GIT GUD

Agenda

1. Background
2. Internals
3. Tips & Tricks
4. Best Practices

Background

what is `Git`?

VCS

Version Control System

[…] a system that records changes to a file or set of files over time

– Scott Chacon & Ben Straub [ProGit]

[ProGit] https://git-scm.com/book/en/v2

Version Control Systems

- CVS
- Perforce
- SVN
- Mercurial
- Git

Open Source

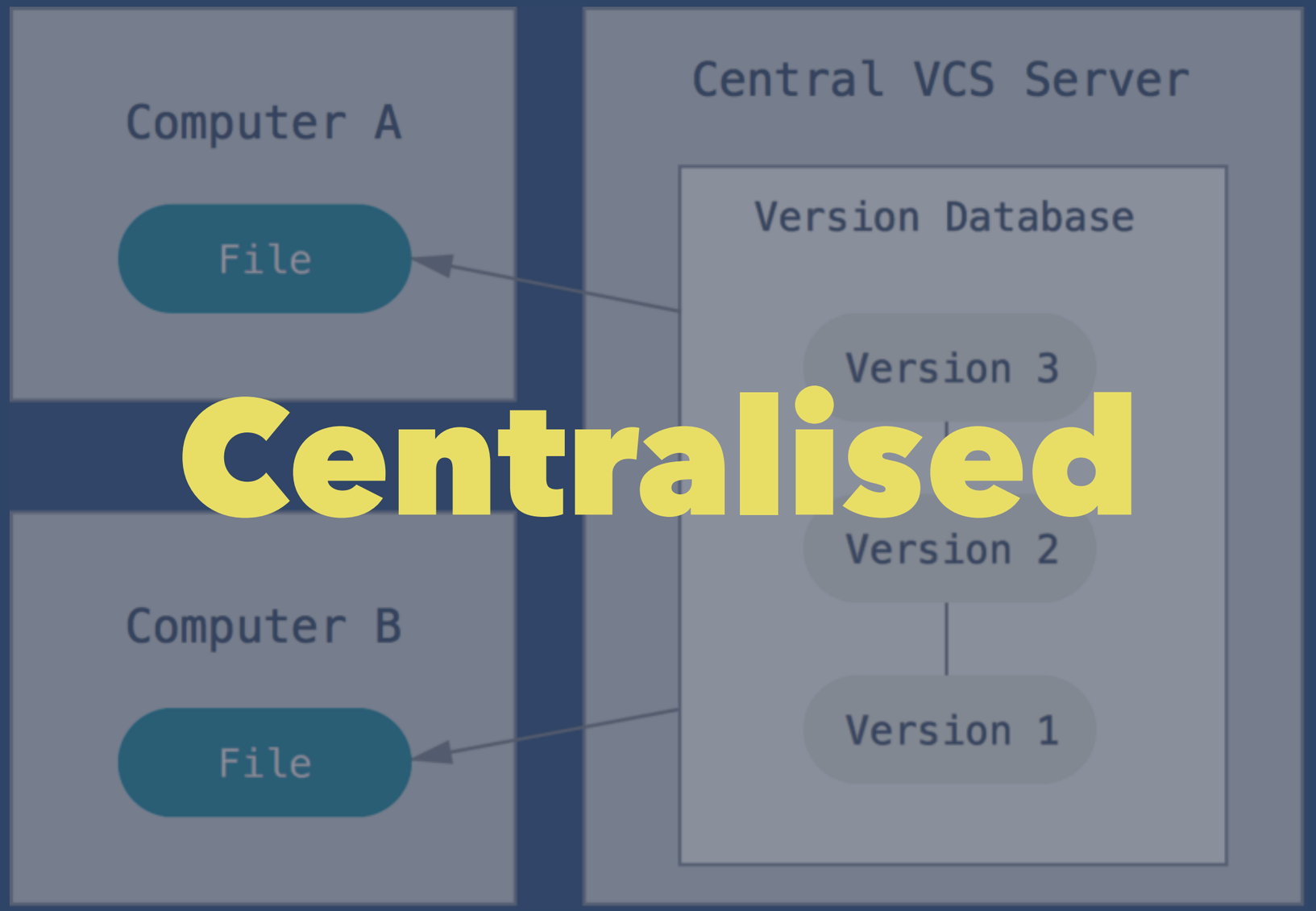
GNU GPL2

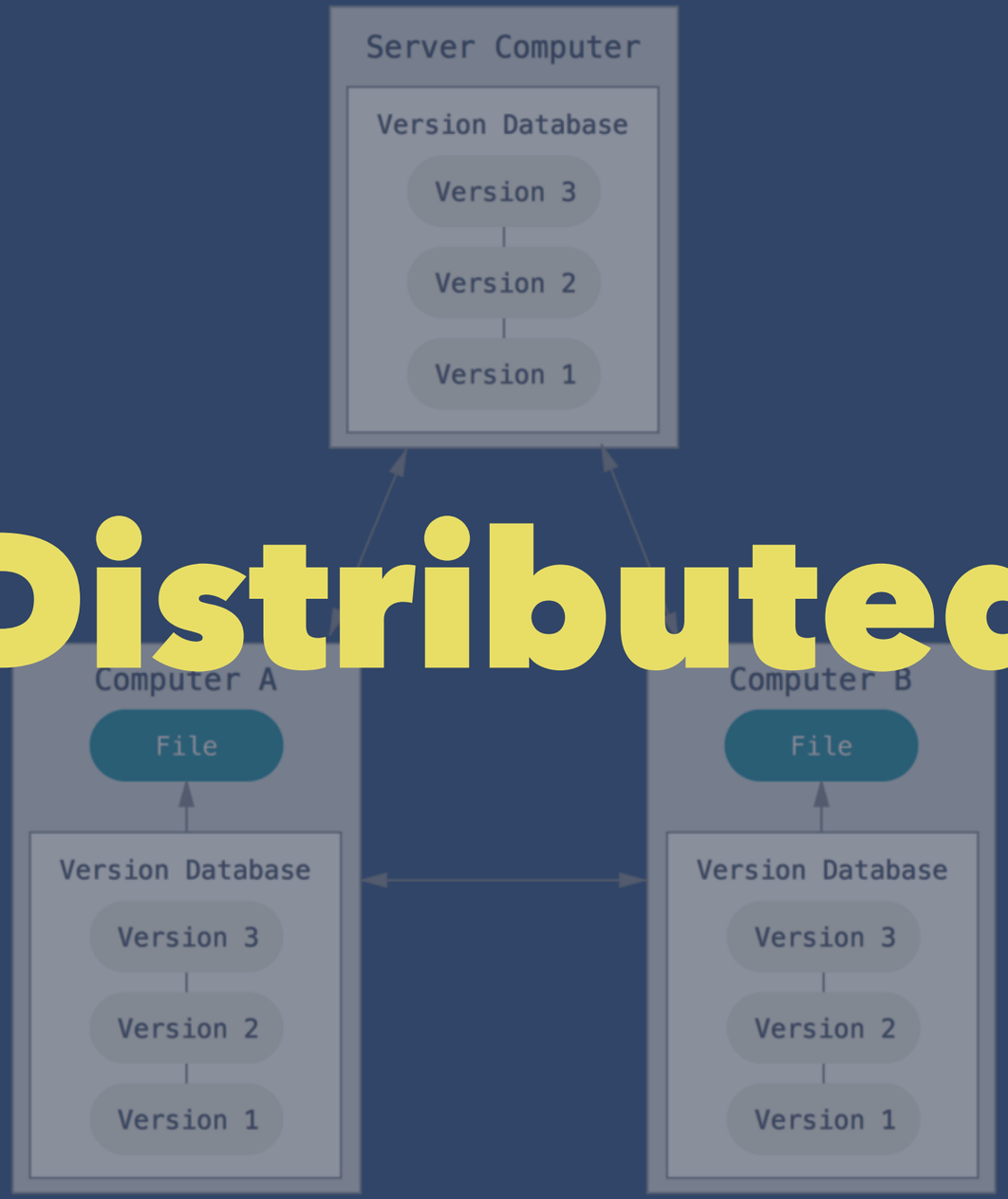
Distributed

Centralised

vs

Distributed


Computer A
File
Computer B
File
Central VCS Server
Version Database
Version 3
Version 2
Version 1
Centralised



