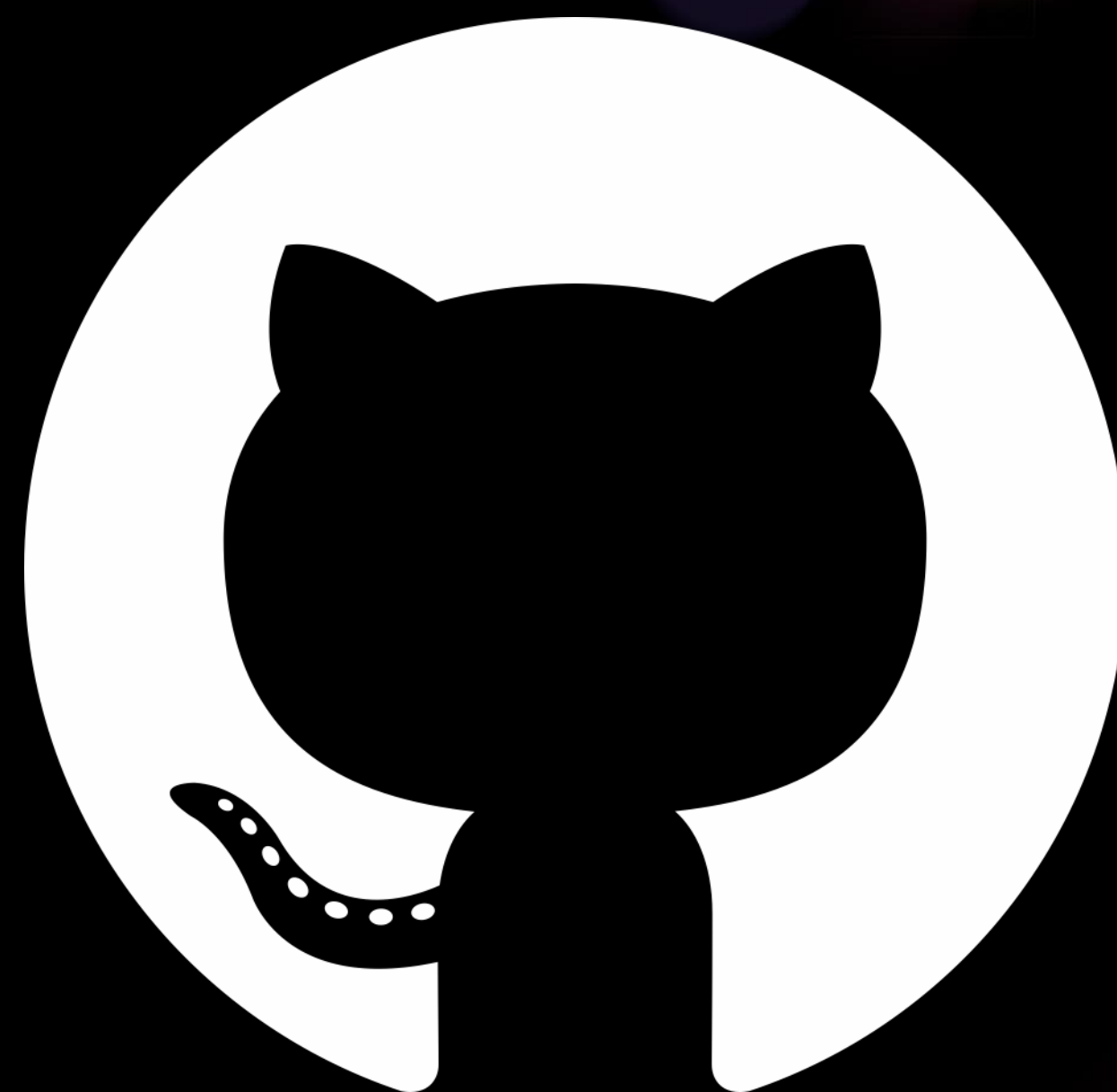




 Universe2020

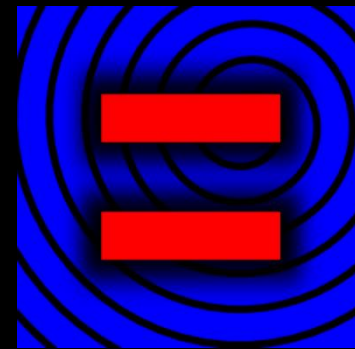


**Optimize your
monorepo experience**

These people make the magic happen

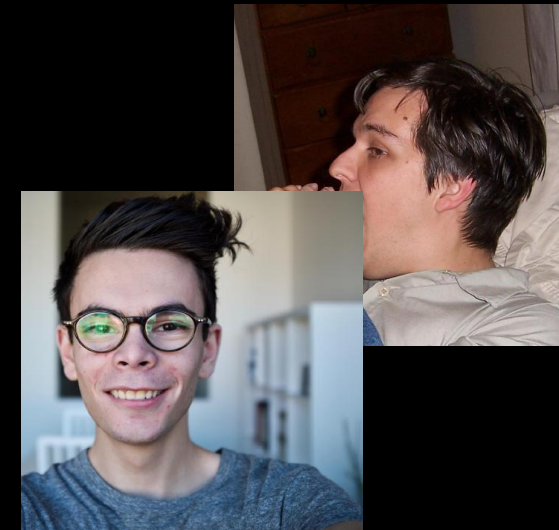


@derrickstolee



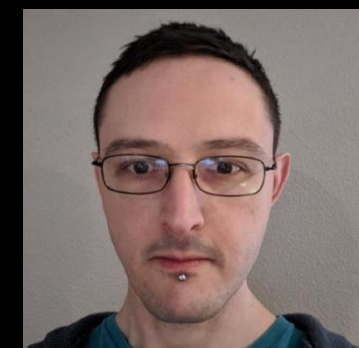
Git Client

@dscho
@jeffhostetler
@mjcheetham
@prplr



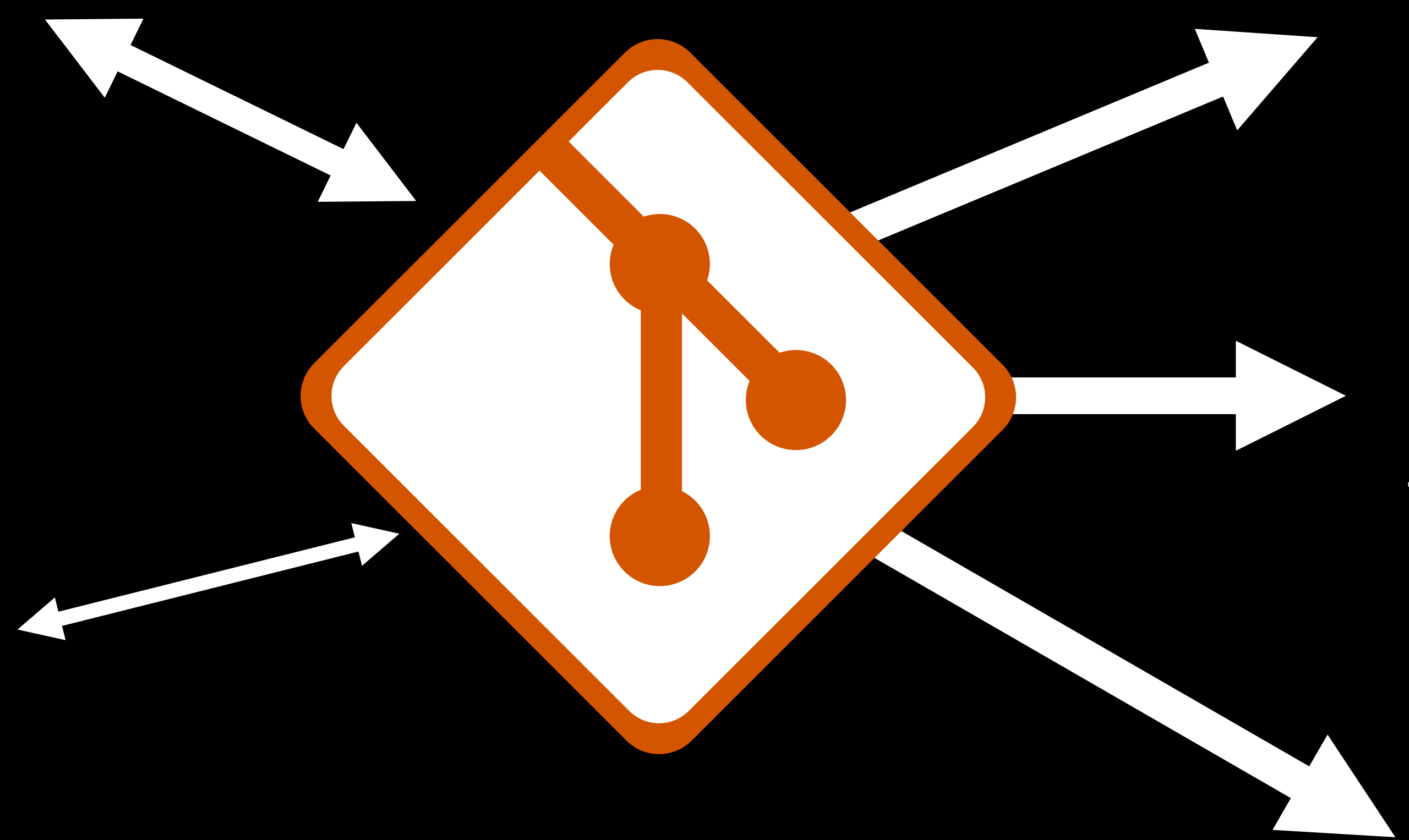
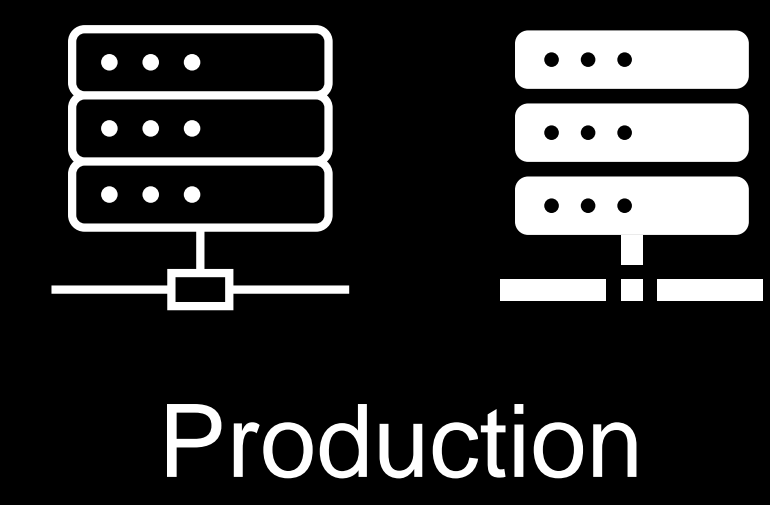
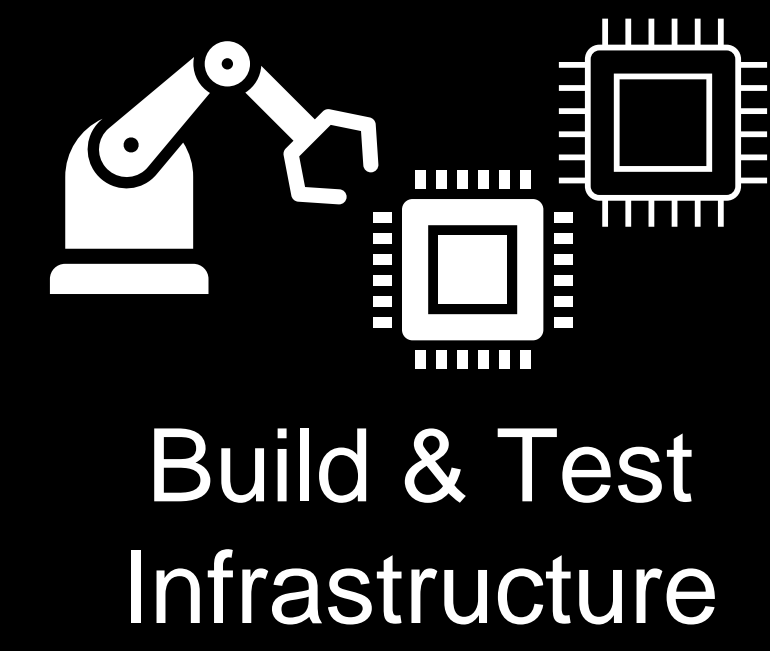
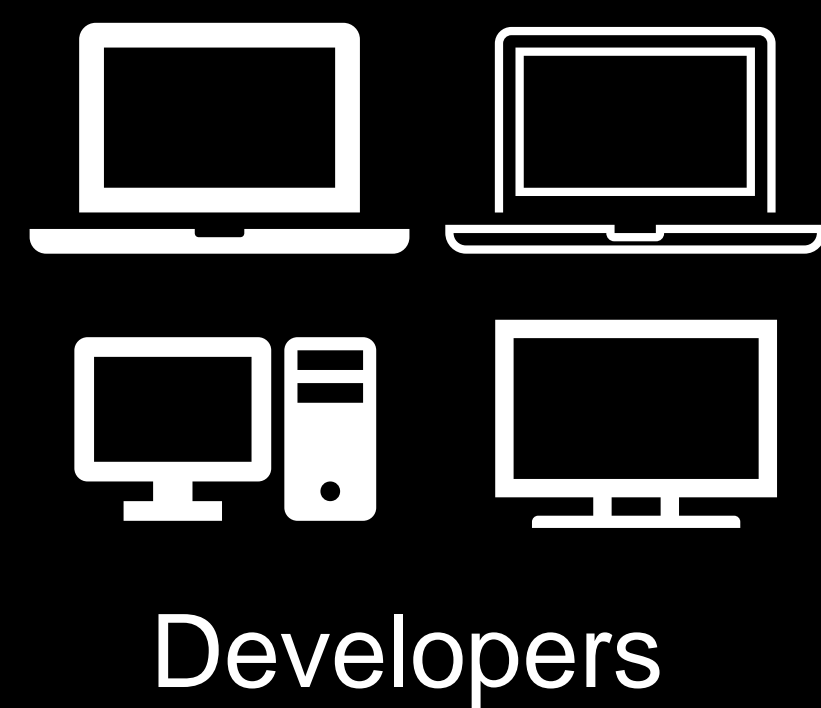
Git Contrib

@peff
@ttaylorr



Git Systems

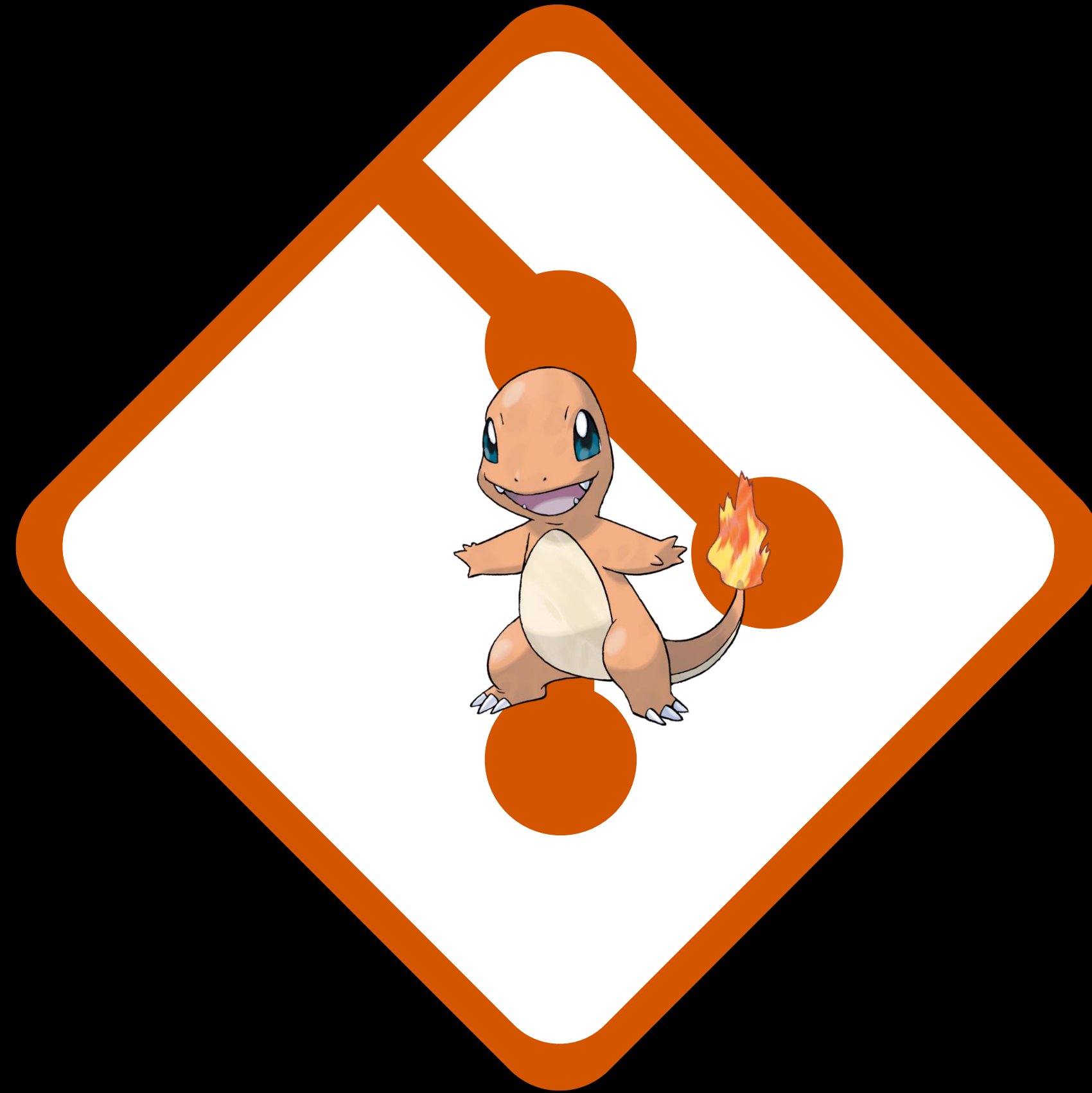
@bk2204
@dkunkler
@mhagger
@wrighty



How do monorepos grow?



