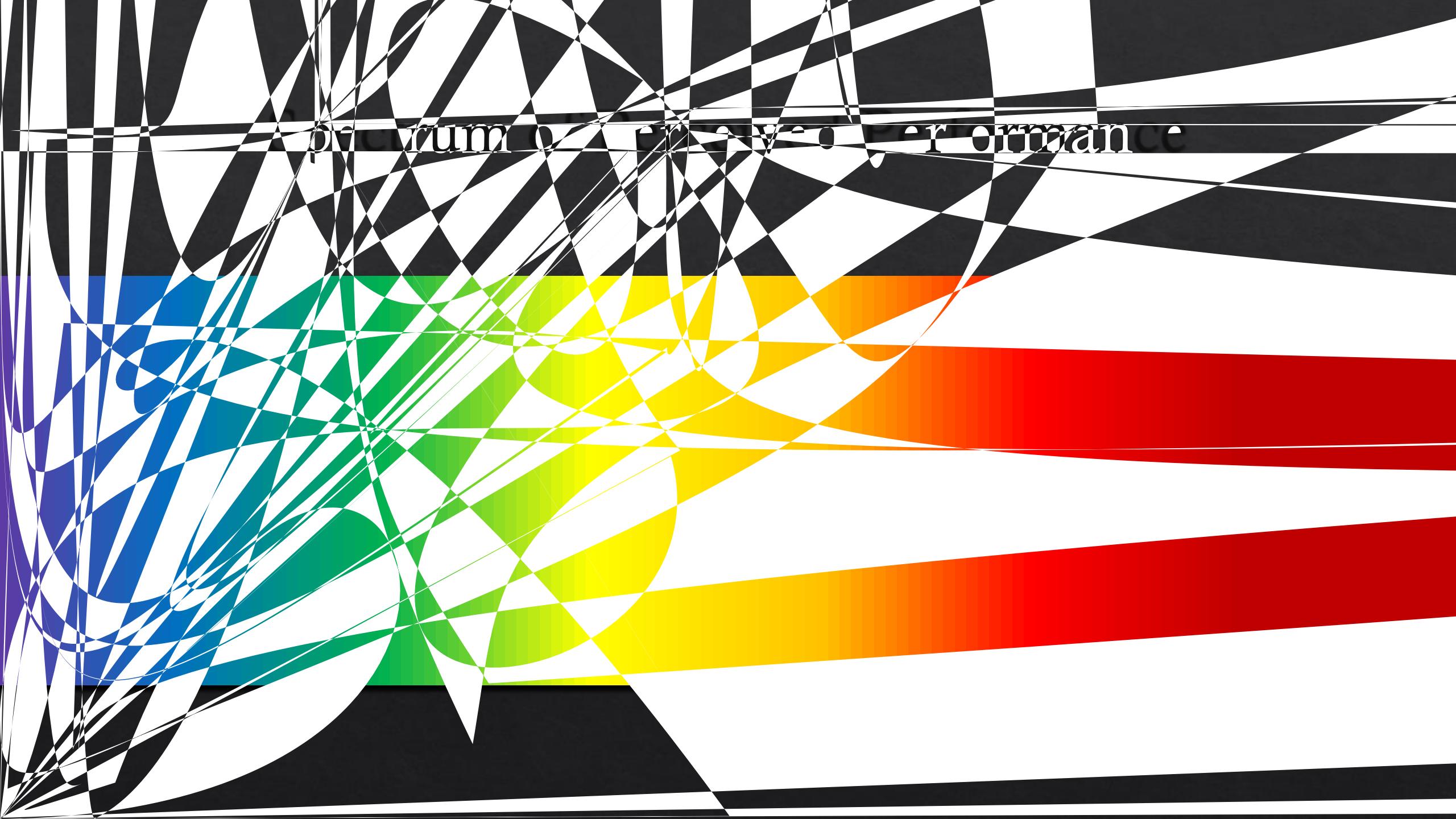
Recommended Updates to Partial Clone

1. Extend multiple promisor remotes to do commit and tree fetches.
2. Extend Git protocol to assist auto-discovery of nearby promisor remotes