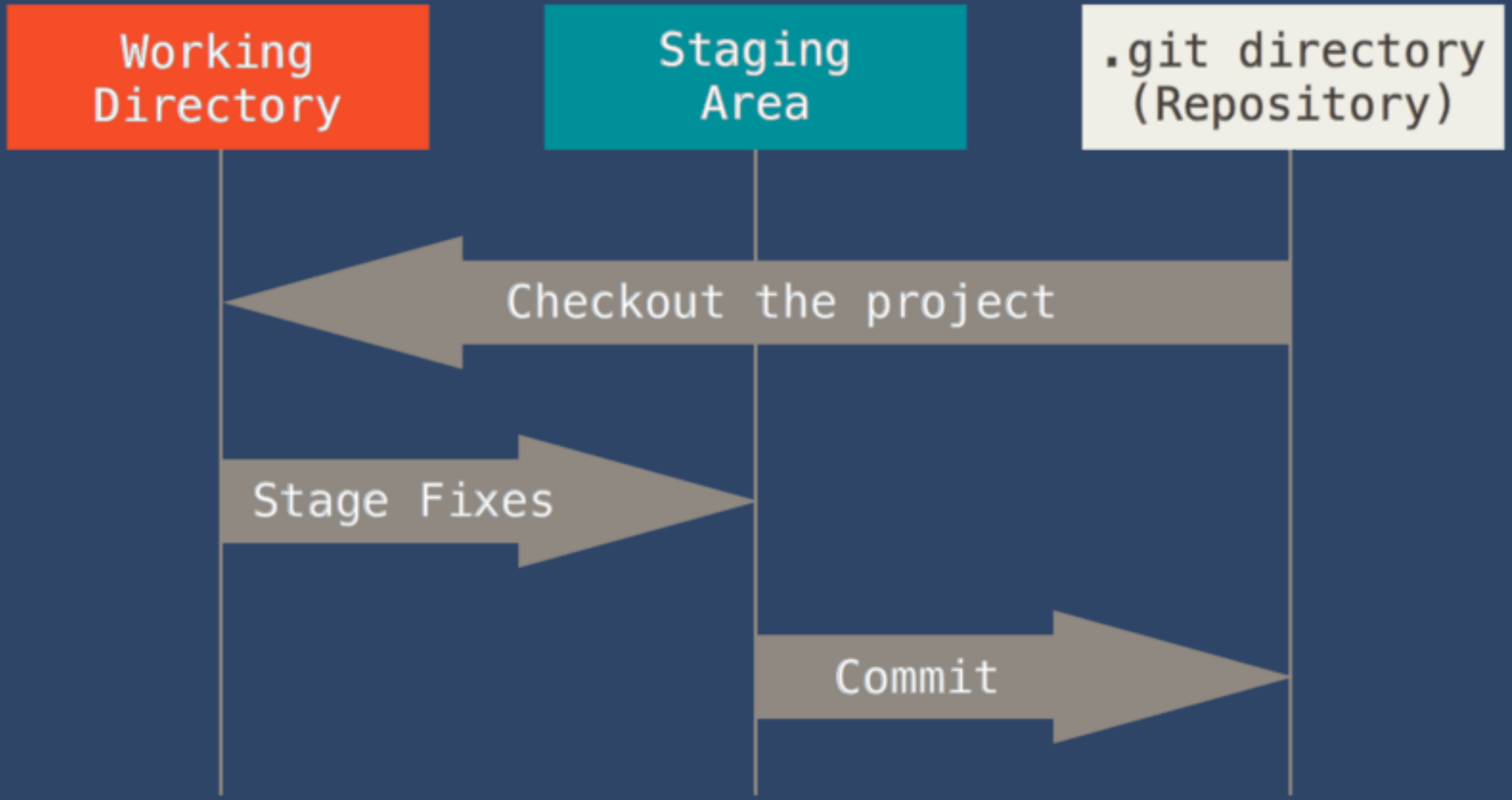
Distributed

Internals

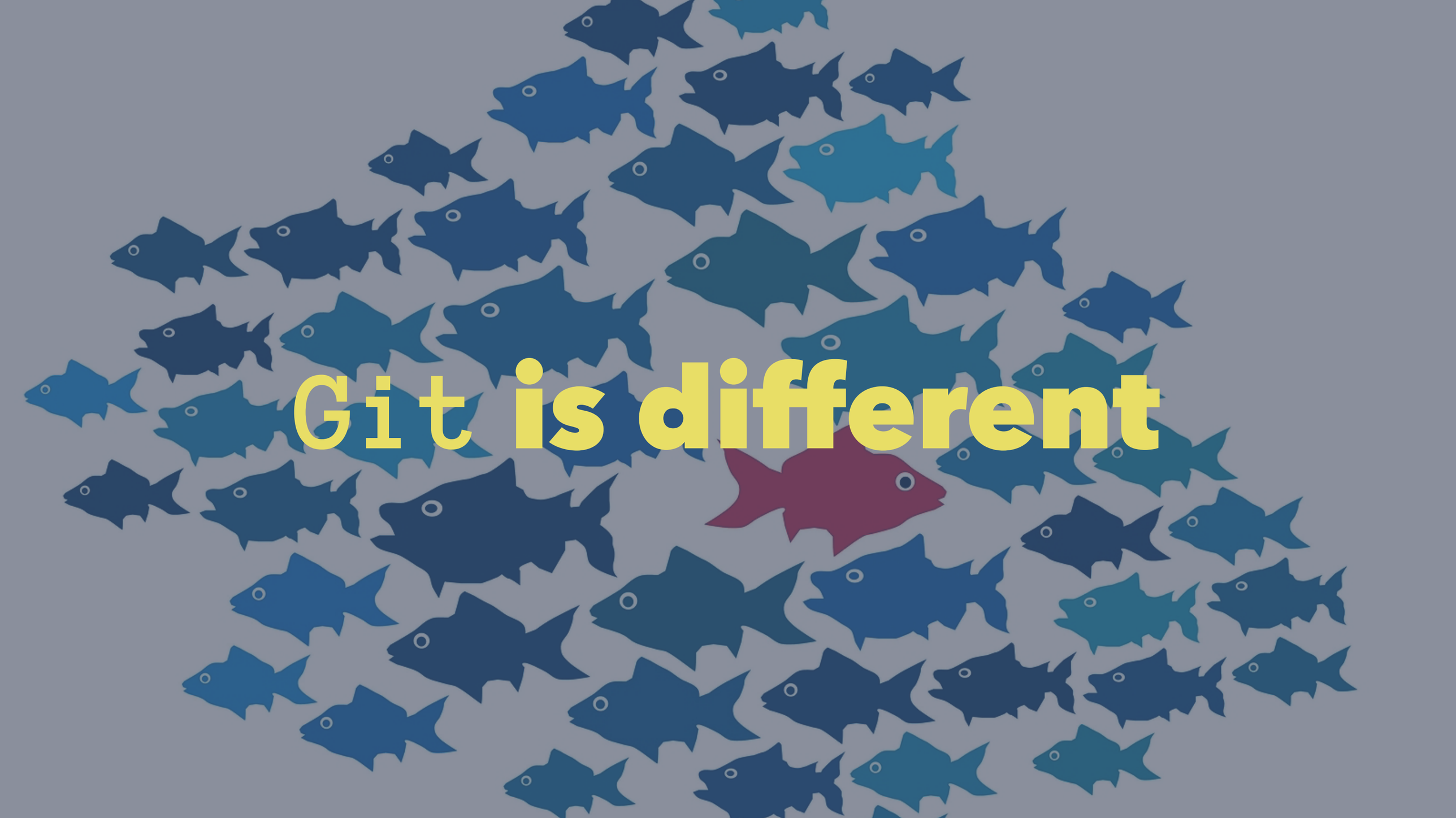
let's get our hands dirty

Expected Knowledge

- `add`
- `reset`
- `commit`
- `merge`


Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

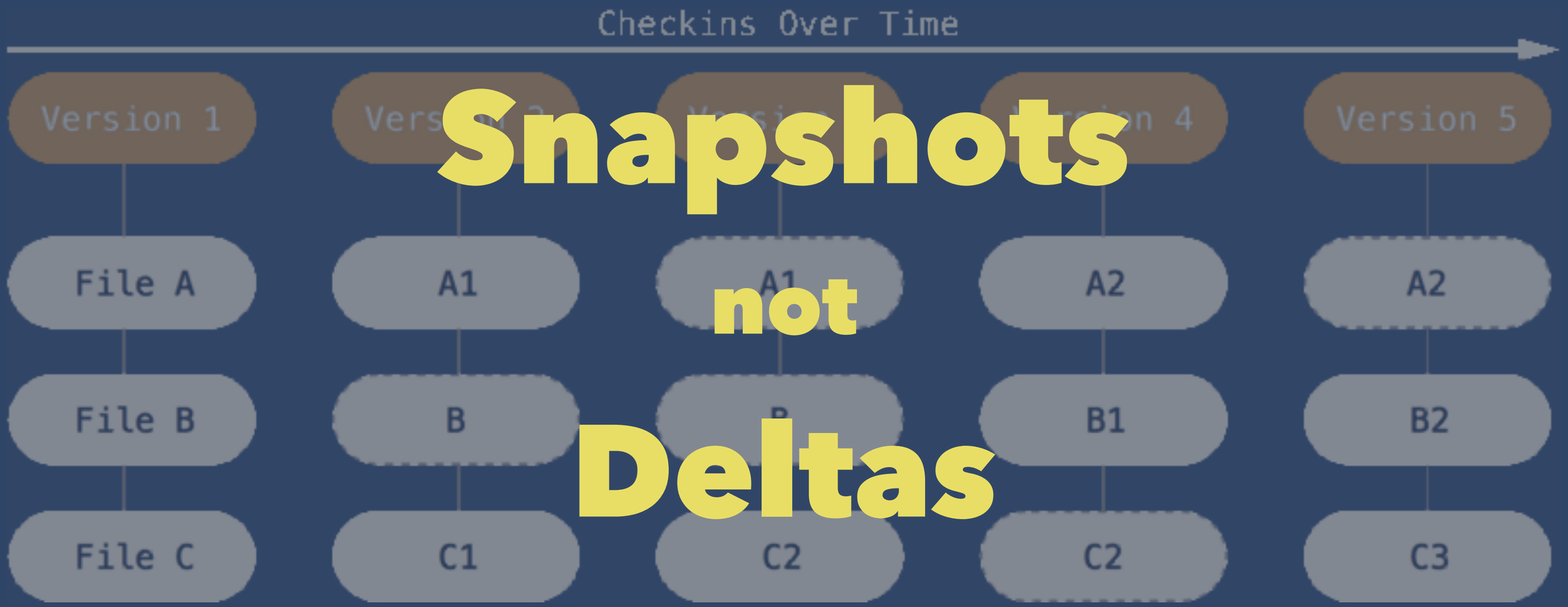
How does `Git` track changes?


Git is different

The major difference [...] is the way Git thinks about its data.