Growing in Git



Growing in Git



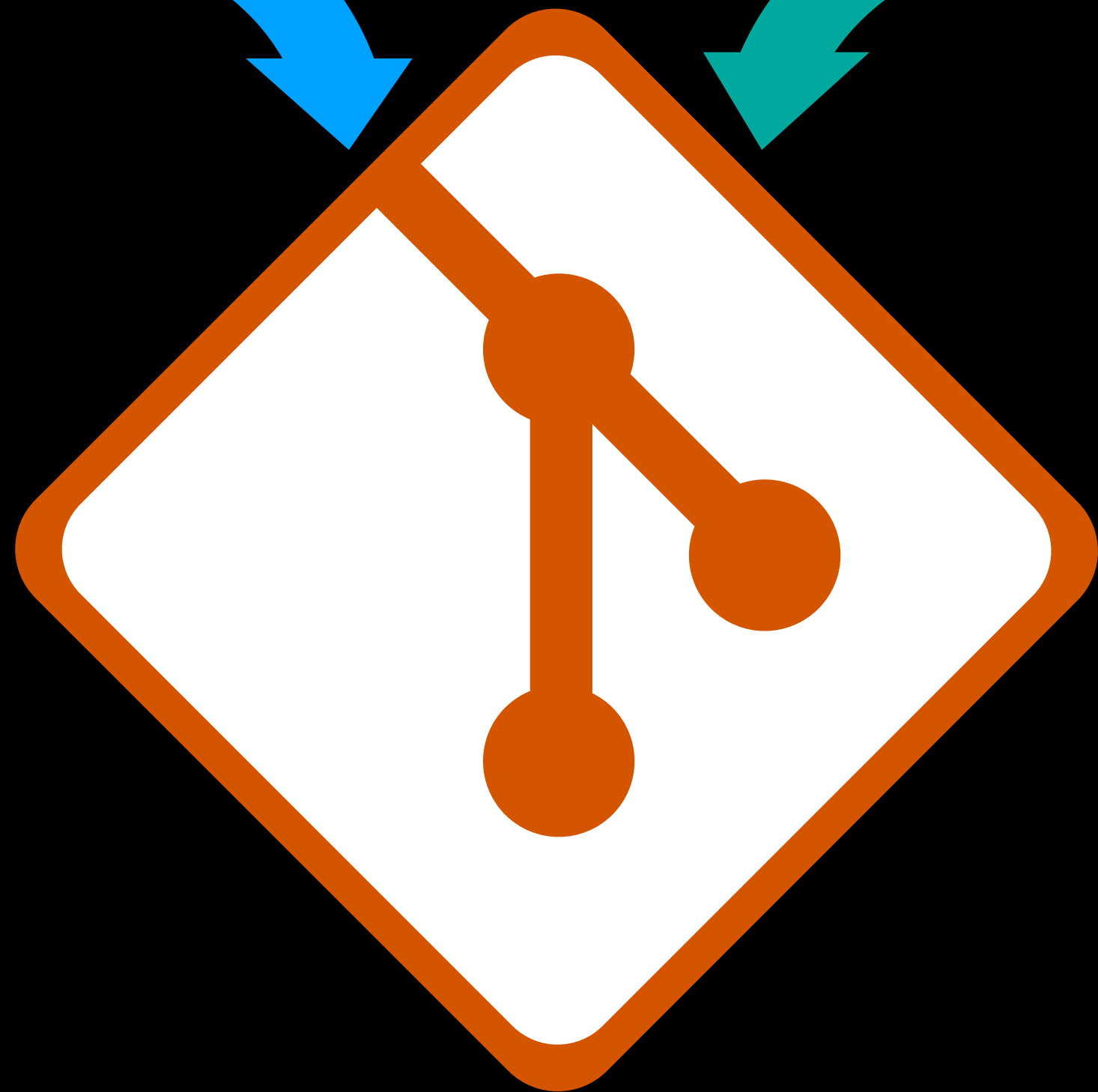
Growing in Git



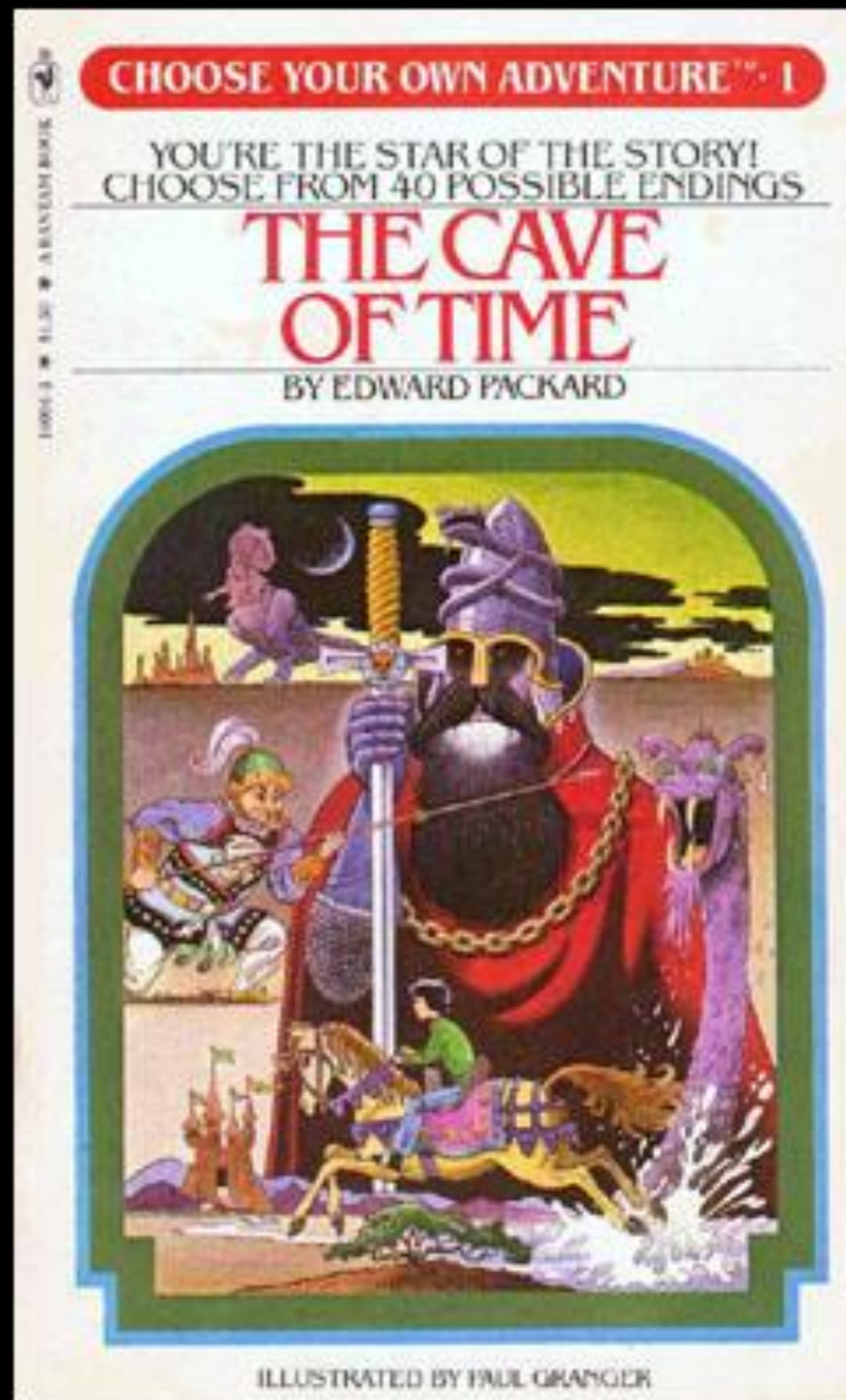
Growing in Git



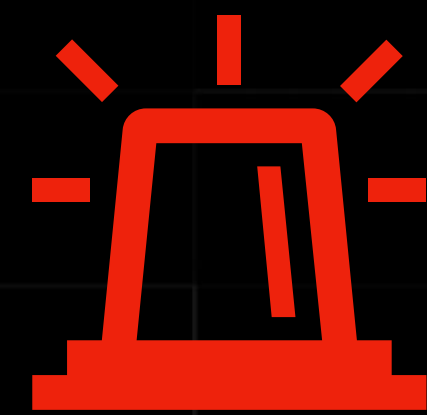
Migrating to Git



Your choices matter!



“



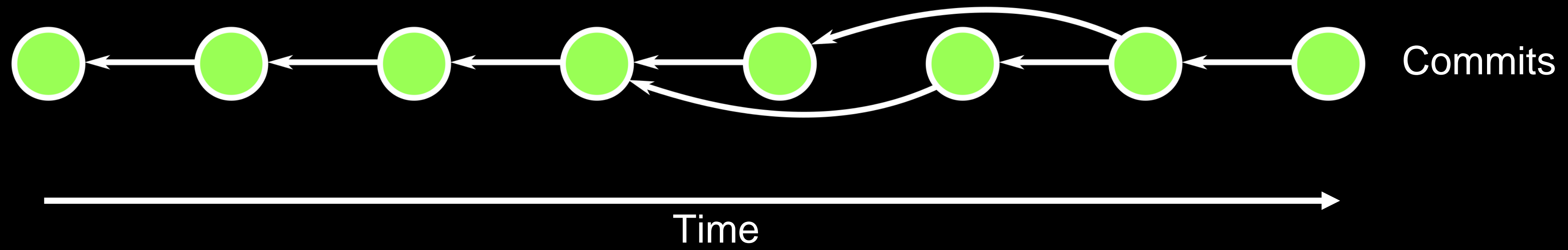
HOT TAKE

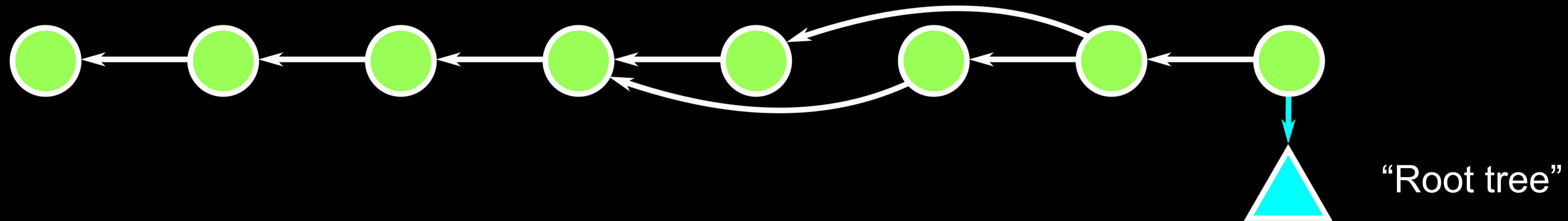


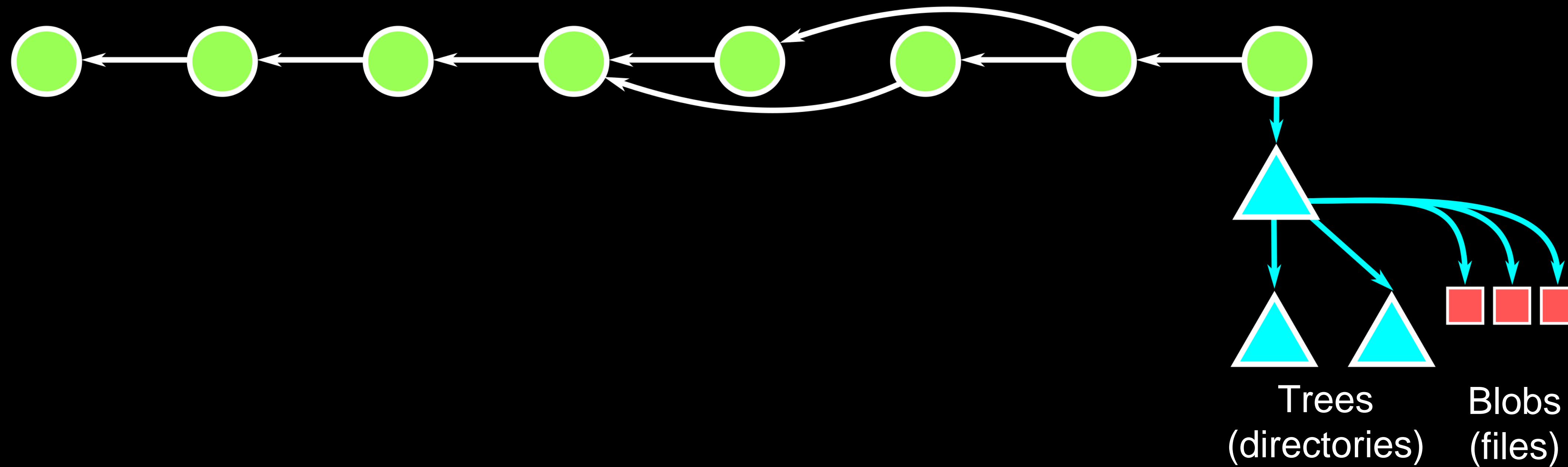
Code never gets *faster*,
it can only do *fewer things*.

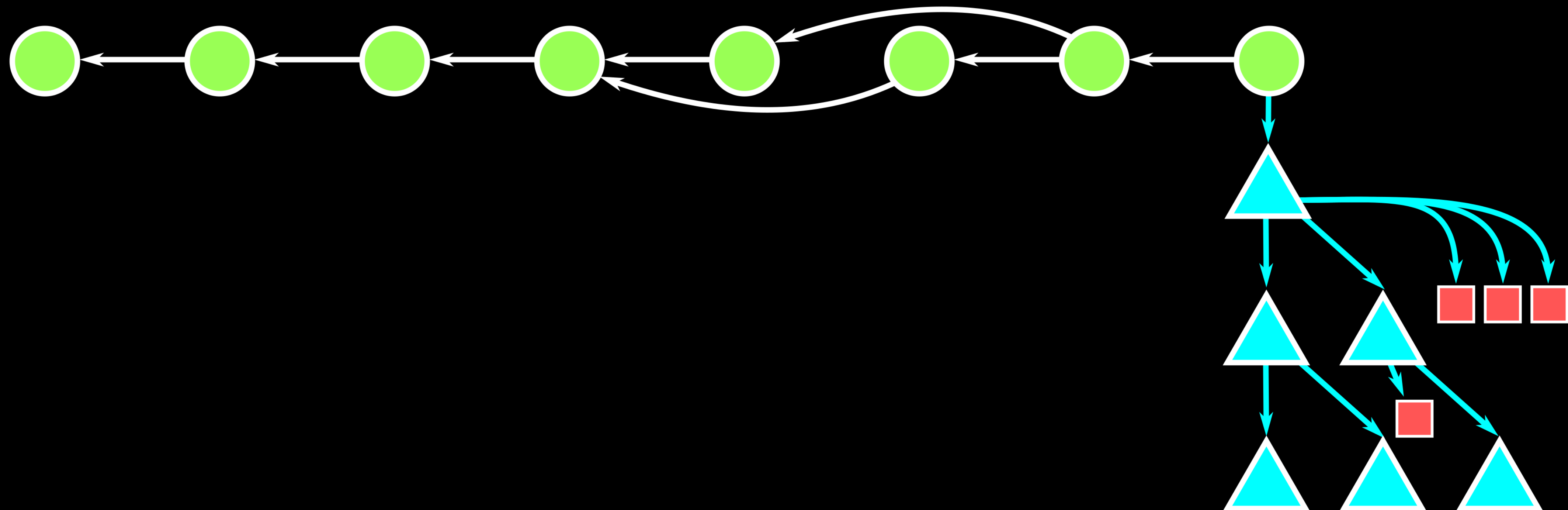
IT'S DANGEROUS TO GO
ALONE! TAKE THIS.