Background

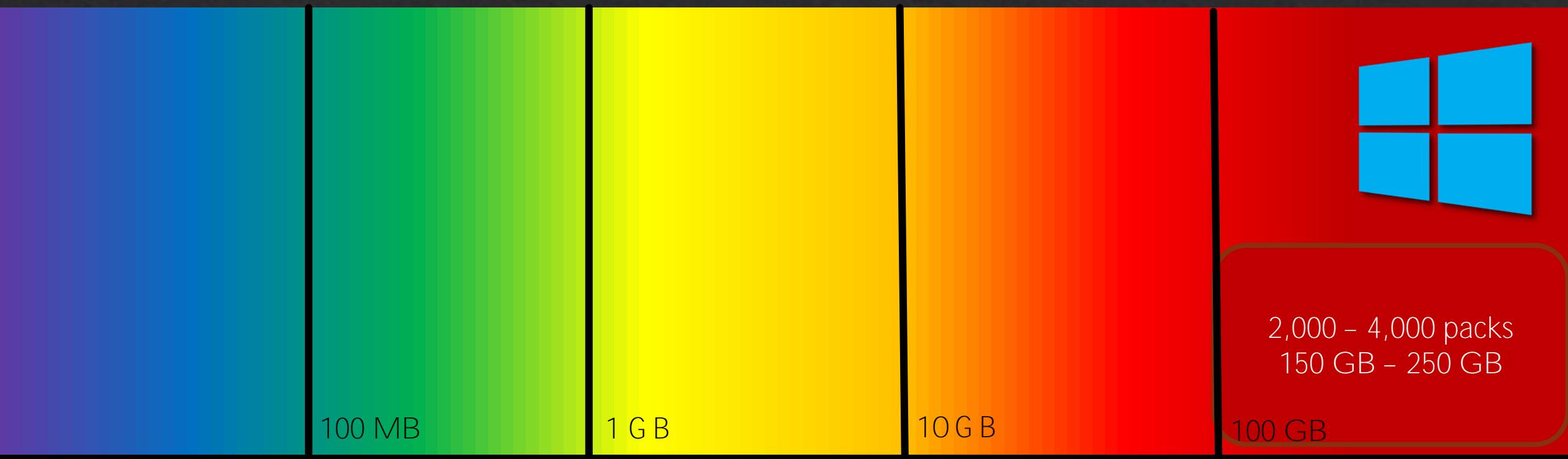
allow what can be done in the b



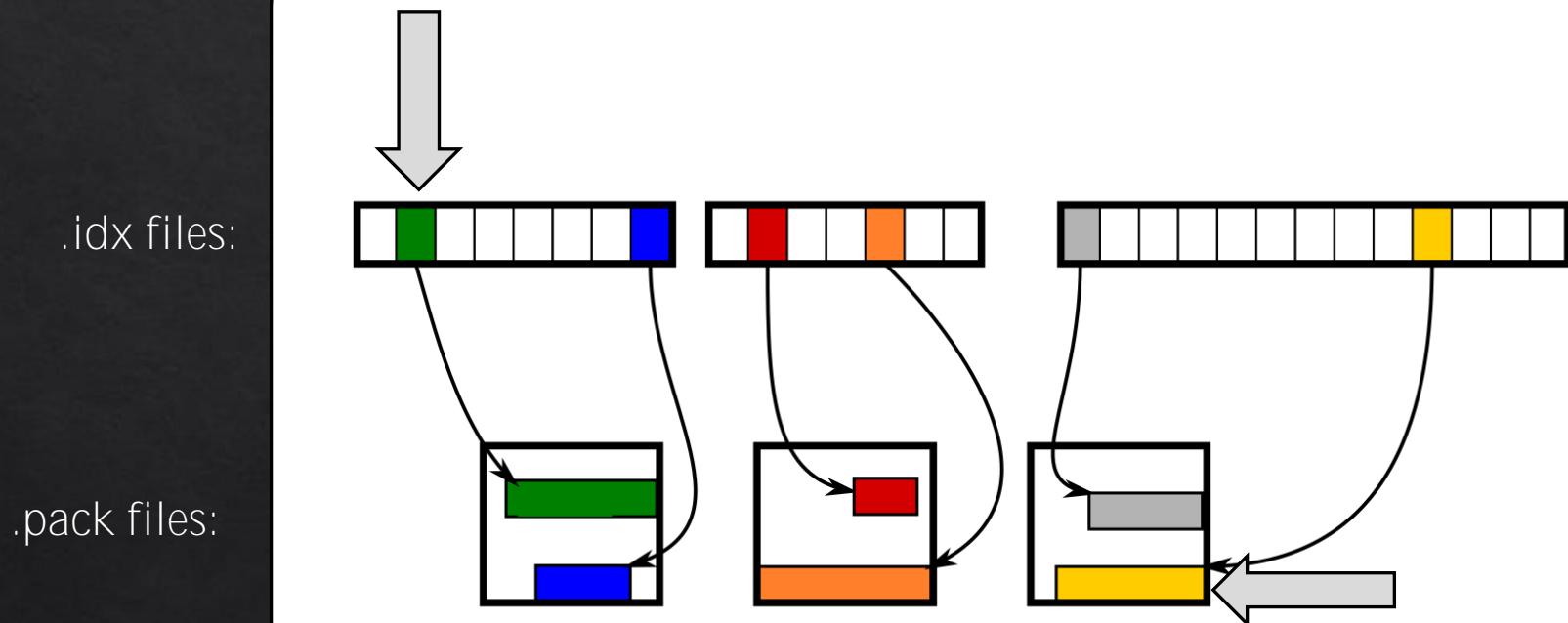
The background features a complex pattern of black and white triangles that overlap and intersect, creating a sense of depth and motion. A horizontal band of color runs across the middle of the image, transitioning from purple on the left to red on the right. This color band is composed of many small, semi-transparent squares.