– Scott Chacon & Ben Straub [ProGit]

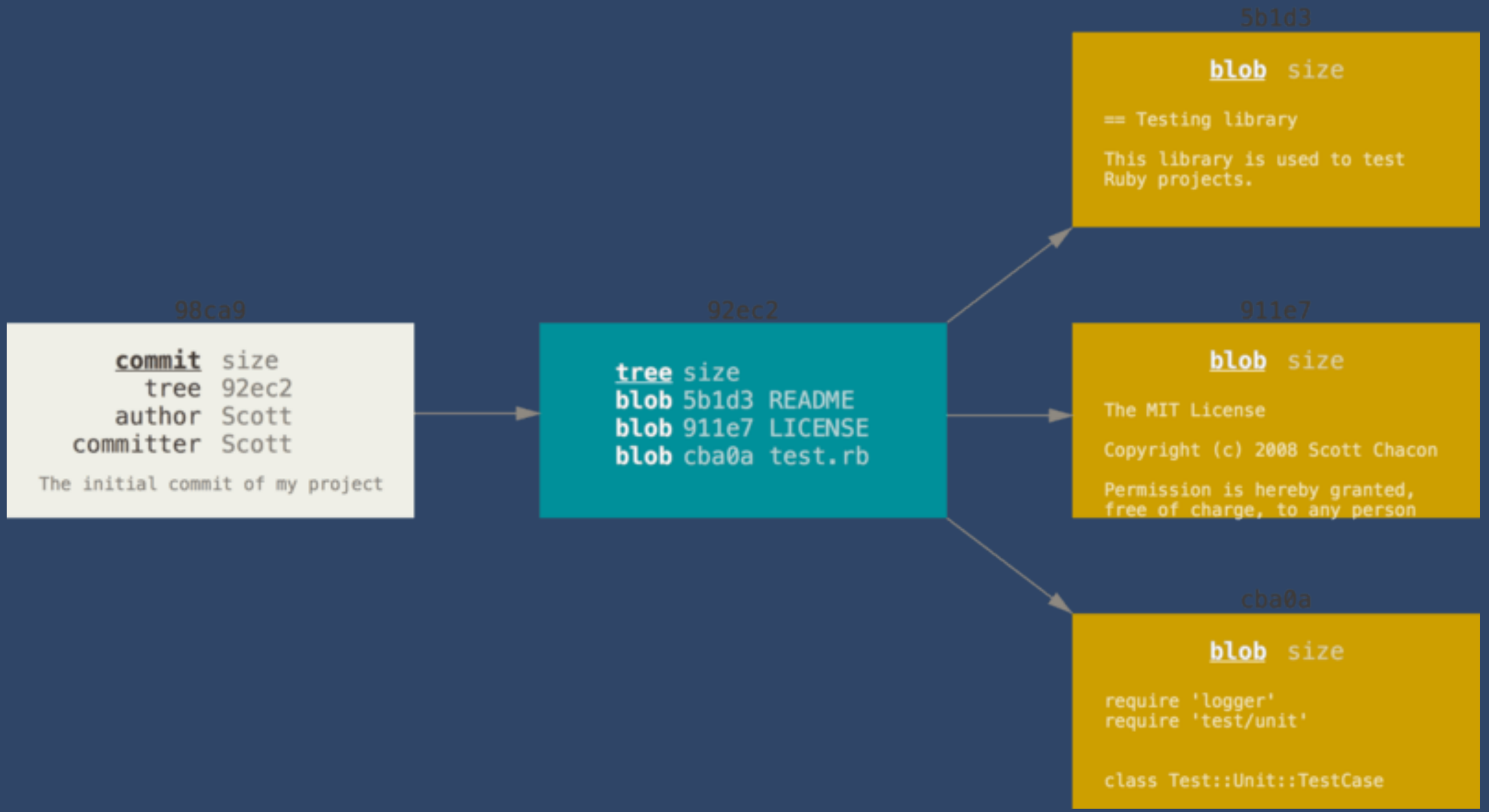
[ProGit] https://git-scm.com/book/en/v2



Snapshots not Deltas

What is a

Commit


98ca9
commit size
tree 92ec2
author Scott
committer Scott
The initial commit of my project
92ec2
tree size
blob 5b1d3 README
blob 911e7 LICENSE
blob cba0a test.rb
5b1d3
blob size
== Testing library
This library is used to test
Ruby projects.
911e7
blob size
The MIT License
Copyright (c) 2008 Scott Chacon
Permission is hereby granted,
free of charge, to any person
cba0a
blob size
require 'logger'
require 'test/unit'
class Test::Unit::TestCase

SHA1

Try it out!