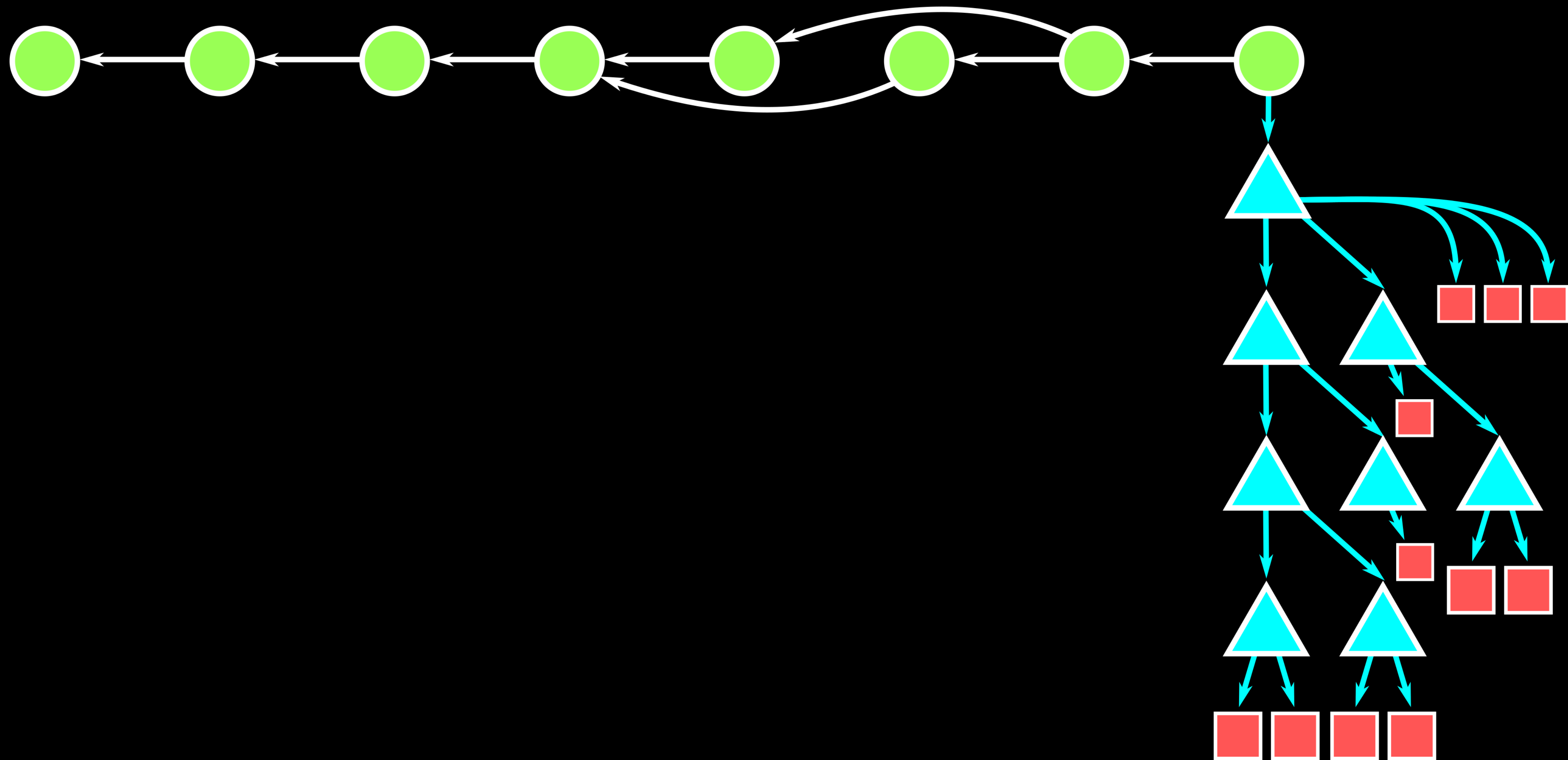


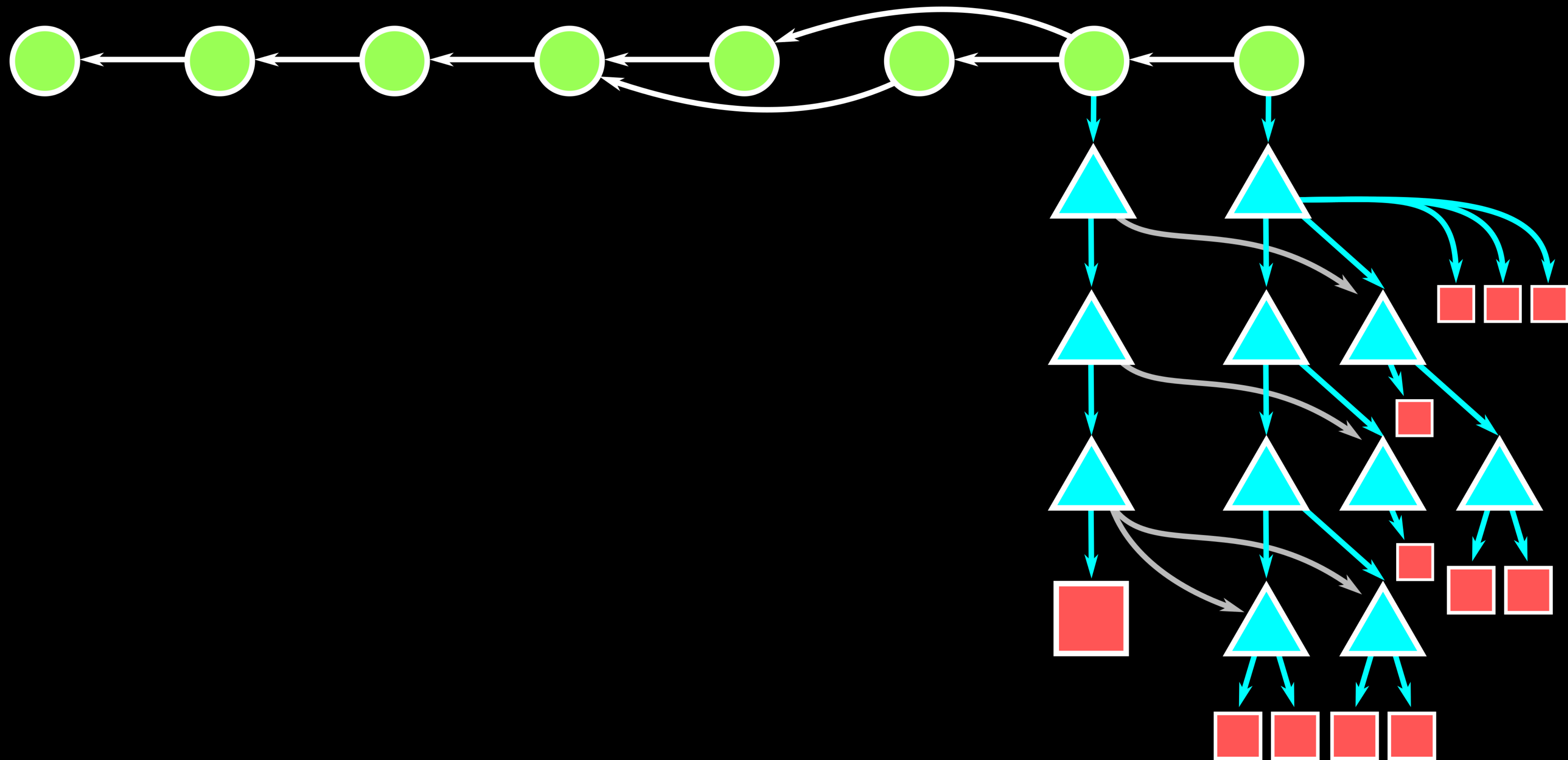


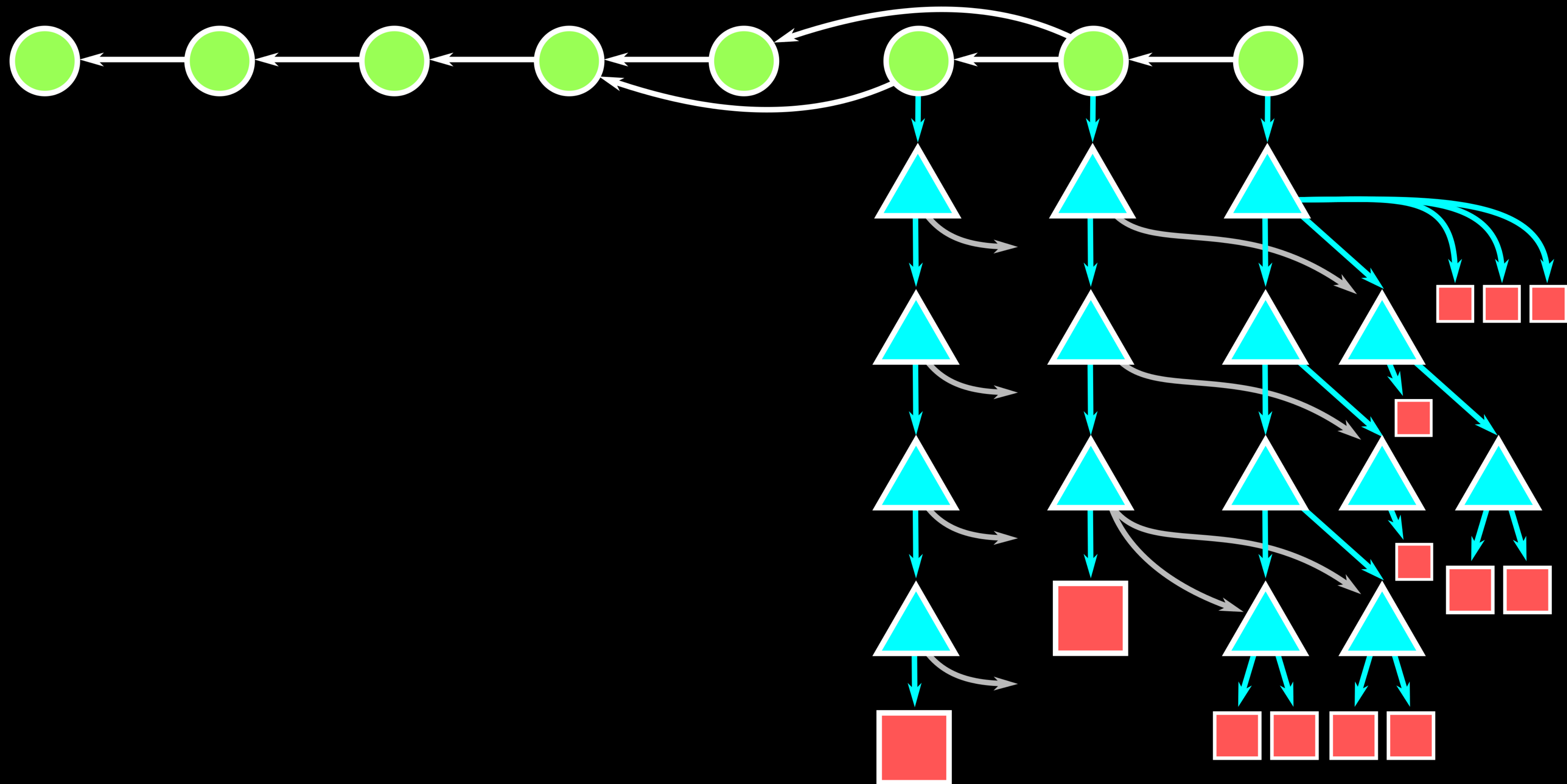


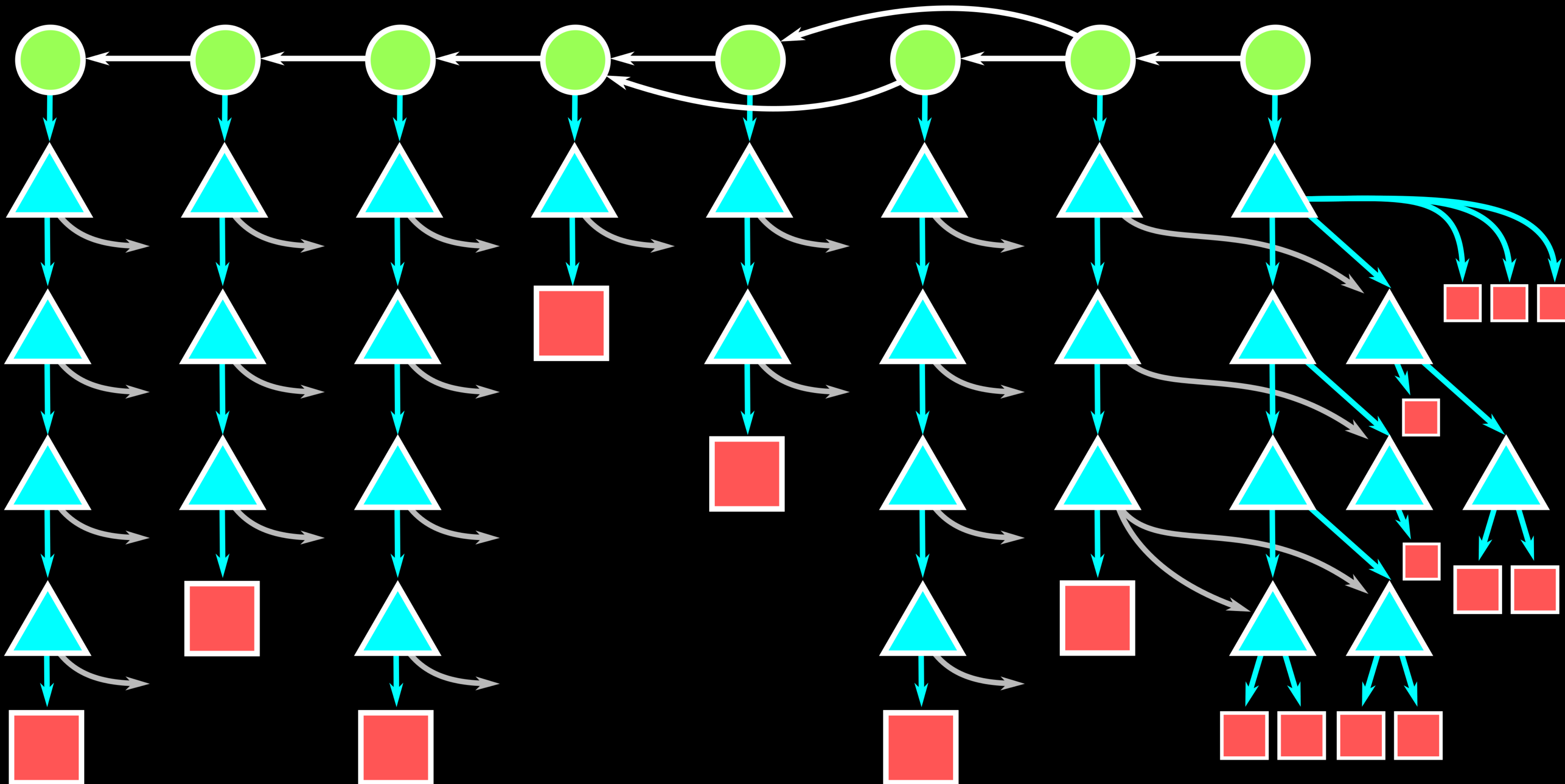












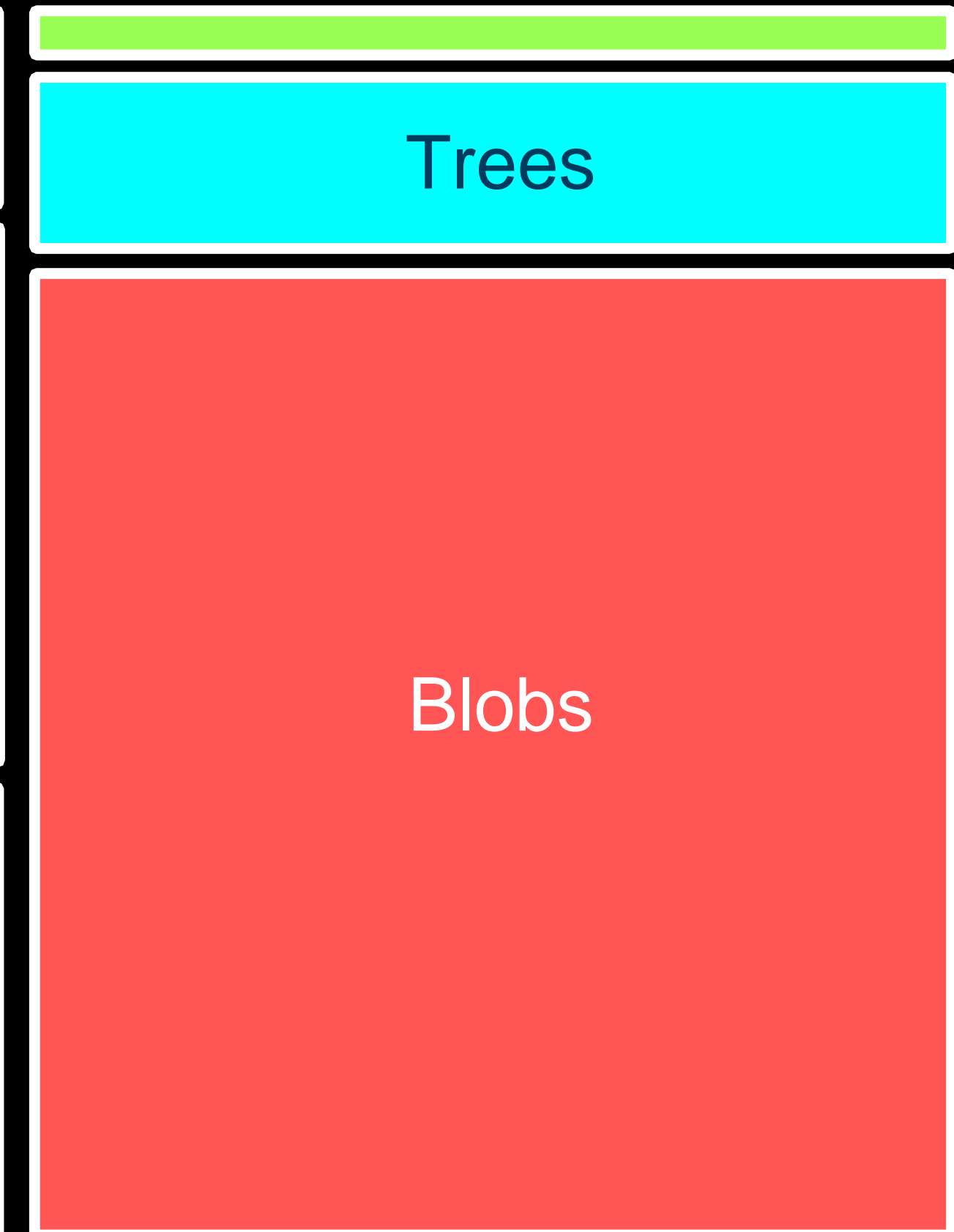
Ratio by Number



Ratio by Size (Good)



Ratio by Size (Worse)



https://github.com/github/git-sizer

github/git-sizer: Compute various statistics about a repository

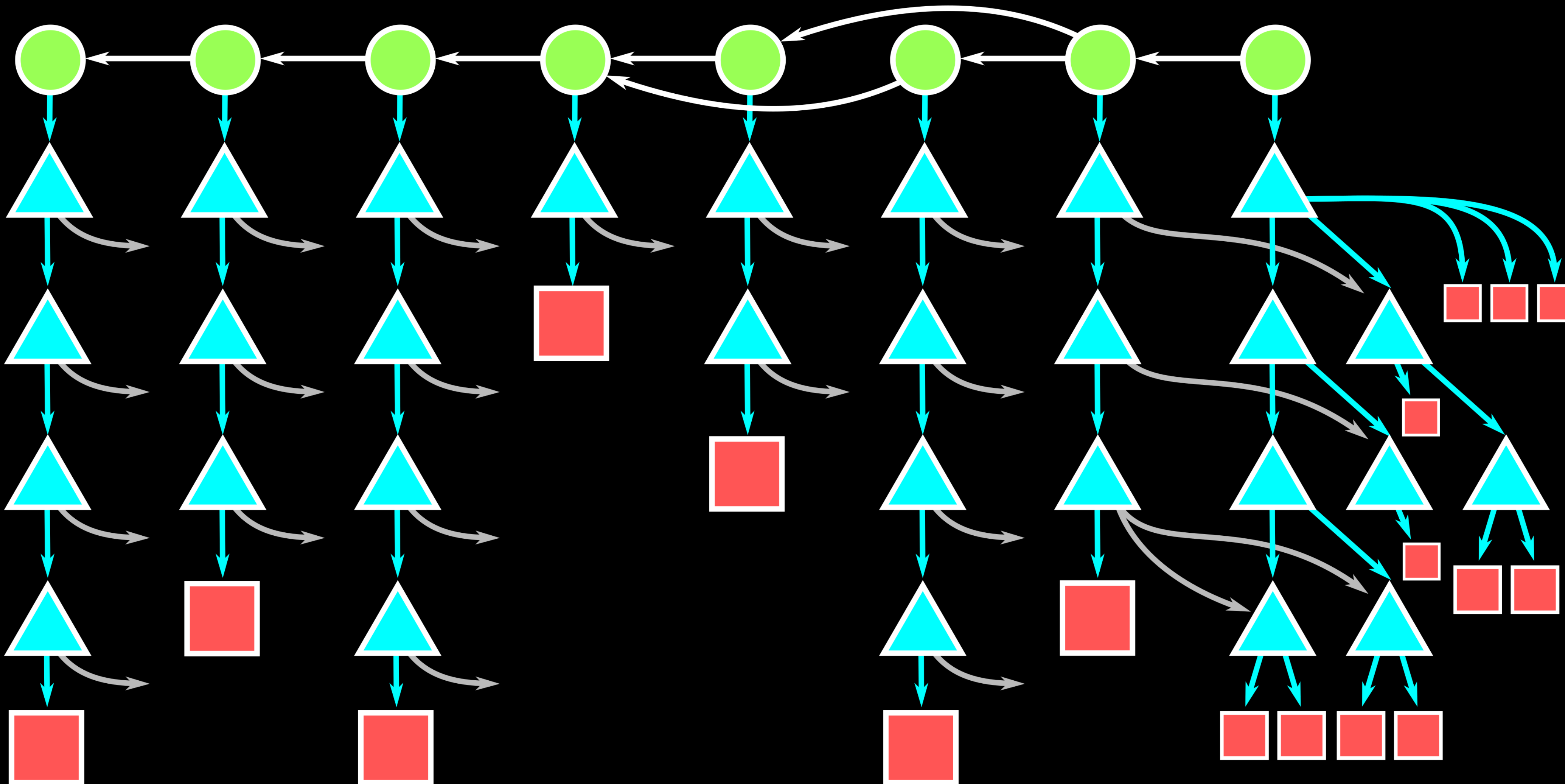
github.com/github/git-sizer#usage

Usage

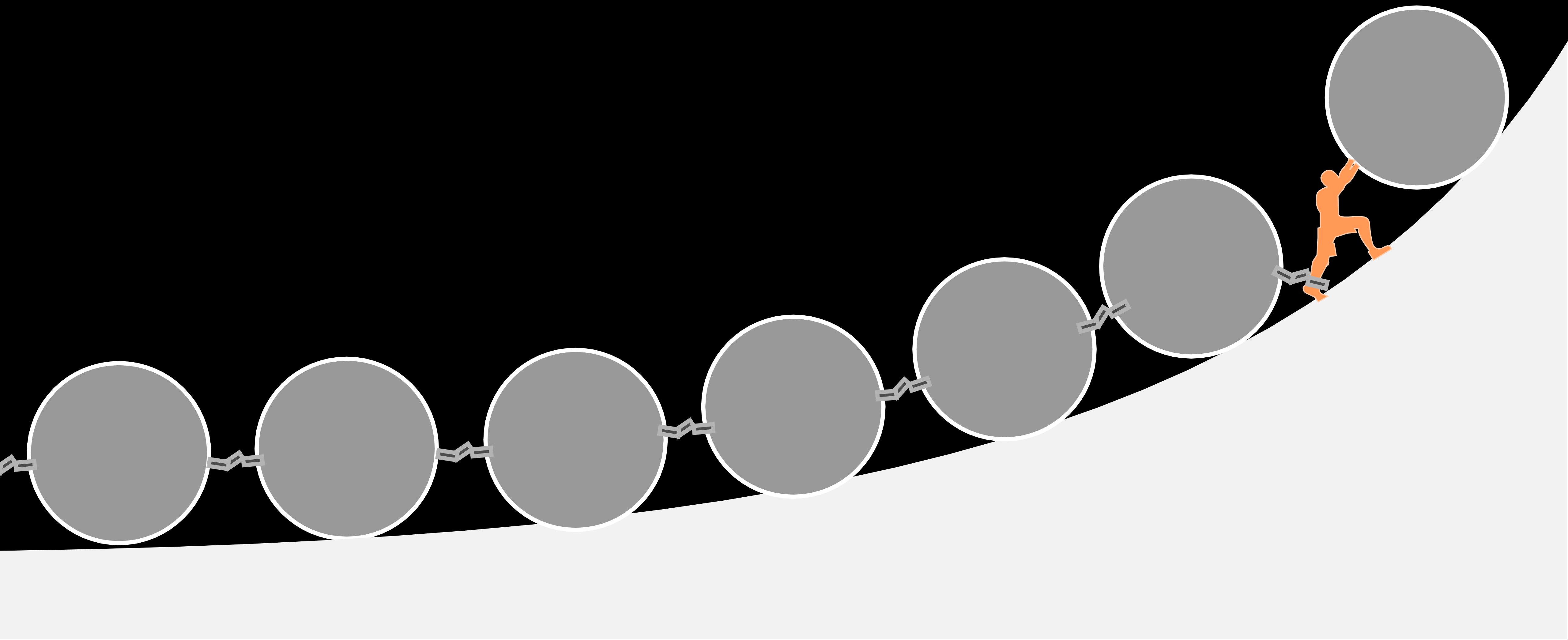
By default, `git-sizer` outputs its results in tabular format. For example, let's use it to analyze [the Linux repository](#), using the `--verbose` option so that all statistics are output:

```
$ git-sizer --verbose
Processing blobs: 1652370
Processing trees: 3396199
Processing commits: 722647
Matching commits to trees: 722647
Processing annotated tags: 534
Processing references: 539
```

| Name | Value | Level of concern |
|-------------------------|----------|------------------|
| Overall repository size | | |
| * Commits | | |
| * Count | 723 k | * |
| * Total size | 525 MiB | ** |
| * Trees | | |
| * Count | 3.40 M | ** |
| * Total size | 9.00 GiB | **** |
| * Total tree entries | 264 M | ***** |
| * Blobs | | |
| * Count | 1.65 M | * |
| * Total size | 55.8 GiB | ***** |
| * Annotated tags | | |
| * Count | 534 | |



Large blobs dragging you down?