Optimum video performance

Too Many Packs?



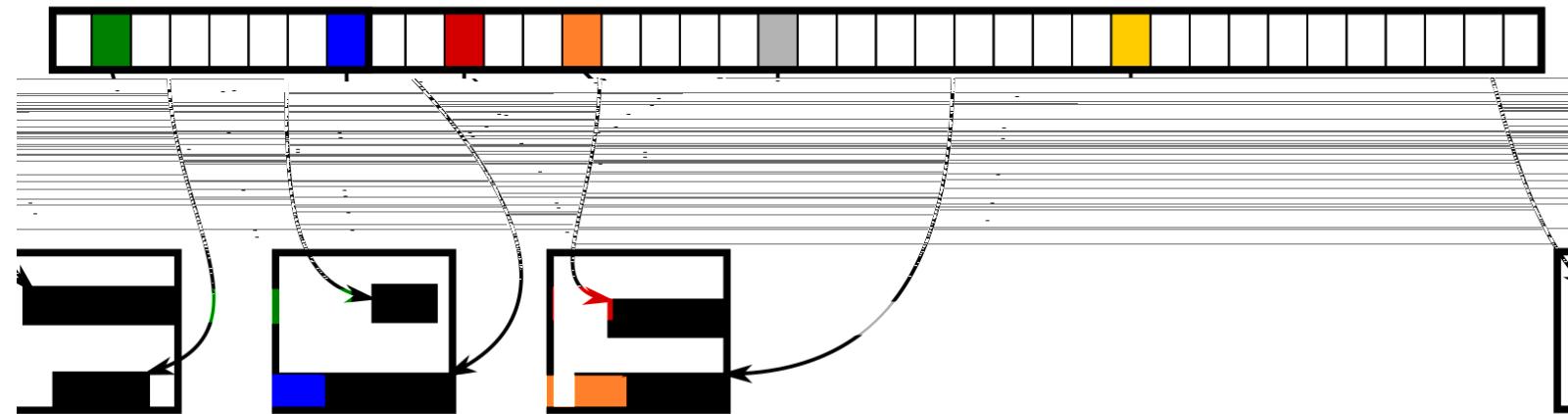
Too Many Packs?



Too Many Packs?

`git multi-pack-index write`

multi-pack-index:

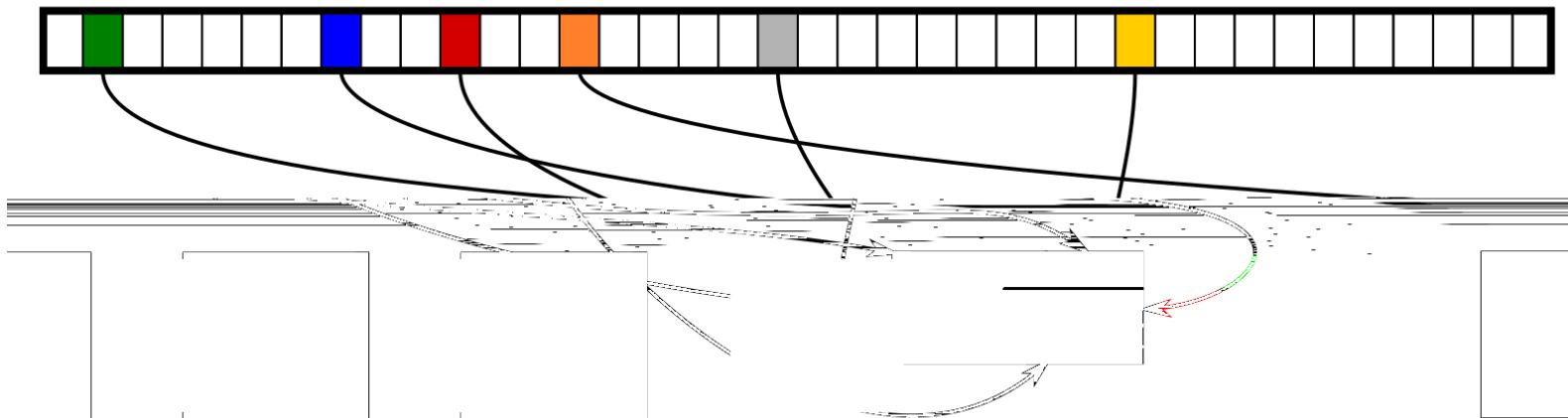


.pack files:

Incremental Repack

`git multi-pack-index repack`

multi-pack-index:



.pack files:

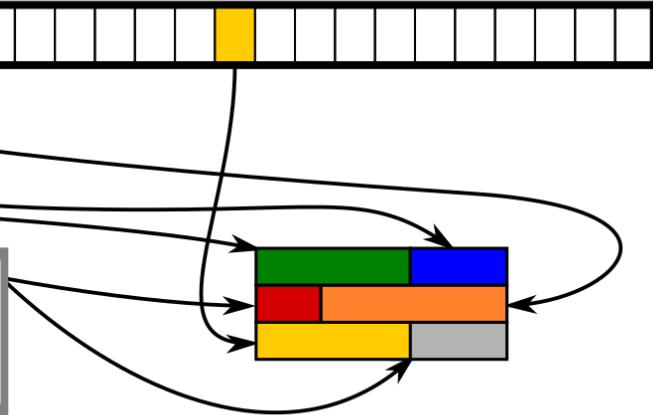
Incremental Repack

`git multi-pack-index expire`

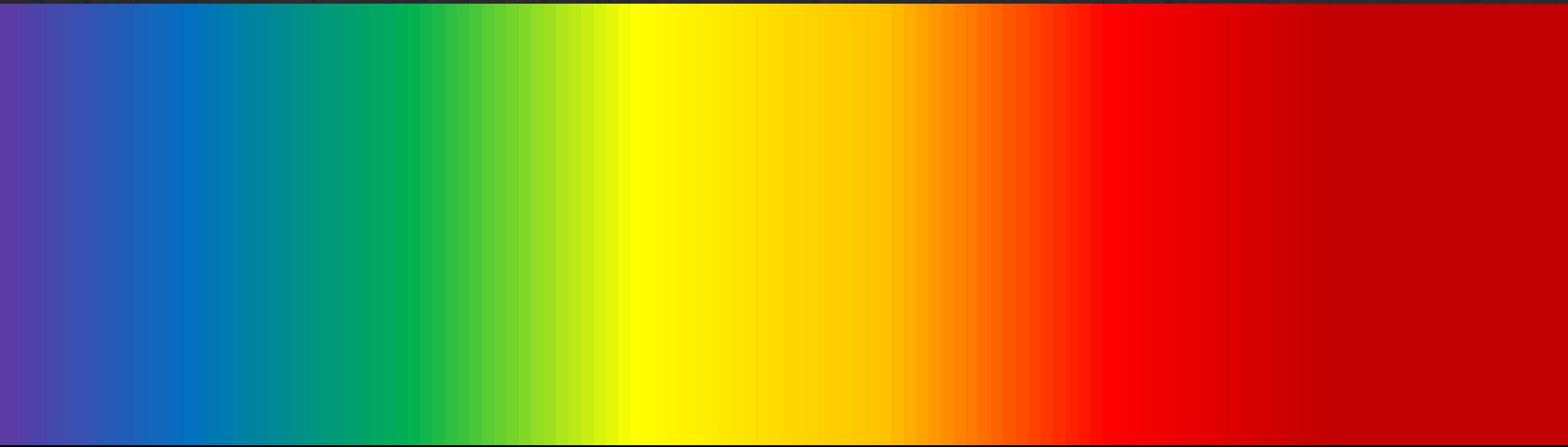
multi-pack-index:



.pack files:



Spectrum of Scale



Background Maintenance

Si no es así, ¿de qué se trata el tema que maneja?