Clone workshop repository

```
git clone
git@github.gcxi.de:swolf/git-workshop.git
```

```
$ cd git-workshop

$ echo 'Some random text' > my_file

$ git hash-object -w my_file
1a76b8a41993e2c667f5b191fb57abdab2102a8b

$ git cat-file -t 1a76b8a41993e2c667f5b191fb57abdab2102a8b
blob

$ git cat-file -p 1a76b8a41993e2c667f5b191fb57abdab2102a8b
Some random text
```

```
➜ git log --all --decorate --graph --oneline
* f097585 (HEAD → waldo-came-and-left, origin/waldo-came-and-left) Add some more distracting stuff
* 0356fc5 Add some distracting stuff
* 8a28c0f Waldo left the building
* eeaf062 Add company for wald
* c85f3cf
* d175476
| *   f20c965 (origin/master, master) Merge branch 'd
| |\
| |/
|/|
* | 12e86b3 (origin/dev, dev) Add bar
* | 4043e69 Add some content to foo
|/
* c9866a9 Add foo
```

How history is made

`git` history at least


98ca9
commit size
tree 92ec2
parent
author Scott
committer Scott
The initial commit of my project
34ac2
commit size
tree 184ca
parent 98ca9
author Scott
committer Scott
Fixed bug #1328 - stack overflow under certain conditions
f30ab
commit size
tree 0de24
parent 34ac2
author Scott
committer Scott
add feature #32 - ability to add new formats to the central interface
Snapshot A
Snapshot B
Snapshot C

Honour thy parents

Navigate through history

^

and

~<N>


wat

```
$ git checkout ancestry
...

$ git log --oneline -1 HEAD
822eca1 Add the number 5 to numbers

$ git log --oneline -1 HEAD^
d4bd446 Add the number 4 to numbers

$ git log --oneline -1 HEAD^^
1b94d52 Add the number 3 to numbers

$ git log --oneline -1 HEAD~2
1b94d52 Add the number 3 to numbers
```


What is a
Branch

Let's take a look

```
$ ls -l .git/refs/heads
total 32
-rw-r--r--  1 swolf  staff  41 Jan 16 09:06 ancestry
-rw-r--r--  1 swolf  staff  41 Jan 16 07:00 dev
-rw-r--r--  1 swolf  staff  41 Jan 16 07:00 master
-rw-r--r--  1 swolf  staff  41 Jan 16 07:02 waldo-came-and-left
```

Huh, `master` is a file ...

I wonder what's in there ...

```
$ cat .git/refs/heads/master
f20c96538c6dca6fb37e631388814fe941afc2ae
```

It's just a hash!

```
$ cat .git/refs/heads/master | git cat-file --batch
f20c96538c6dca6fb37e631388814fe941afc2ae commit 540
tree 9ab6d12a4587fb4a02de1b5e745277c6c832cf5a
parent c9866a994539e01938ceec6c75ada013ce456e9b
parent 12e86b3370f931c423022a8c8d7d7fa7413e0bc4
author Sascha Wolf <sascha.wolf@grandcentrix.net> 1516082444 +0100
committer Sascha Wolf <sascha.wolf@grandcentrix.net> 1516082444 +0100
gpgsig -----BEGIN PGP SIGNATURE-----

 iHUEABEIAB0WIQTQyMji07ff76Vkk26j80vFo8w6AAUCWl2VDAAKCRCj80vFo8w6
 AOqtAPoCPrGCN1QZKKGmlqZu43n824v2wviWbUujd9CBwuUURQD/dn2EJ2zHi+zQ
 qBKm1cfsi5vwTiYj31O4UbOZ1rJbW1M=
 =vi7g
 -----END PGP SIGNATURE-----

Merge branch 'dev'
```

What about HEAD?

```
$ cat .git/HEAD
ref: refs/heads/master

$ git checkout --detach master
...

$ cat .git/HEAD
f20c96538c6dca6fb37e631388814fe941afc2ae
```


What is a
Merge

What does the log tell us?

```
$ git log --decorate --graph --oneline
*   f20c965 (origin/master, master) Merge branch 'dev'
|\
| * 12e86b3 (origin/dev, dev) Add bar
| * 4043e69 Add some content to foo
|/
* c9866a9 Add foo
```

Remember the commit object?

```
$ cat .git/refs/heads/master | git cat-file --batch
f20c96538c6dca6fb37e631388814fe941afc2ae commit 540
tree 9ab6d12a4587fb4a02de1b5e745277c6c832cf5a
parent c9866a994539e01938ceec6c75ada013ce456e9b
parent 12e86b3370f931c423022a8c8d7d7fa7413e0bc4
author Sascha Wolf <sascha.wolf@grandcentrix.net> 1516082444 +0100
committer Sascha Wolf <sascha.wolf@grandcentrix.net> 1516082444 +0100
gpgsig -----BEGIN PGP SIGNATURE-----

 iHUEABEIAB0WIQTQyMji07ff76Vkk26j80vFo8w6AAUCWl2VDAAKCRCj80vFo8w6
 AOqtAPoCPrGCN1QZKKGmlqZu43n824v2wviWbUujd9CBwuUURQD/dn2EJ2zHi+zQ
 qBKm1cfsi5vwTiYj31O4UbOZ1rJbW1M=
 =vi7g
 -----END PGP SIGNATURE-----

Merge branch 'dev'
```