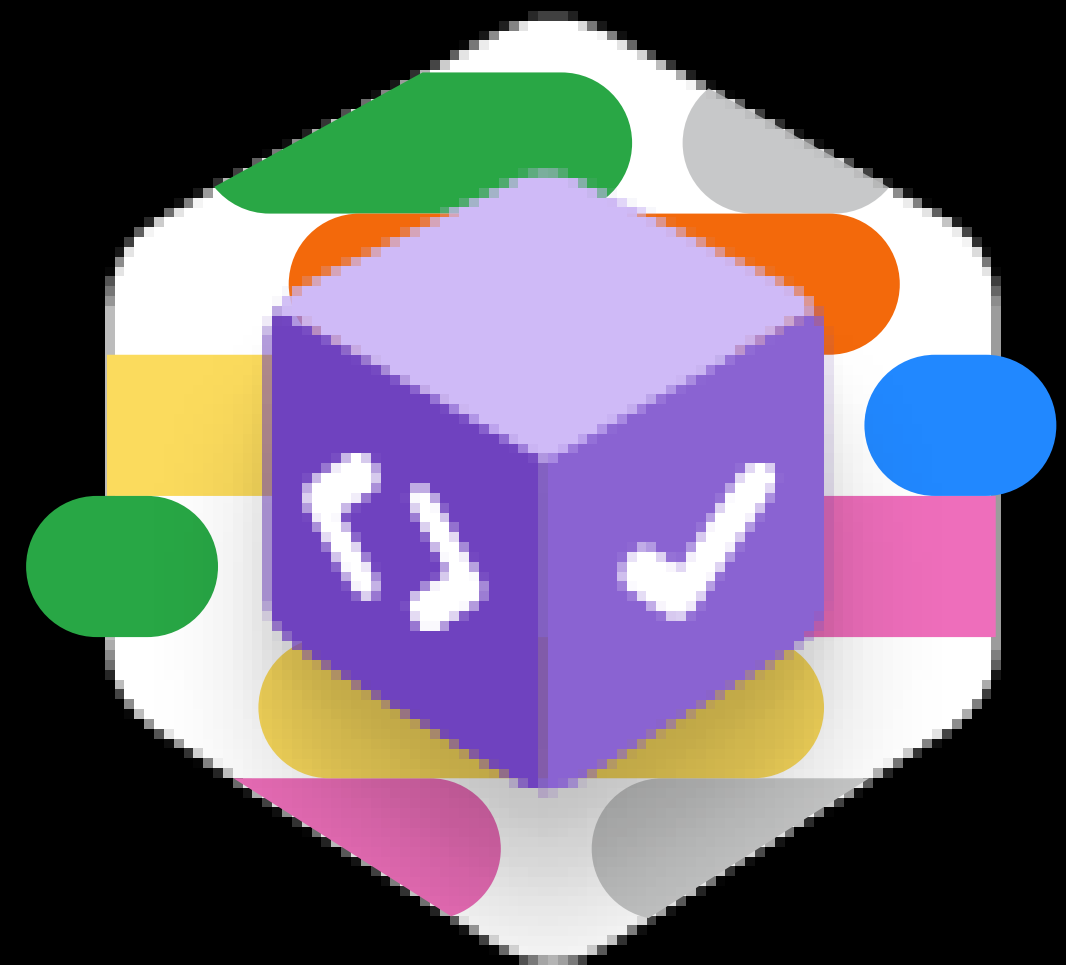


Kick out bad blobs!

Git LFS



GitHub Packages

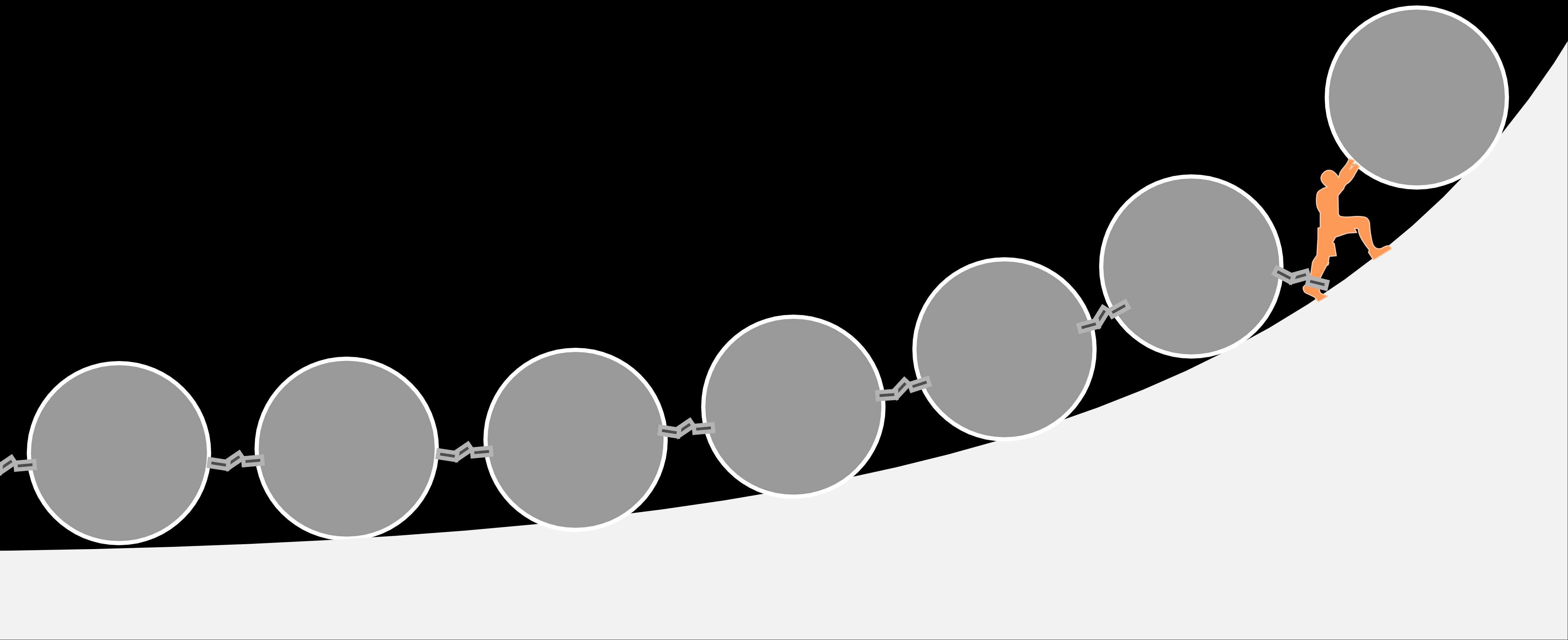


“

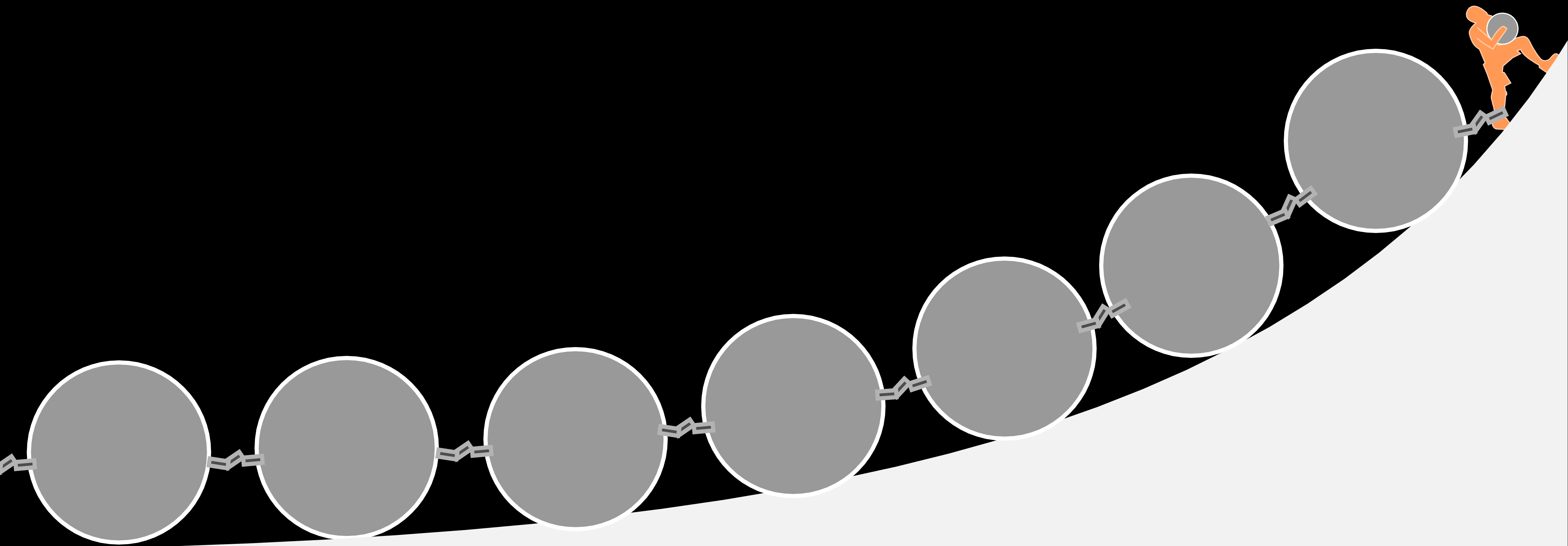
HOT TAKE

If you can't **review it**,
then **delete it!**

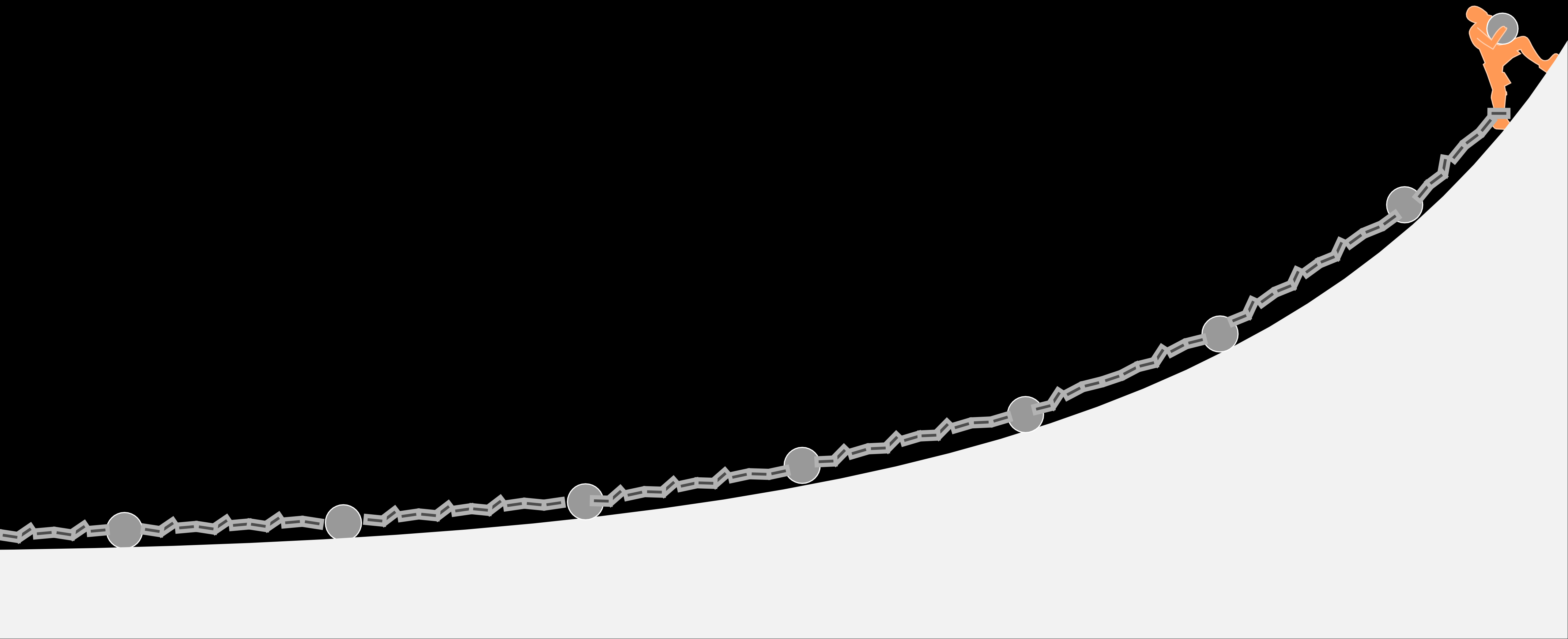
Carry that weight



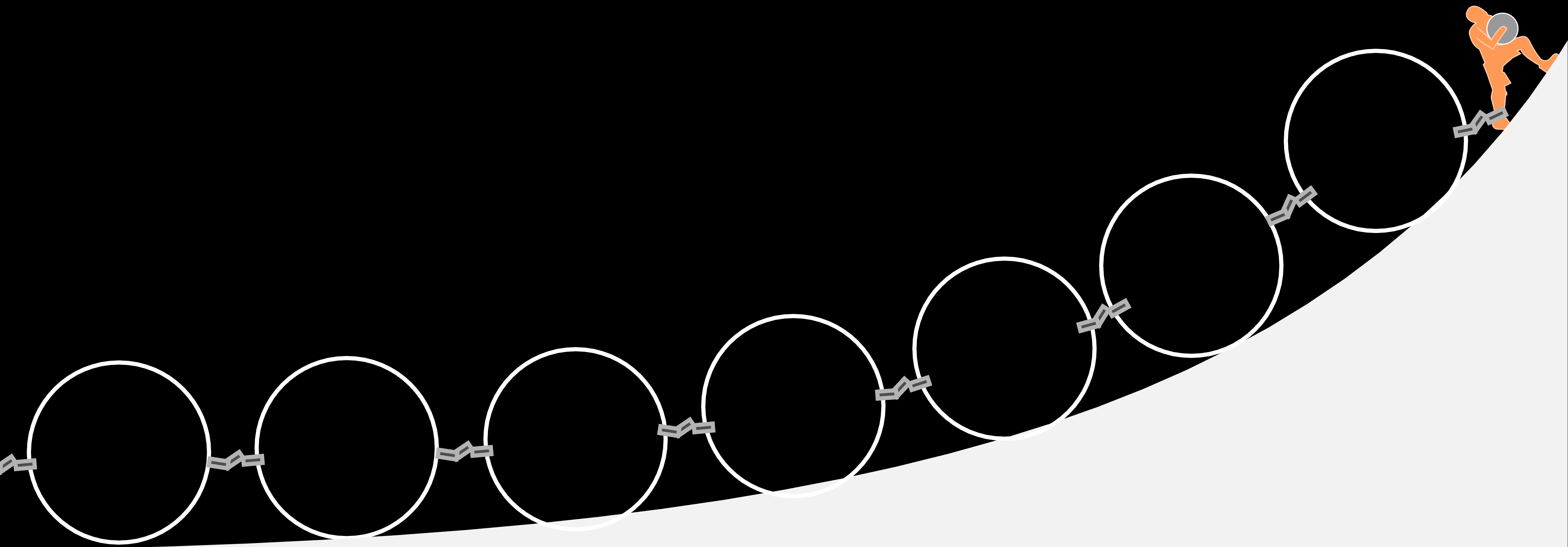
Remove files at tip



Rewrite history?



Lighten the weight



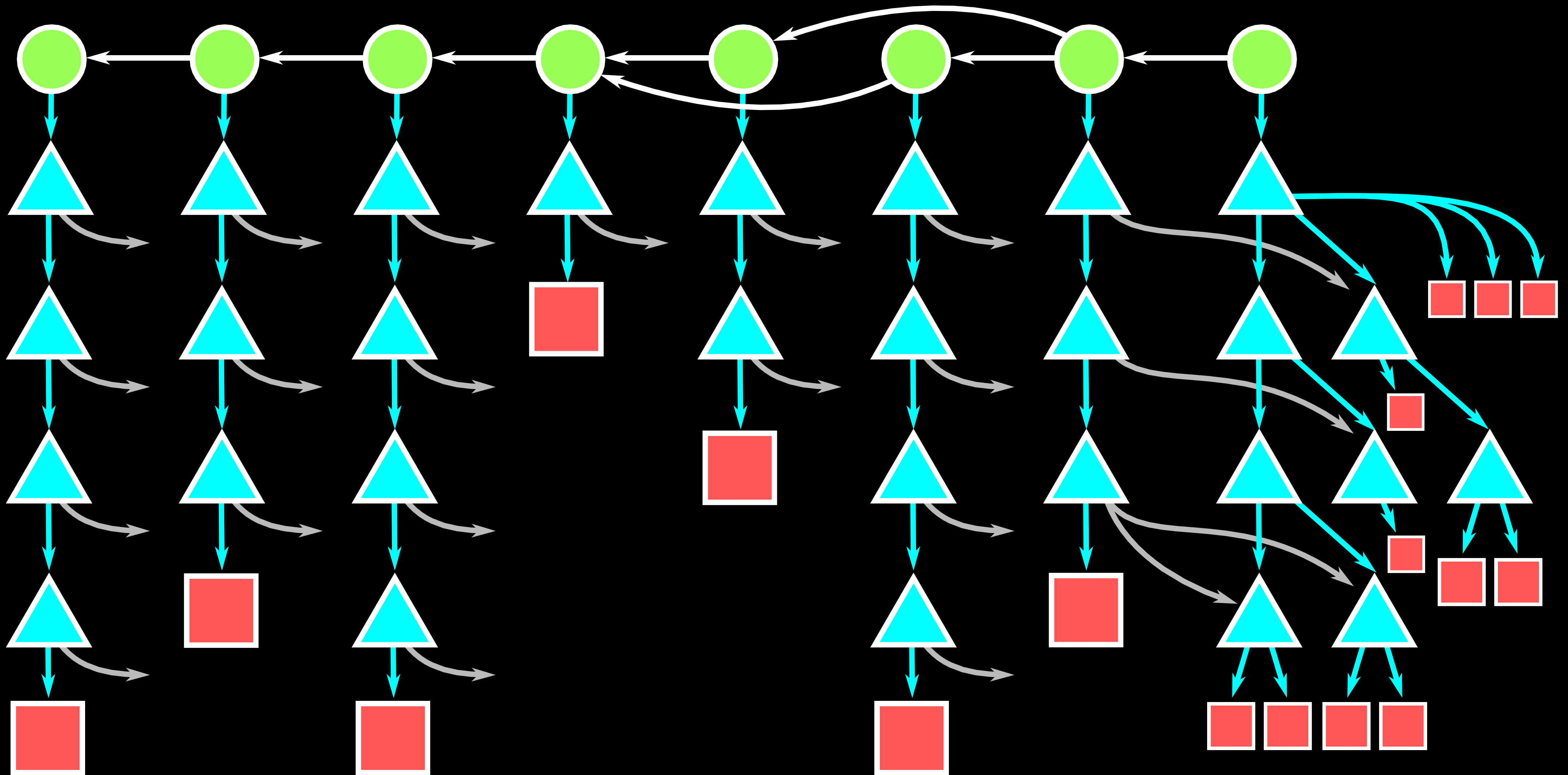
Pro Tip: Use partial clone

```
git clone --filter=blob:none <url>
```