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
```

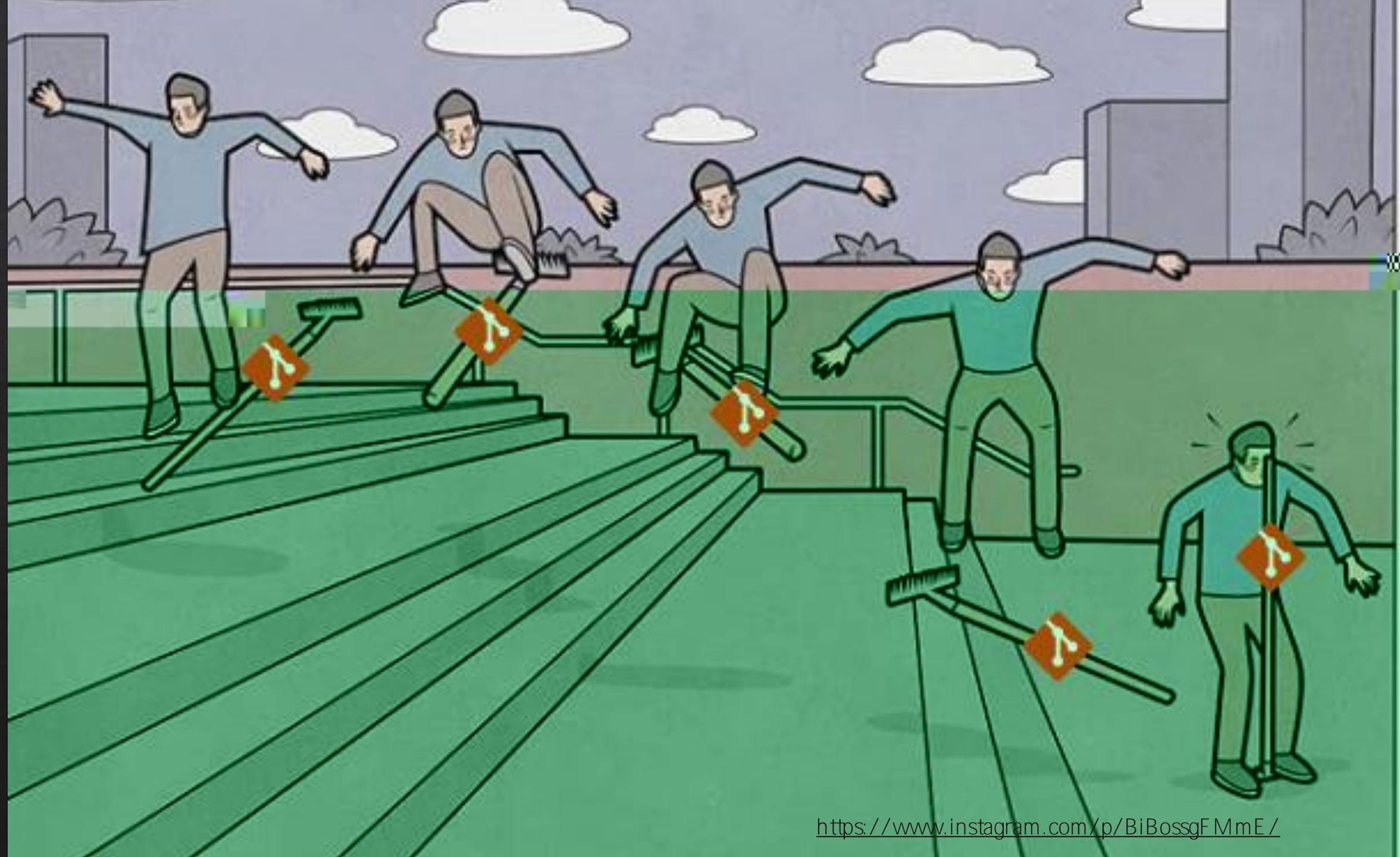
```
scalar 20.03.167.1
```

```
$ 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**.

out init --cone

lar register

5. git sparse-checkout

6. Everything from scalar

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