Tips & Tricks

We're going to look at

- `bisect`
- `--patch` mode
- `stash`
- `reflog`
- `rebase`
- `filter-branch`

One thing before we start …

Aliase

`git config --global alias.<your-alias> <command>`

In Action

```
$ git config --global alias.l 'log --decorate --graph --oneline'

$ git l -3
*   f20c965 (HEAD -> master, origin/master) Merge branch 'dev'
|\
| * 12e86b3 (origin/dev, dev) Add bar
| * 4043e69 Add some content to foo
|/
```

git bisect

a smart bug hunters weapon

git bisect

Search for the first commit which introduces an issue

```
$ git bisect start
$ git bisect bad <known bad commit>
$ git bisect good <known good commit>
```

Shortcut

```
$ git bisect <known bad commit> <known good commit>
```

Automate it!

```
git bisect run <command>
```

Let's try it out!

```
git bisect start waldo-came-and-left d175
```

Stop bisecting

```
git bisect reset
```


--patch
add | checkout | reset

Let's try it out!

```
$ git checkout patch-practice
...

$ git apply patch-practice.diff
...

$ git add --patch # Try to add only the above line!
```

Level up your `--patch`

edit mode


git stash
quicksave for git

`git stash`

- `push` create a stash from current changes
- `pop` apply and delete a stash
- `drop` delete a stash

Use `stash@{<n>}` to reference older stashes

`git reflog`

a history for your `HEAD`

Try it out!

```
git reflog
```

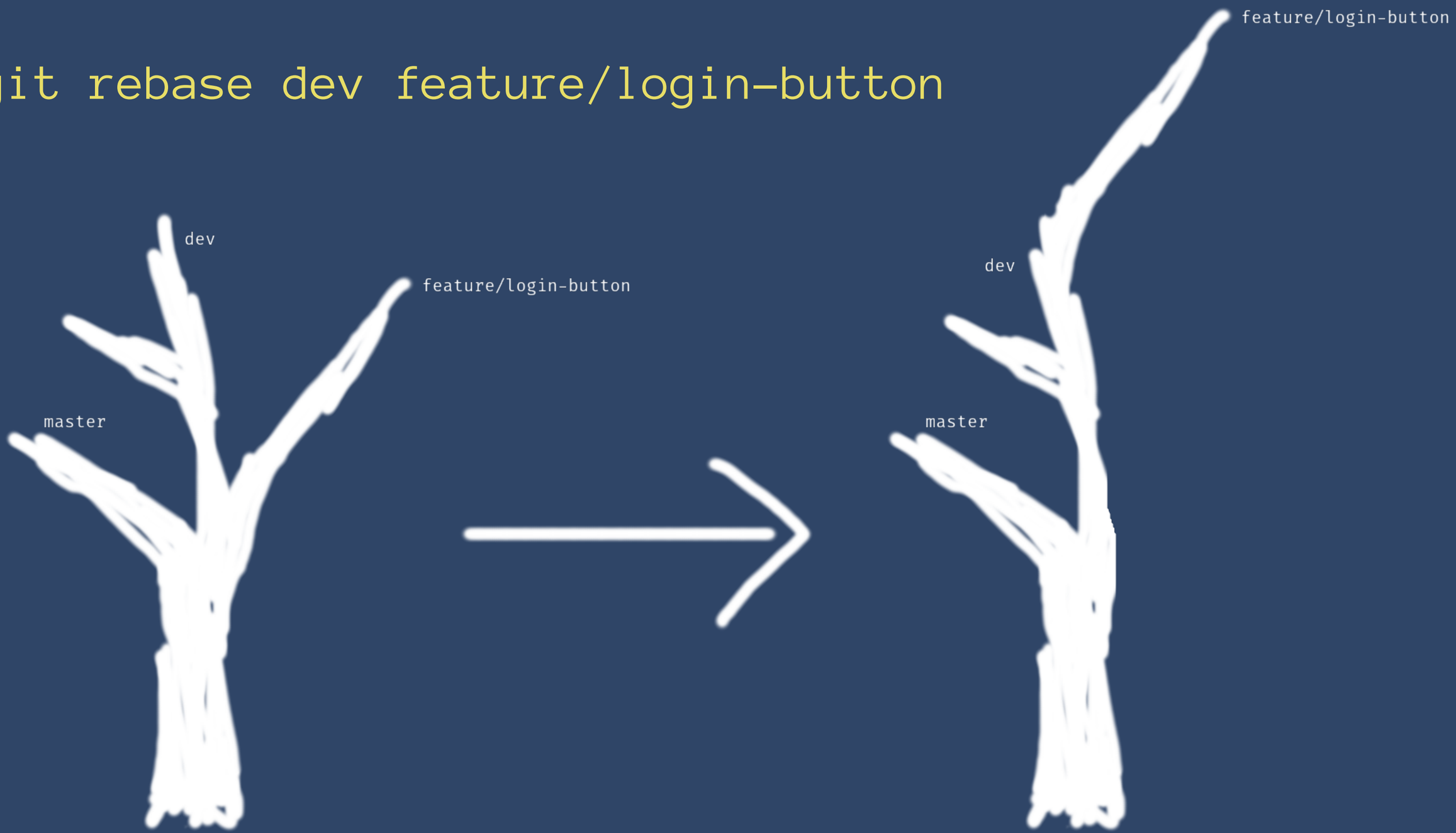
Branches have `reflog`s too!

```
git reflog <some branch>
```