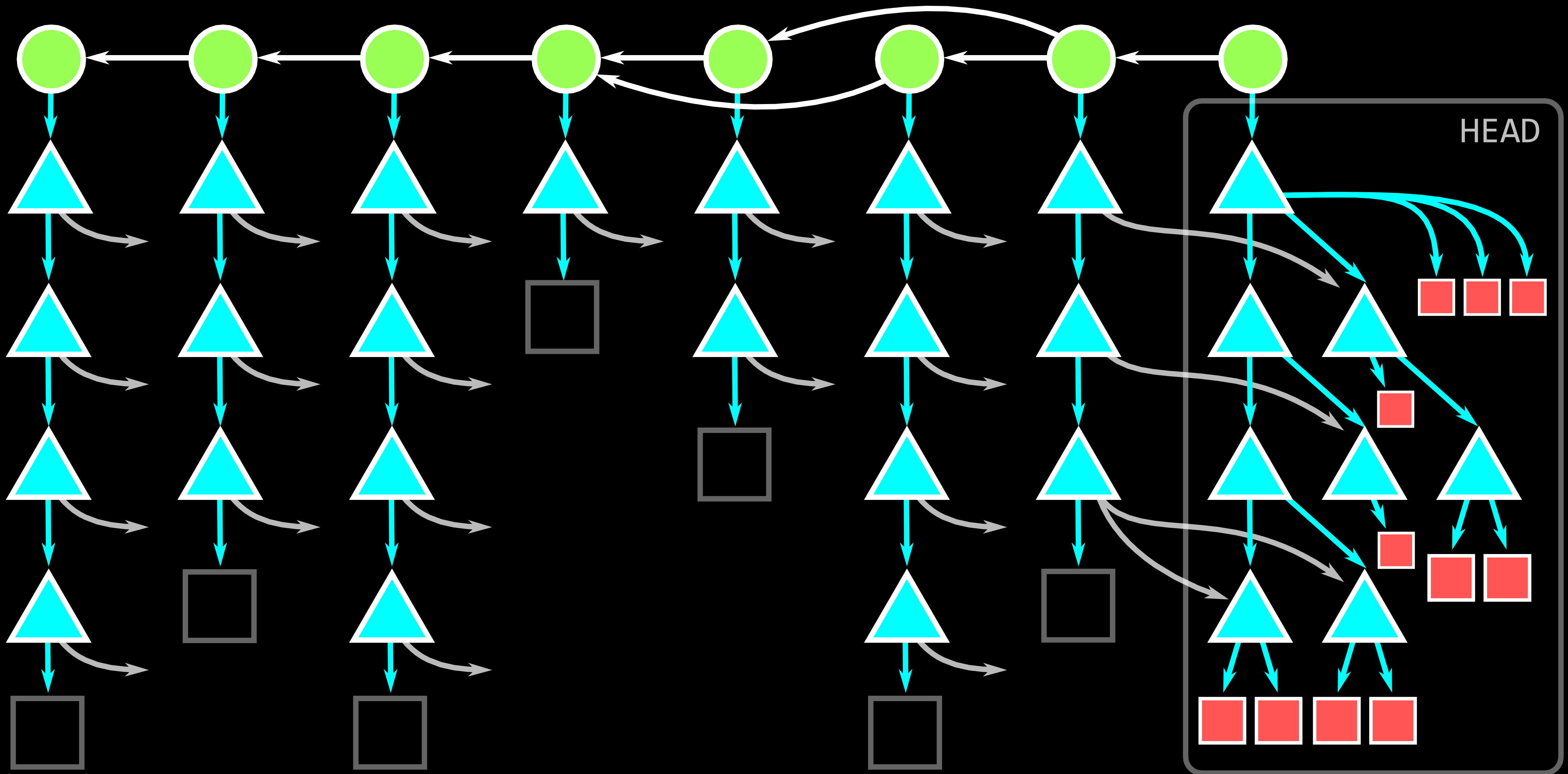
Partial clones are available now!

github.com

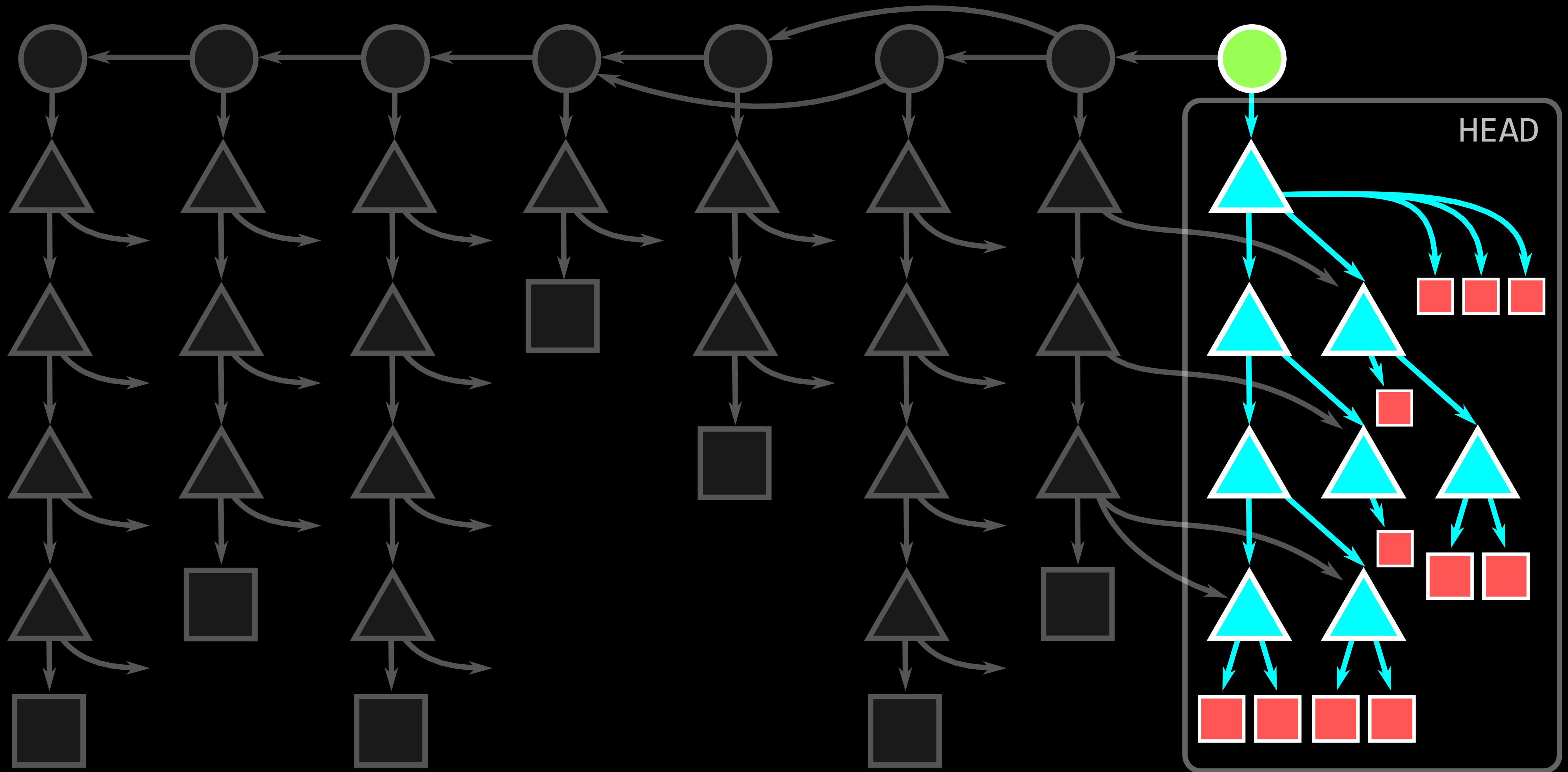
GHES 2.22+



Full Clone: All commits, trees, and blobs

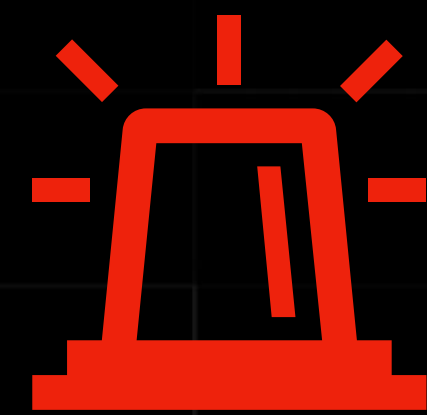


Partial Clone: All commits and trees; *blobs as needed*



Shallow Clone: Trees and blobs for *tip commit*

“



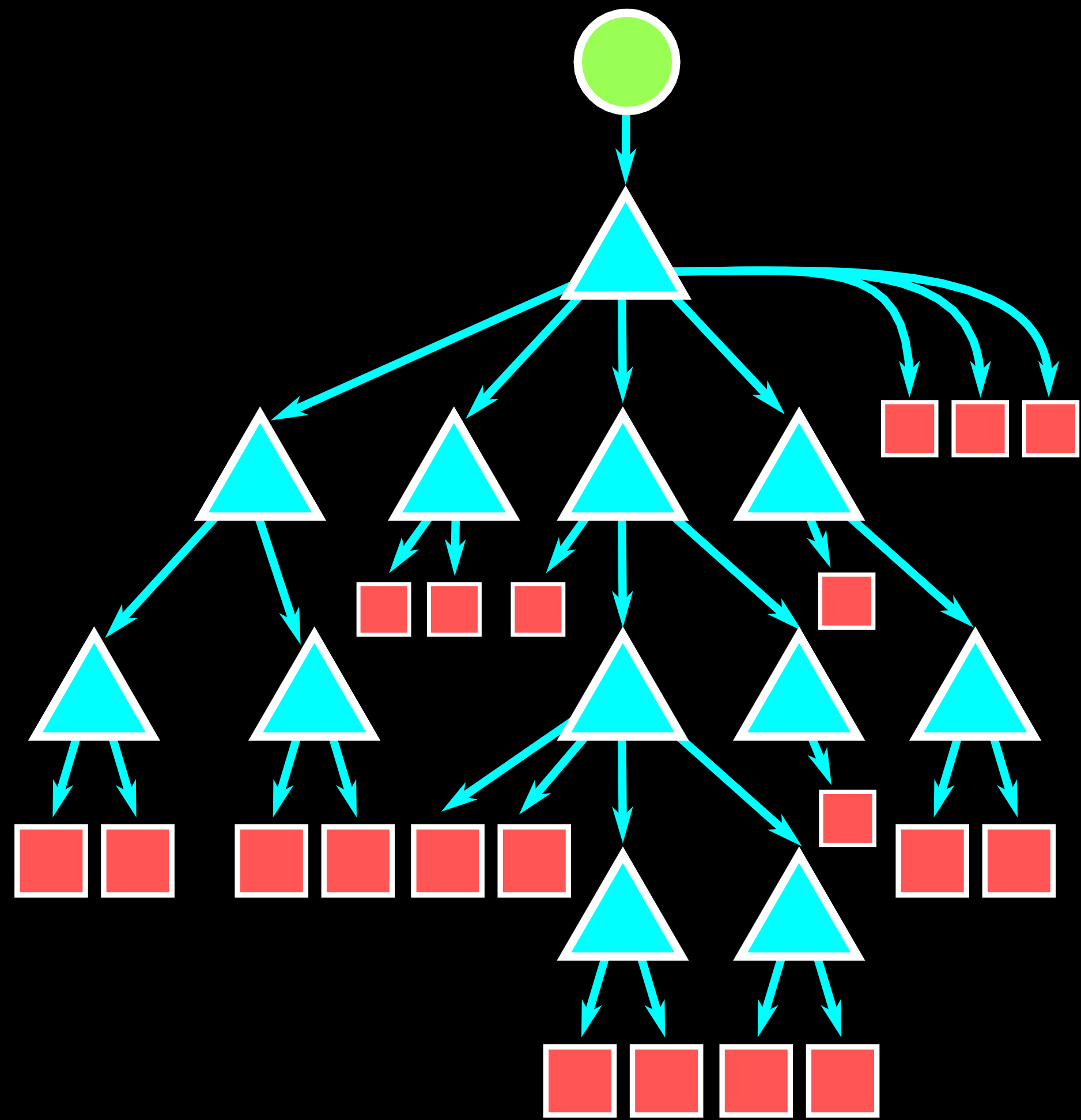
HOT TAKE



Shallow clones should be
thrown away!

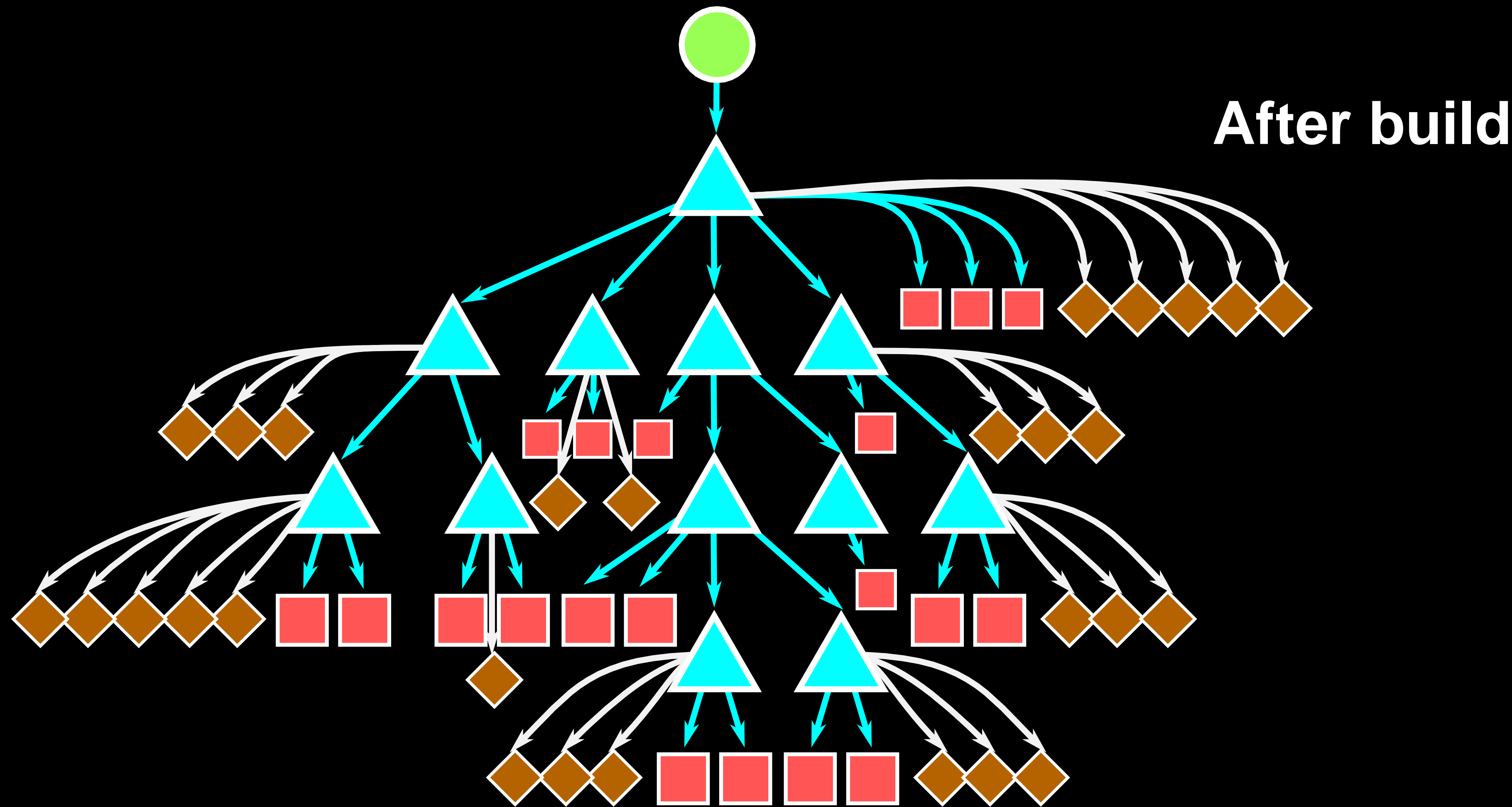


Optimize the Working Directory

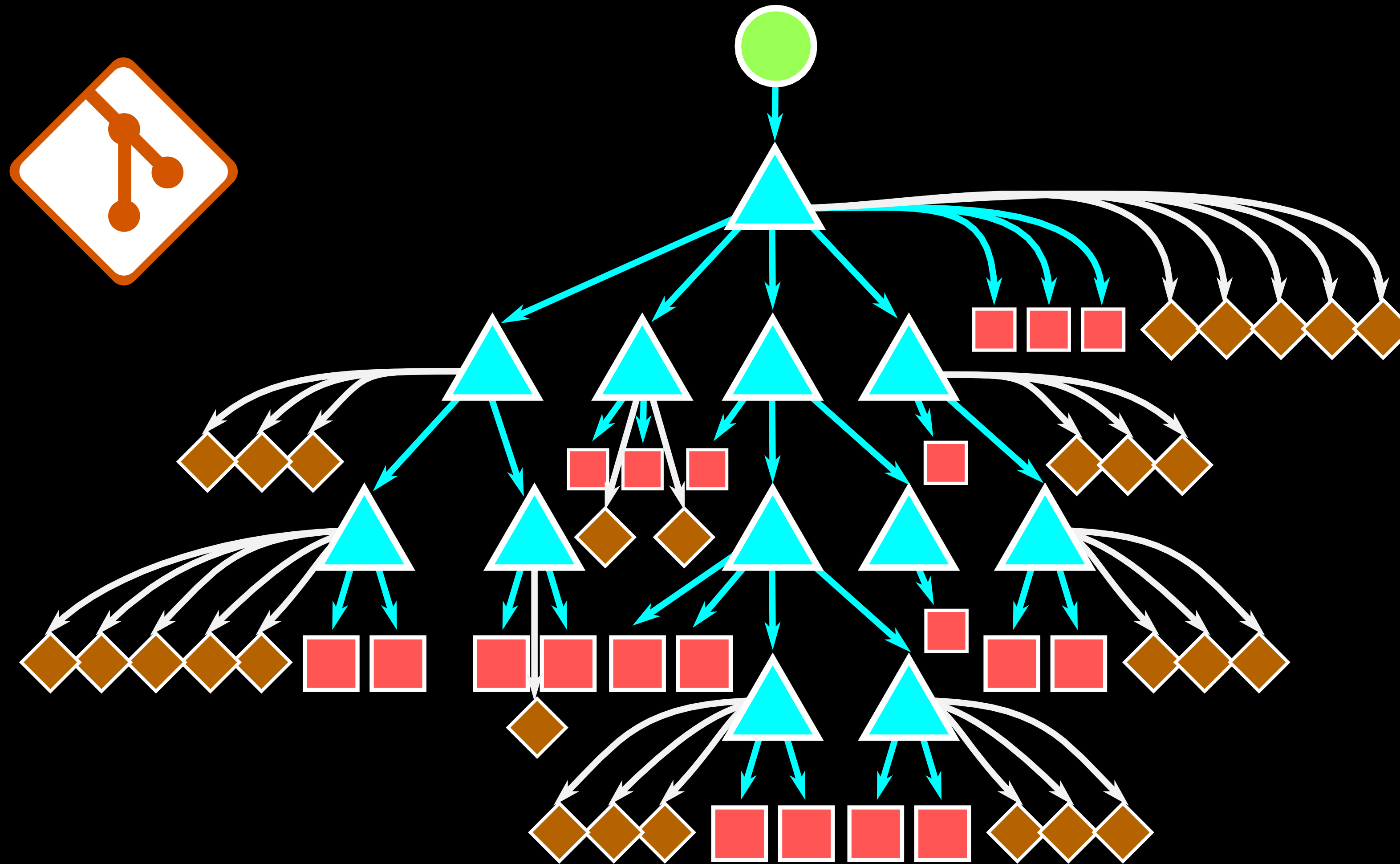


Before build

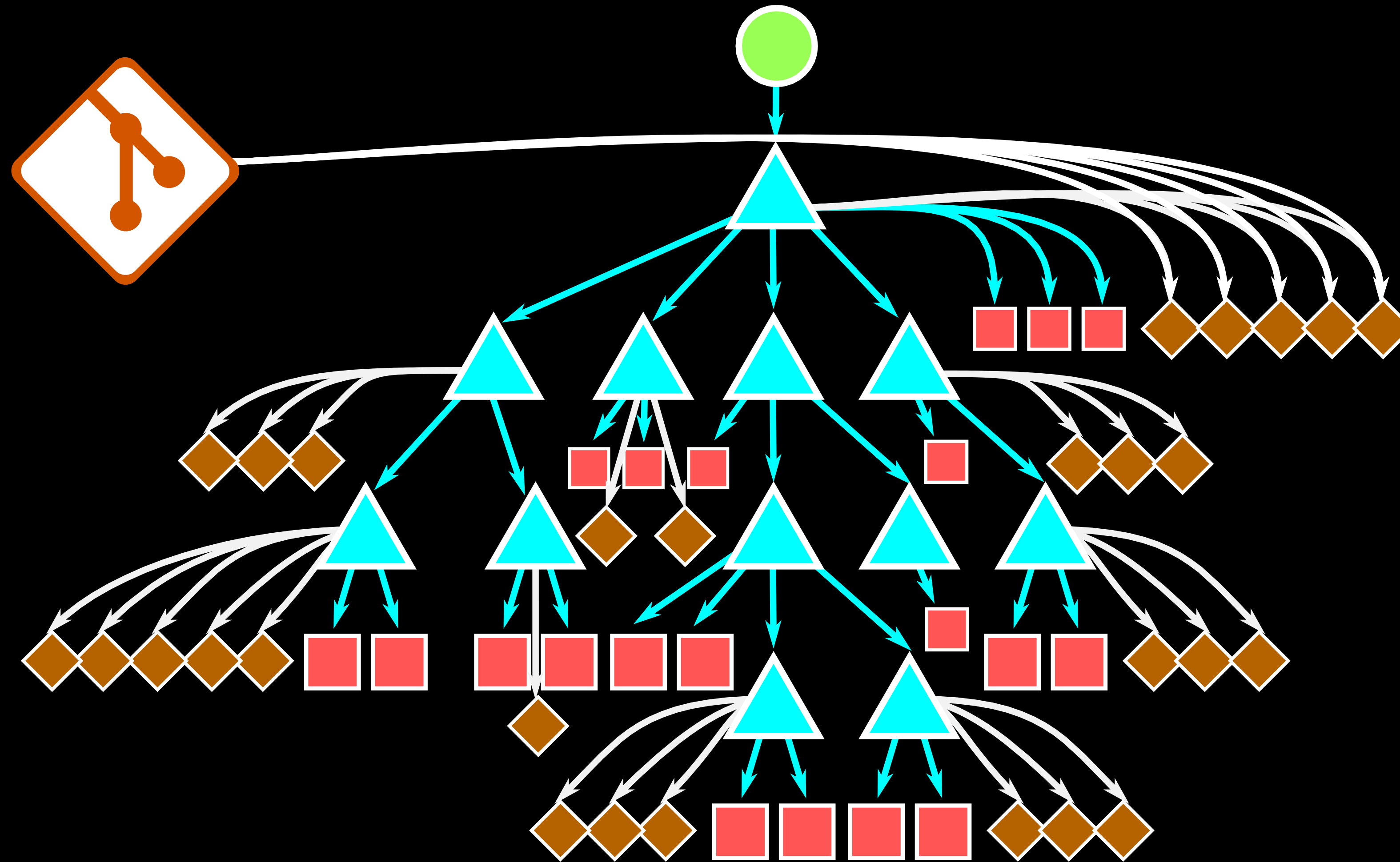
Optimize the Working Directory



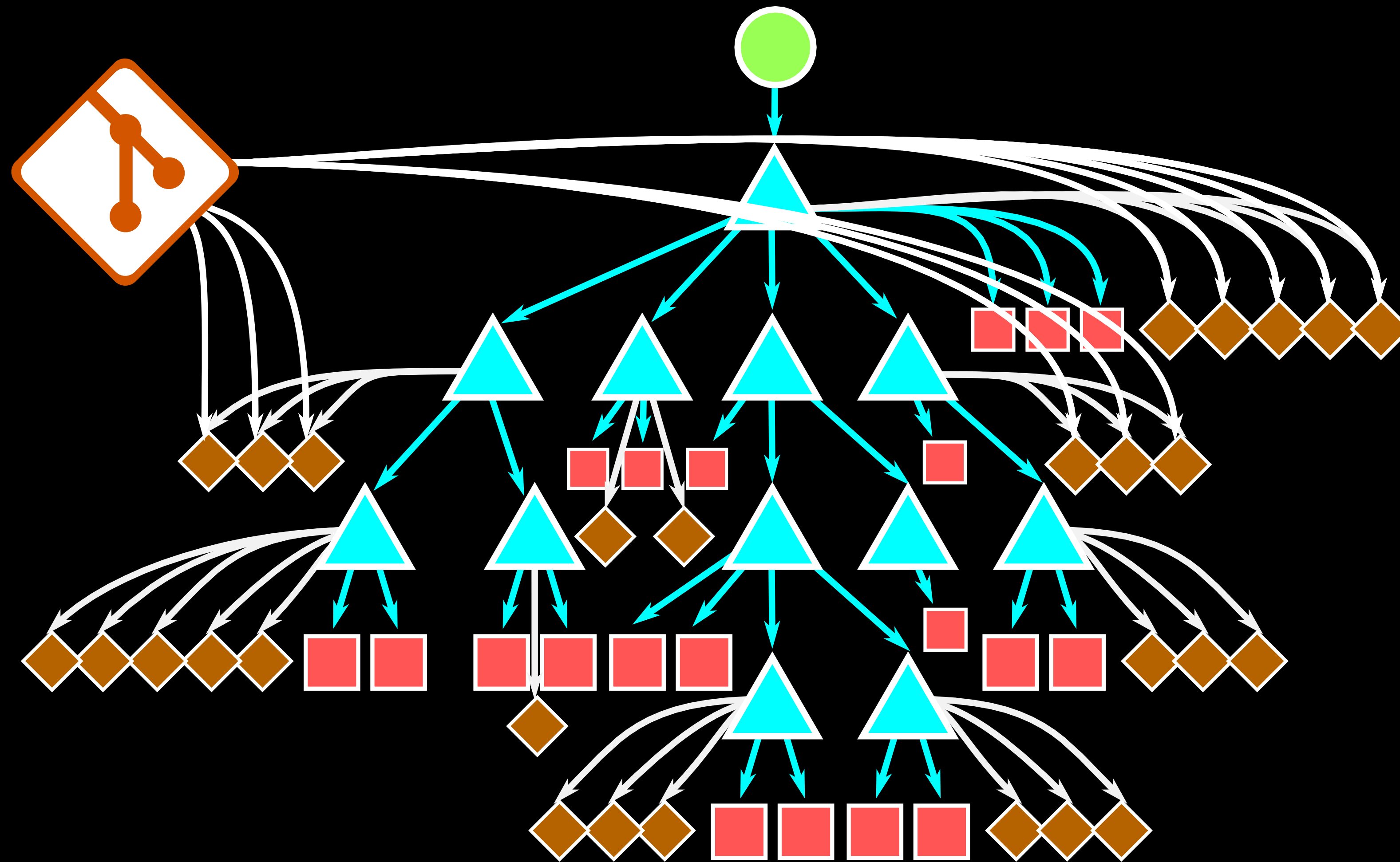
Optimize the Working Directory



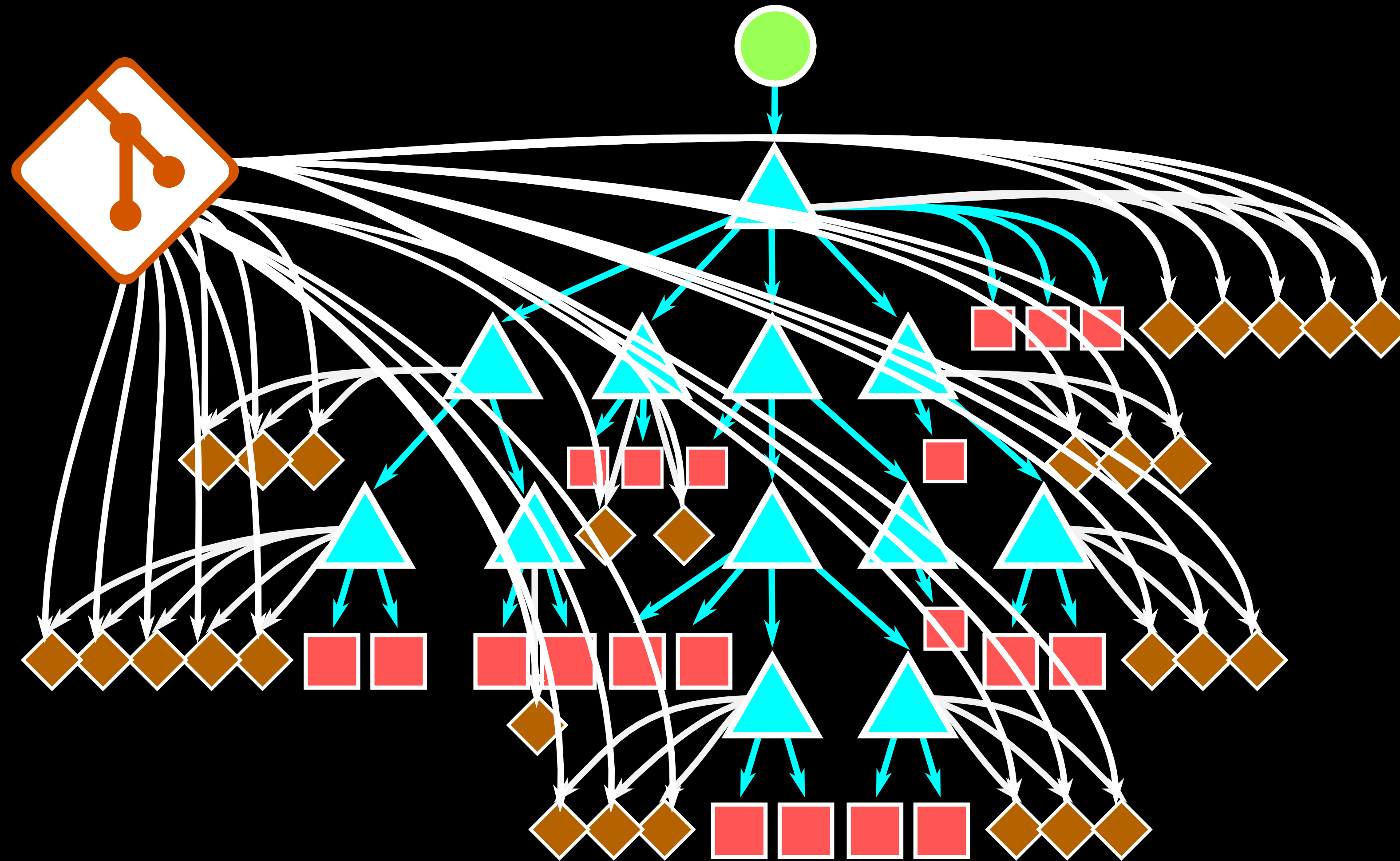
Optimize the Working Directory



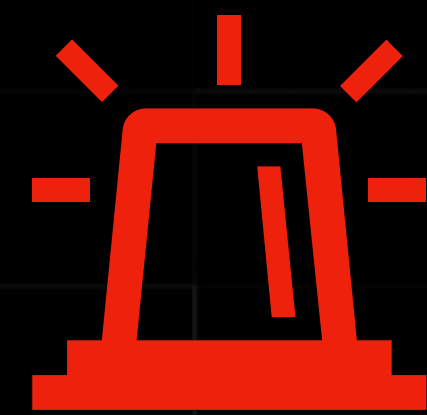
Optimize the Working Directory



Optimize the Working Directory



“

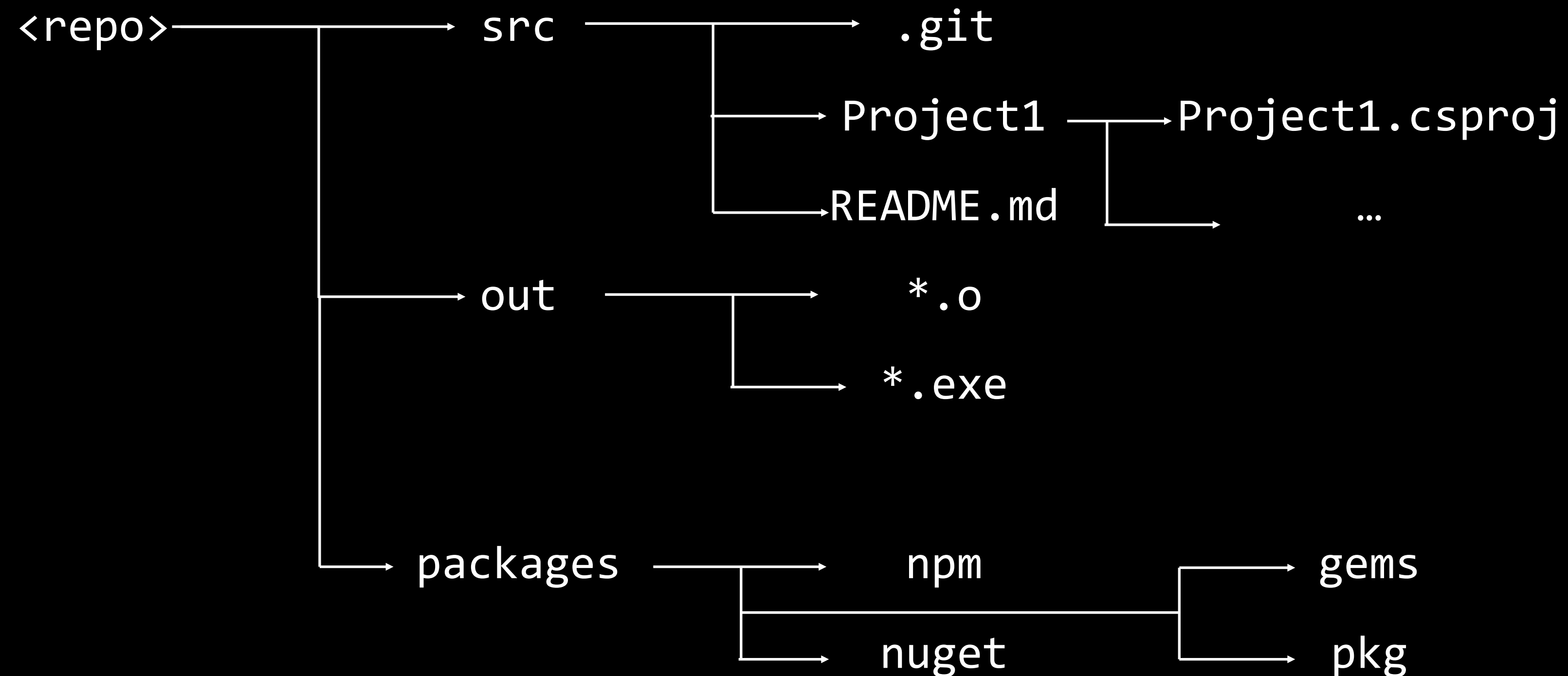


HOT TAKE



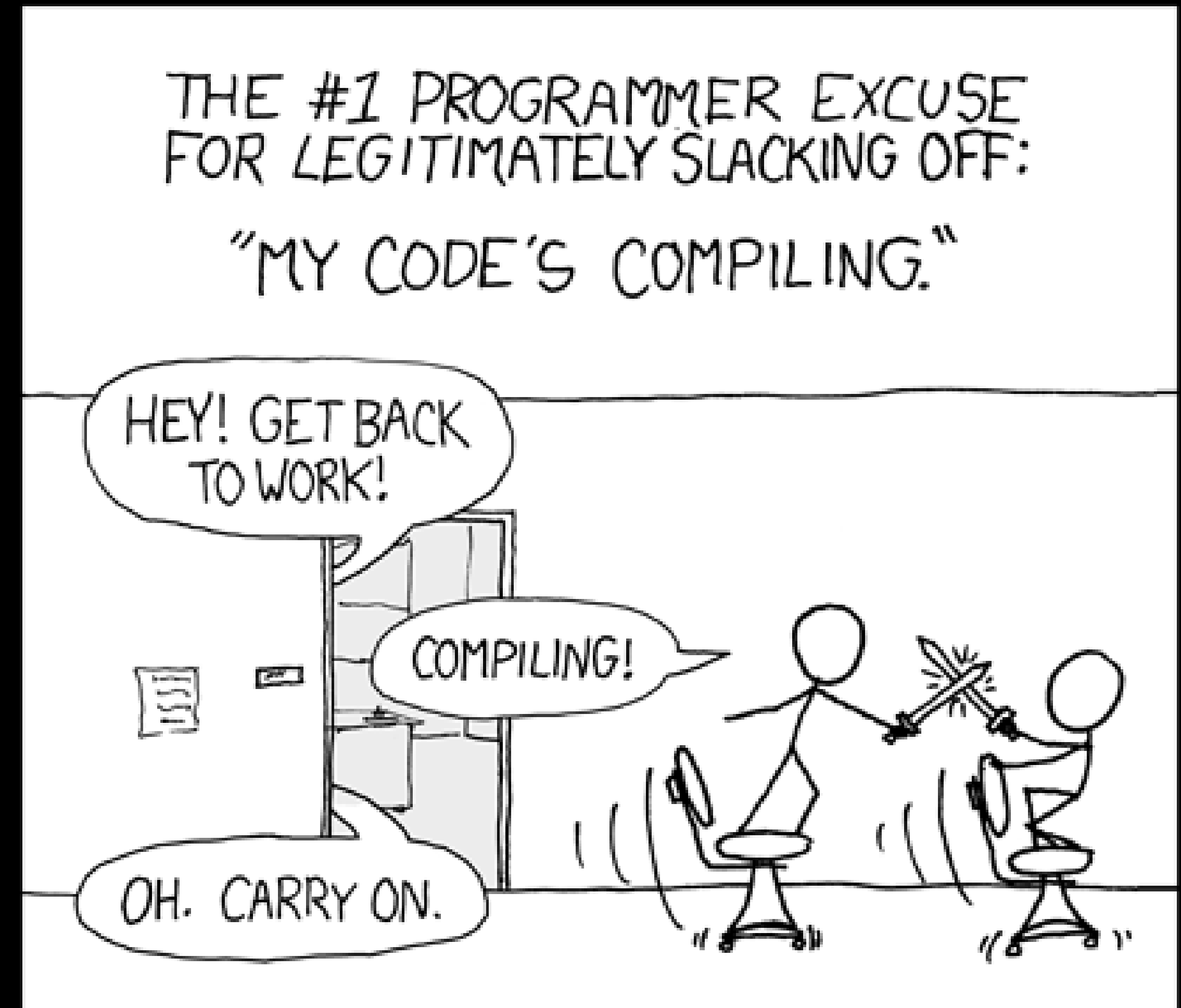
`.gitignore` files
should be *tiny*!

Get your builds out of src!

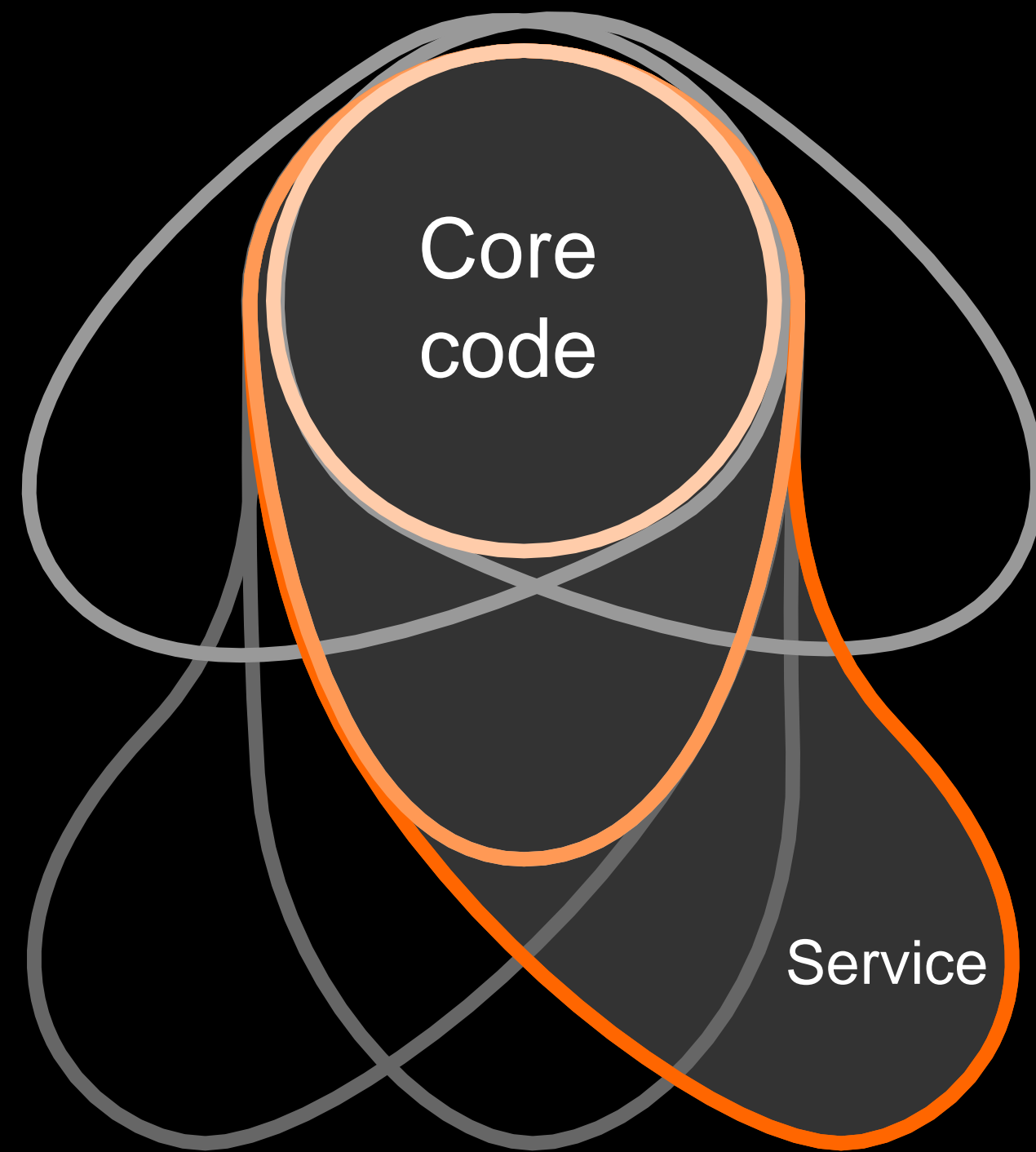


Use: `git clone <url> <repo>/src`

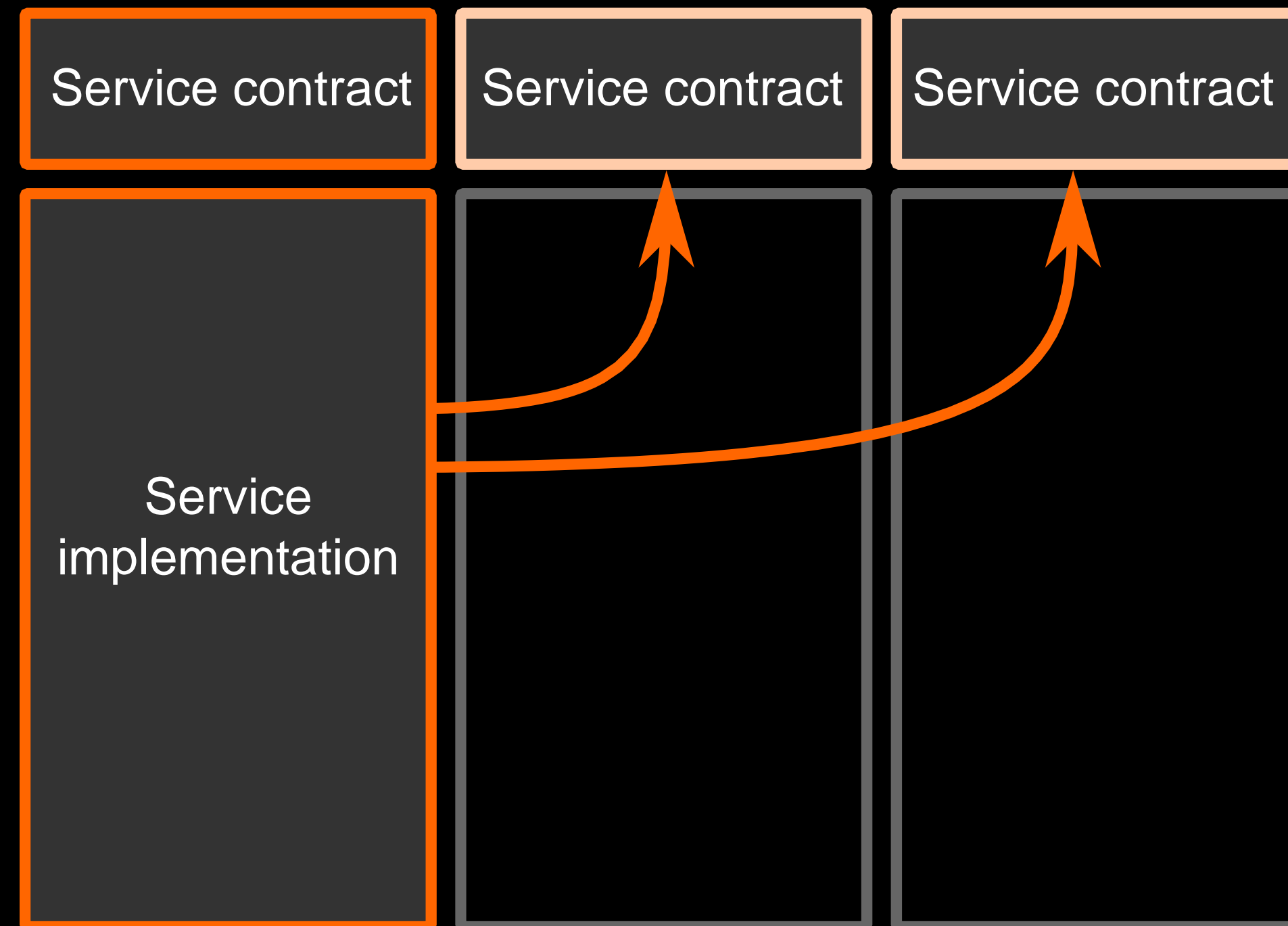
1. Focused builds
2. Focused devs
3. Profit!



Architectures for build cones

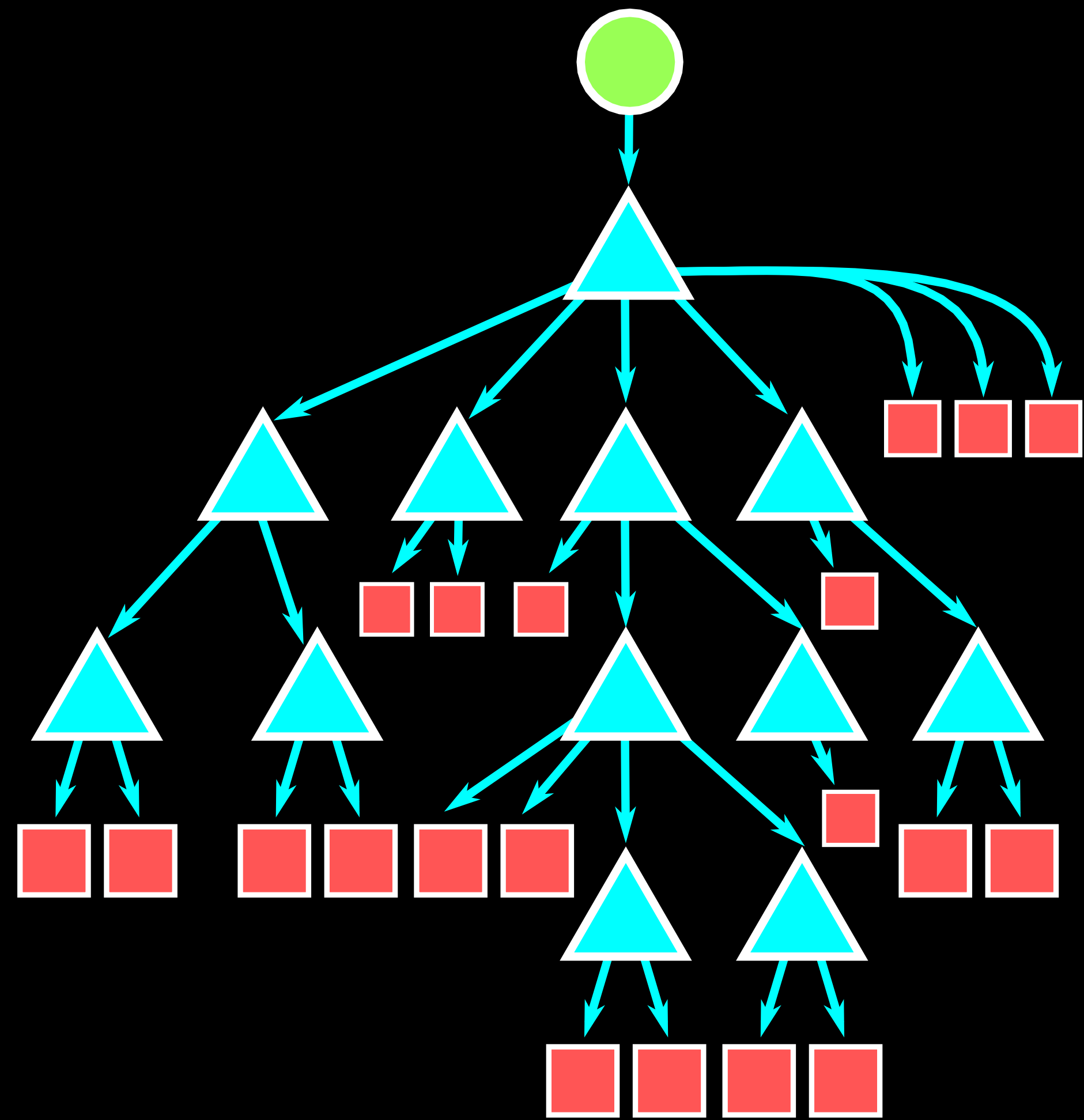


Flywheel architecture



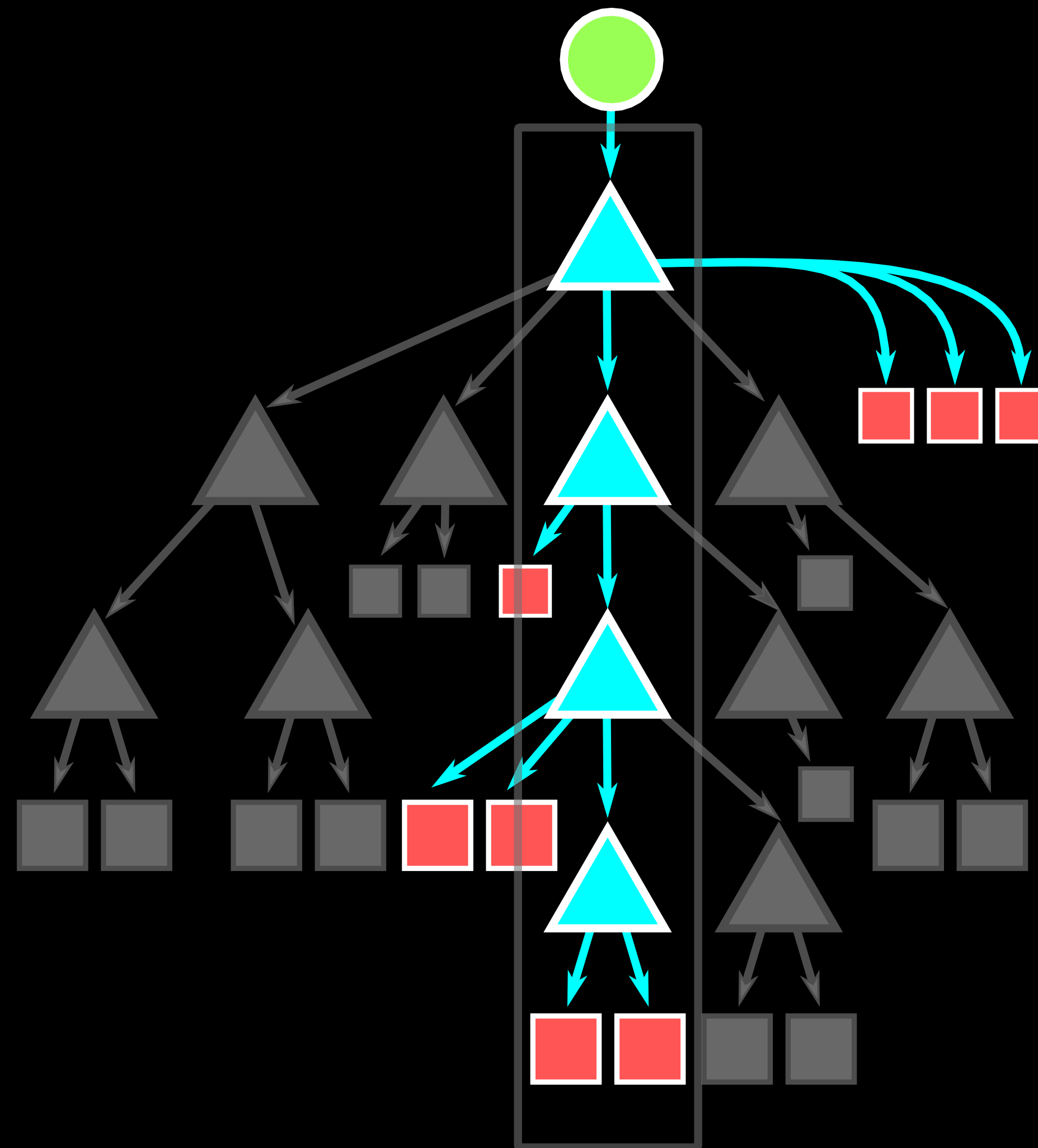
Contract-Implementation boundaries

Optimize the Working Directory



Full tree

Optimize the Working Directory



Sparse tree

Git sparse-checkout usage

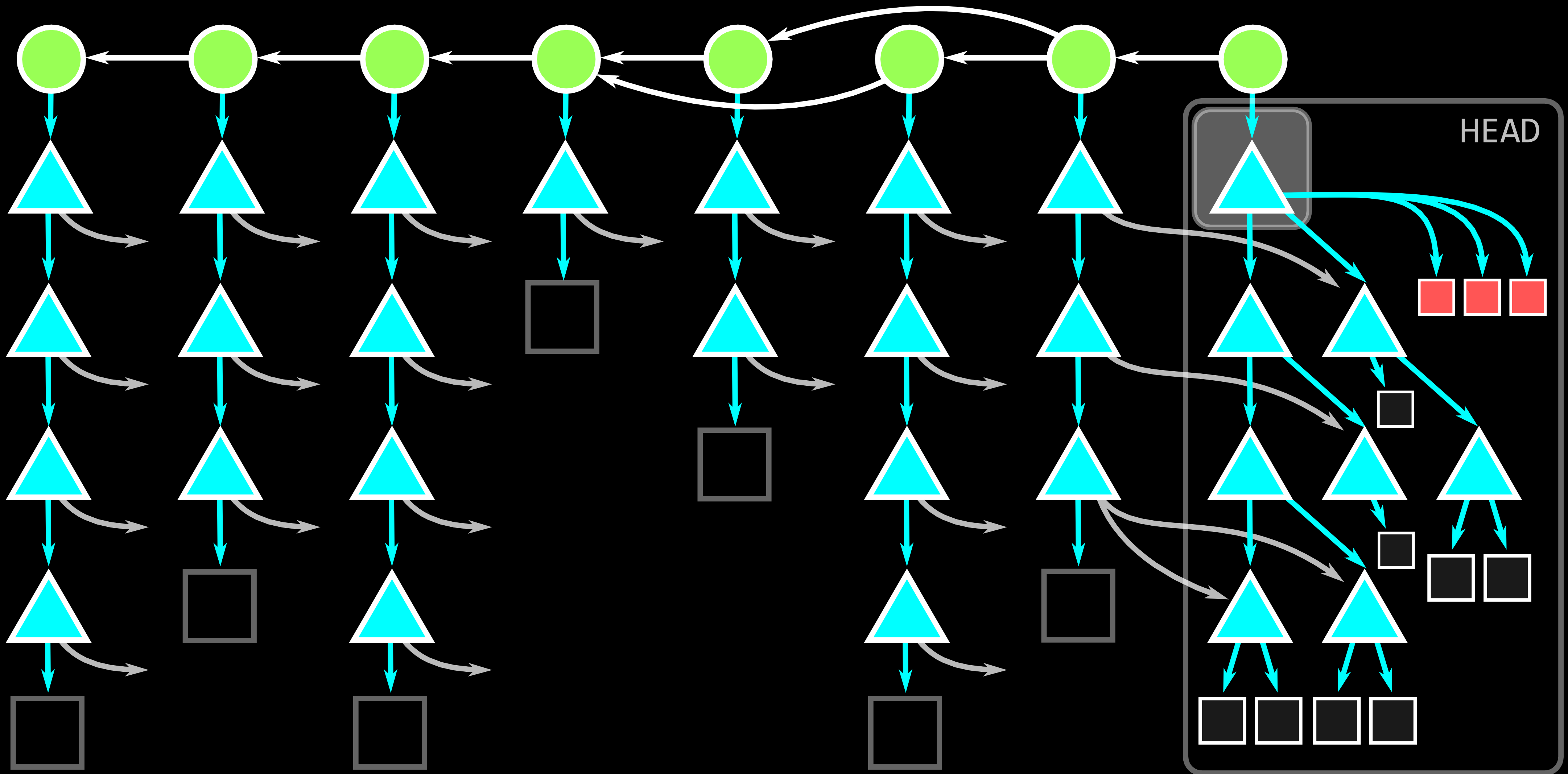
```
git sparse-checkout init --cone
```

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

```
git sparse-checkout add <dir>
```

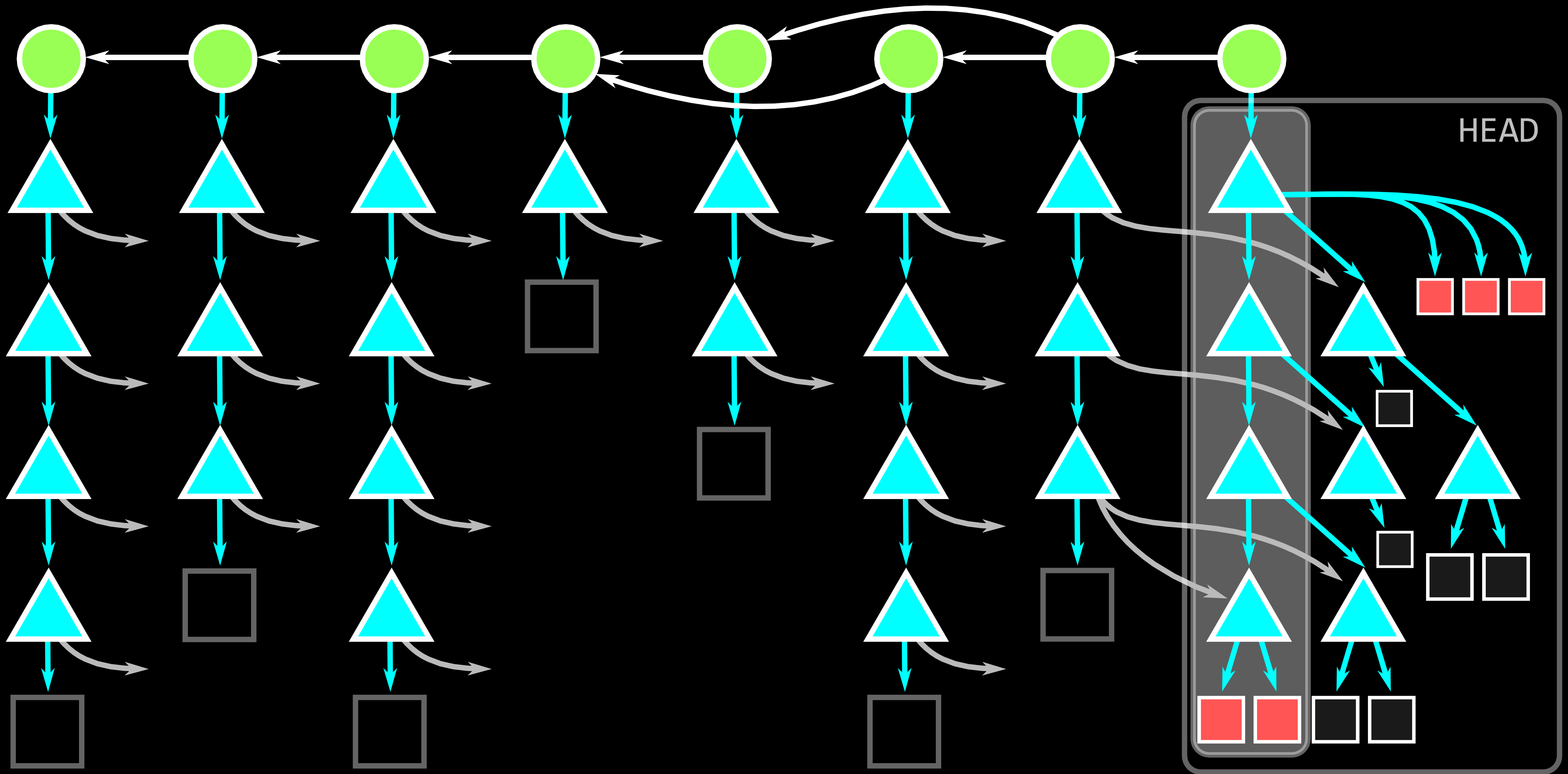
```
git sparse-checkout disable
```

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



Partial Clone with Sparse-Checkout

```
git clone --filter=blob:none --sparse <url> <repo>/src
```



Partial Clone with Sparse-Checkout

```
git sparse-checkout add dirA/dirB/dirC
```

Call to Action

YOU are an expert in **YOUR** build system!



Integrate builds with sparse-checkout!

If you use these features, then you are at
the forefront of Git at scale!

Know what's up

January 17, 2020 — Community, Open source

Bring your monorepo down to size with sparse-checkout



Derrick Stolee

This post
perform
push f

Git 2.2
existin
reposit

Does y
of com
proble
small f
feature
Specifi
develo

Previo
in your
The ins
sparse

July 2, 2020 — Client apps, Security

Git Credential Manager Core: Building a universal authentication experience



Matthew John Cheetham

Authentication is a critical component to your daily development. When working in open source, you need to prove that you have rights to update a branch with `git push`. Additionally when working on proprietary software, you need a way to prove that you even have read permission to access your code during `git fetch` or `git pull`.

Git currently supports two authentication mechanisms for accessing remotes. When using HTTP(S), Git sends a username and password, or a personal access token (PAT) via HTTP headers. When using SSH, Git relies on [the server knowing your machine's public SSH key](#). Though SSH-based authentication is considered most secure, setting it up correctly can often be a challenge. On the other hand, PATs are often much easier to set up, but also far less secure.

To manage all of this, Git relies on tools called [credential managers](#) which handle authentication to different hosting services. When first designed, these tools simply stored usernames and passwords in a secure location for later retrieval (e.g., your keychain, in an encrypted file, etc). These days, two-factor authentication (2FA) is commonly required to keep your data secure. This complicates the authentication story significantly since new and existing tools are required to meet the demands of these stricter authentication models.

April 14, 2020 — Community, Open source

Git credential helper vulnerability announced



Taylor Blau

Today, the
credential

These up
clone , eit
credential
exfiltrate y

Upgrac

The most
can't upda

- Avoid
- Avoid

GitHub ha
Specificall

April 7, 2020 — Community, Open source

Celebrating 15 years of Git: An interview with Git maintainer Junio Hamano



Jeff King

In celebration of Git's 15th anniversary, GitHub's Jeff King (@peff) interviewed Git's maintainer Junio Hamano about Git's 15 years and what's coming next. Jeff King is a distinguished Software Engineer at GitHub and has worked on scaling and maintaining Git on GitHub since 2011, but he's also been an active contributor to the Git project since 2006. Both Jeff and Junio serve on Git's project leadership committee at [Software Freedom Conservancy](#).

Junio, thanks so much for chatting with us as we celebrate Git's 15th anniversary. For the benefit of our readers, and because I got this wrong for the first several years of working with you, what is the correct pronunciation of your name?

January 13, 2020 — Community, Open source

Highlights from Git 2.25



Taylor Blau

March 22, 2020 — Open source

January 17 up

Highlights from Git 2.26

July 27, 2020 — Engineering, Open source

Highlights from Git 2.28



Taylor Blau

October 19, 2020 — Community, Open source

The open s
contributors
2.26 was rel
introduced s

Introduc

When you in
created an i
option, ini
more backg
is an excell

Starting in G
creating the
defaults to

1. This co
as easy
\$ git

Highlights from Git 2.29



Taylor Blau

The open source Git project [just released Git 2.29](#) with features and bug fixes from over 89 contributors, 24 of them new. Last time [we caught up](#) with you, Git 2.28 had just been released. One version later, let's take a look at the most interesting features and changes that have happened since then.

Experimental SHA-256 support

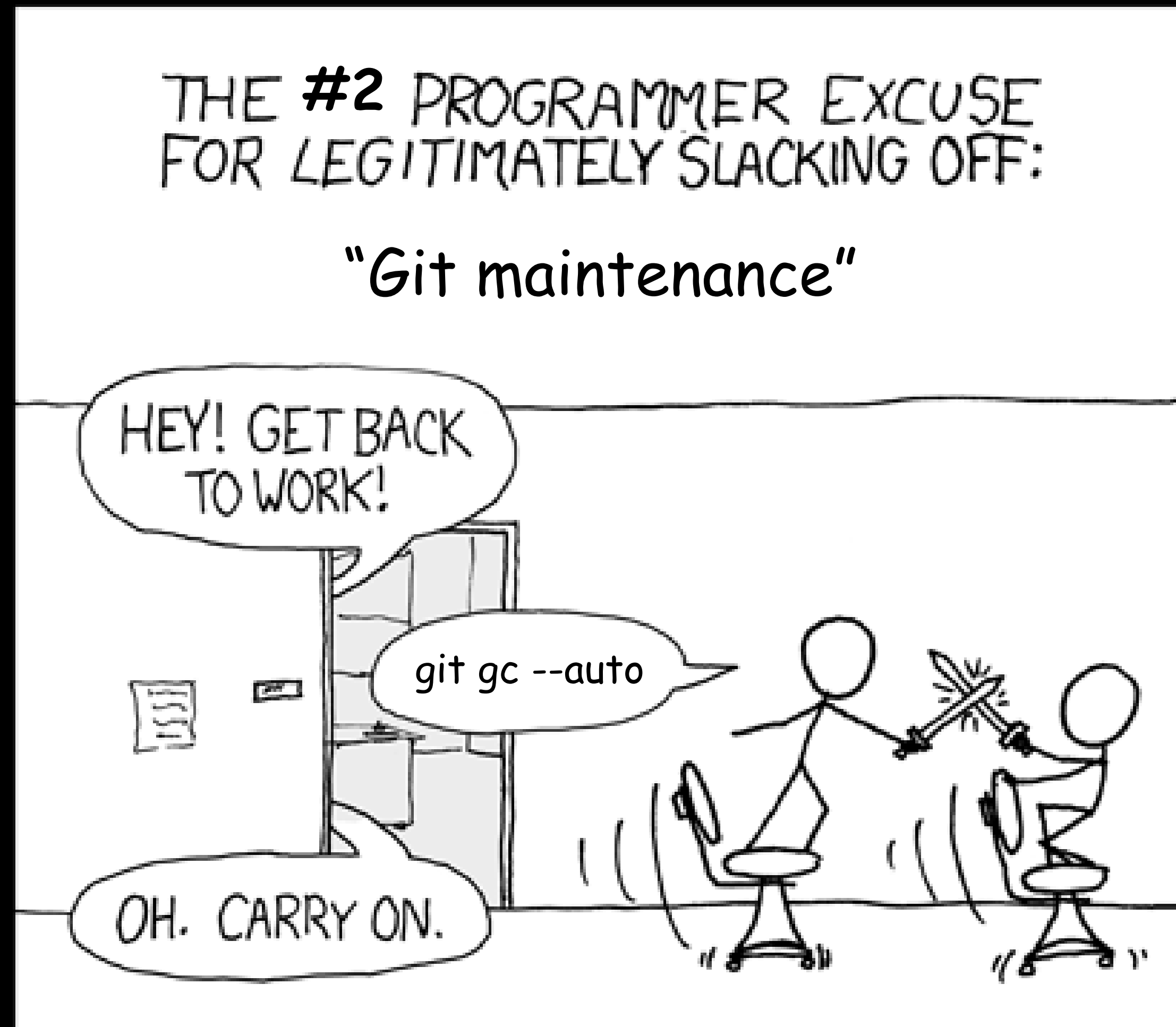
Git 2.29 includes experimental support for writing your repository's objects using a SHA-256 hash of their contents, instead of using SHA-1.

What does all of that mean? To explain, let's start from the beginning.

When you add files to a repository, Git copies their contents into `blob` objects in its local database, and creates `tree` objects that refer to the blobs. Likewise, when you run `git commit`, this creates a `commit` object that refers to the tree representing the committed state. How do these objects "refer" to each other, and how can you identify them when interacting with Git? The answer is that each object is given a unique name, called its *object id*, based on a hash of its contents. Git uses SHA-1 as its hash algorithm of choice, and depends on the object ids of different objects to be unique.

<https://github.blog/>

Coming Soon: Background Maintenance





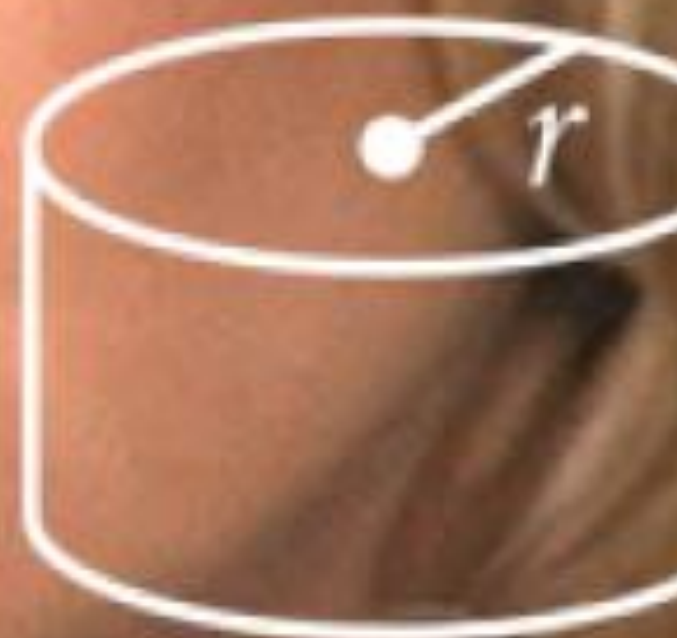
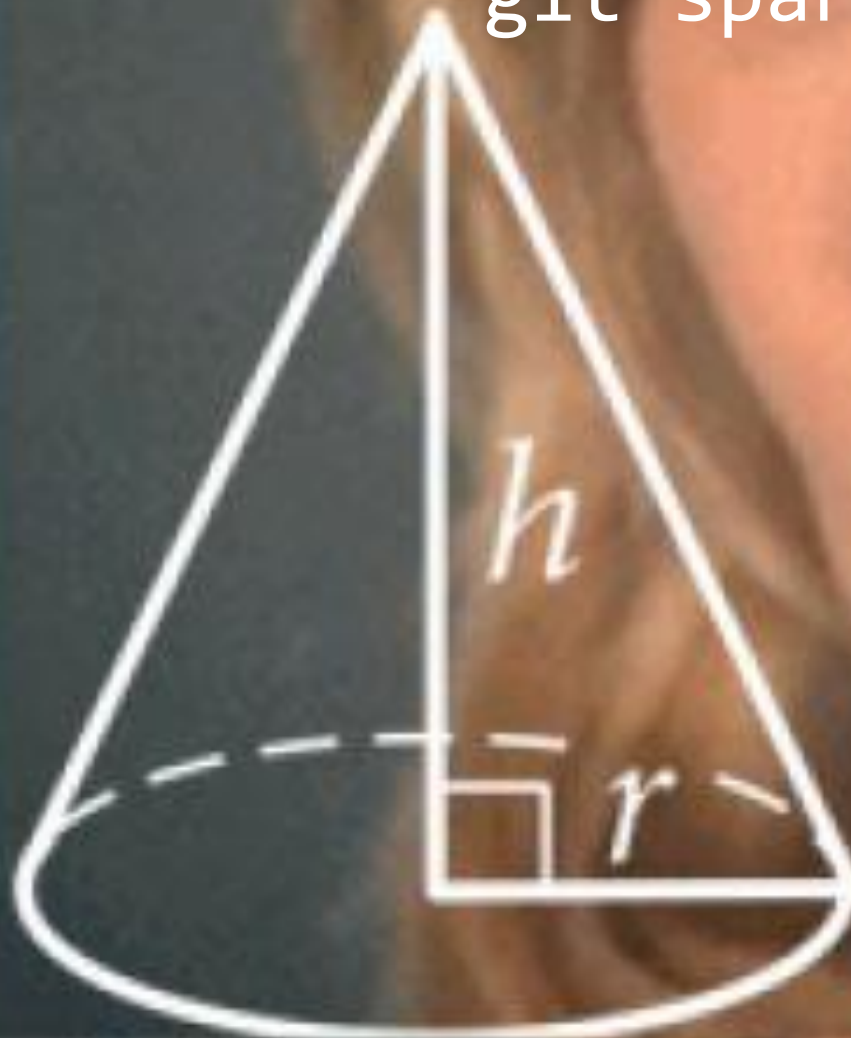
$$A = \pi r^2$$

$$C = 2\pi r$$

--filter=blob:none

$$V = \frac{1}{3} \pi r^2 h$$

git sparse-checkout init --cone



$$V = \pi r^2 h$$

| | 30° | 45° | 60° |
|-----|----------------------|----------------------|----------------------|
| sin | $\frac{1}{2}$ | $\frac{\sqrt{2}}{2}$ | $\frac{\sqrt{3}}{2}$ |
| cos | $\frac{\sqrt{3}}{2}$ | $\frac{\sqrt{2}}{2}$ | $\frac{1}{2}$ |
| tan | $\frac{\sqrt{3}}{3}$ | 1 | $\sqrt{3}$ |



$$\int \sin x dx = -\cos x + C$$

$$\int \frac{dx}{\cos^2 x} = \tan x + C$$

$$\int \tan x dx = -\ln|\cos x| + C$$

$$\int \frac{dx}{\sin x} = \ln\left|\tan \frac{x}{2}\right| + C$$

$$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} \arctg \frac{x}{a} + C$$

$$\int \frac{dx}{x^2 - a^2} = \frac{1}{2a} \ln\left|\frac{x-a}{x+a}\right| + C$$



.gitignore
src
LFS

$$ax^2 + bx + c = 0$$

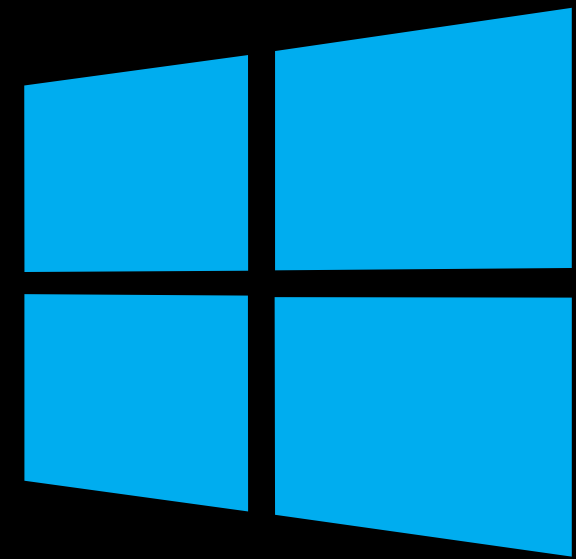
$$a\left(x^2 + \frac{b}{a}x + \frac{c}{a}\right) = 0$$

$$x^2 + 2\frac{b}{2a}x + \left(\frac{b}{2a}\right)^2 - \left(\frac{b}{2a}\right)^2 + \frac{c}{a} = 0$$

$$\left(x + \frac{b}{2a}\right)^2 - \frac{b^2 - 4ac}{4a^2} = 0$$



Scalar



<https://github.com/microsoft/scalar>

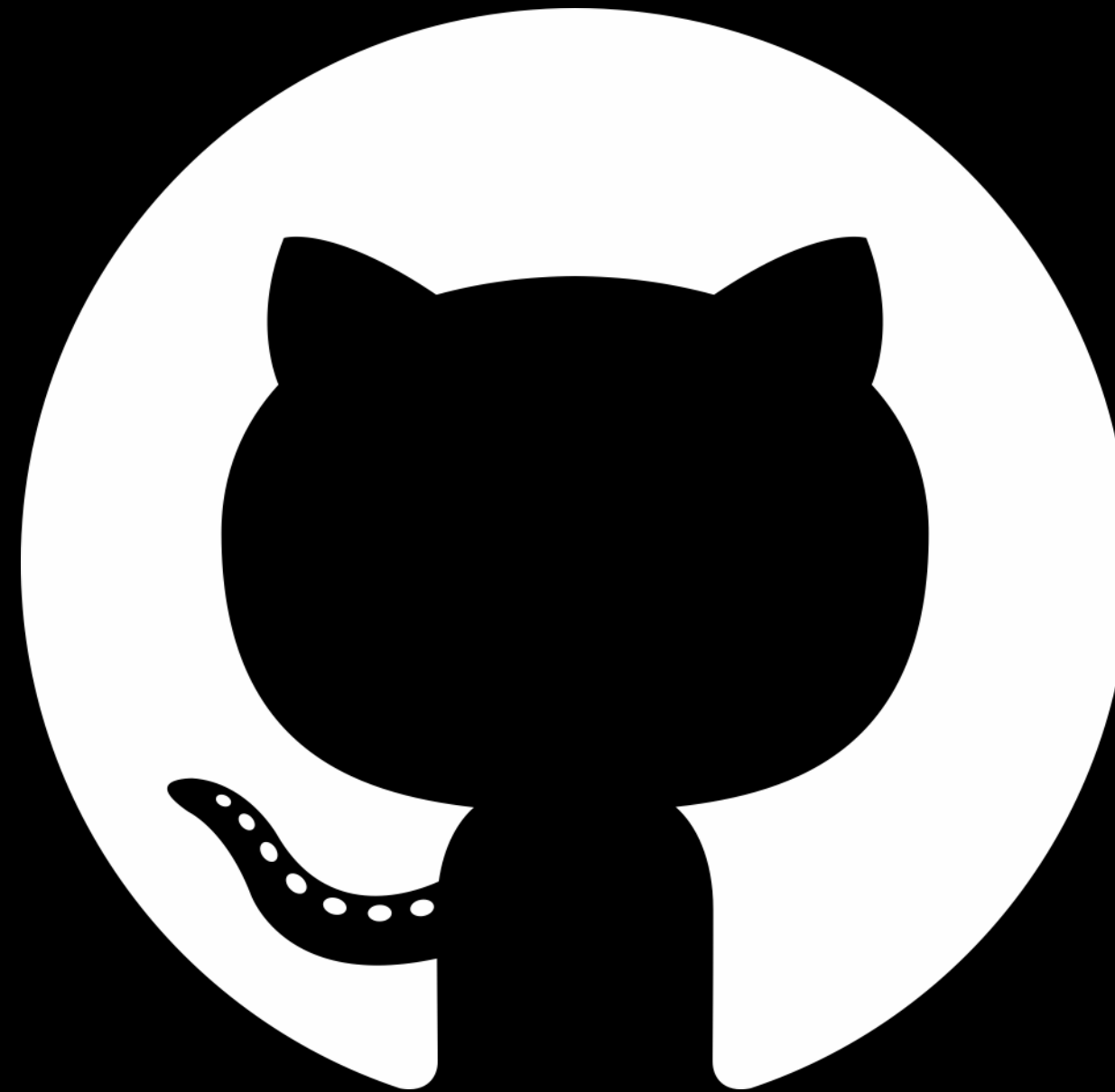
scalar clone <url> will...

- Create your repo in <repo>/src
- Default to using partial clone (--filter=blob:none)
- Default to using sparse-checkout
- Set up background maintenance
- Initialize recommended, advanced config settings

<https://github.com/microsoft/scalar>



Derrick Stolee



GitHub: @derrickstolee
Twitter: @stolee
Website: <https://stolee.dev>