git rebase

one command to rule them all

git rebase dev feature/login-button



Let's try it out

```
$ git checkout rebase-practice
...

$ git rebase dev
???
```

Let's fix this

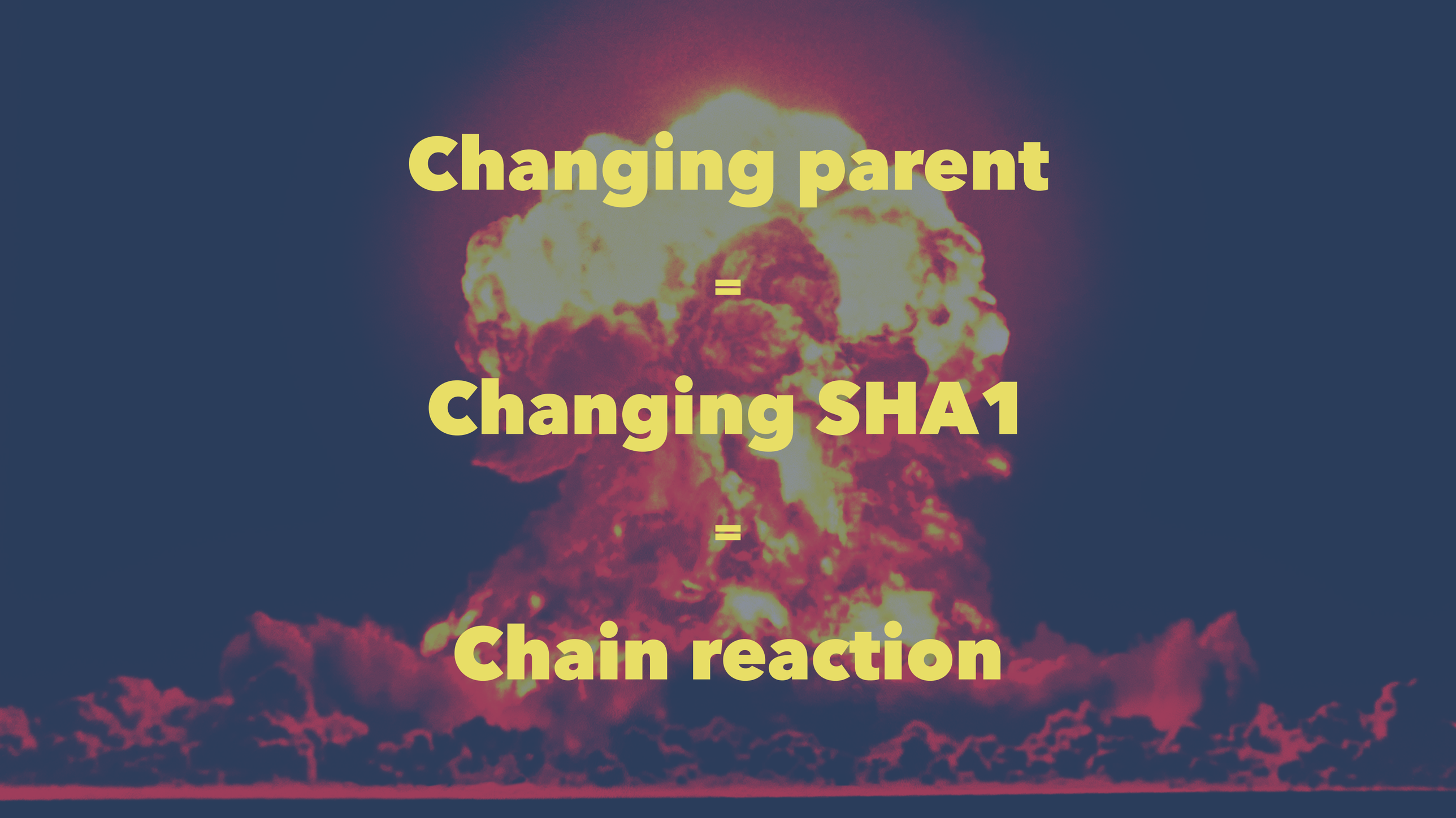
We don't want to remove foo!

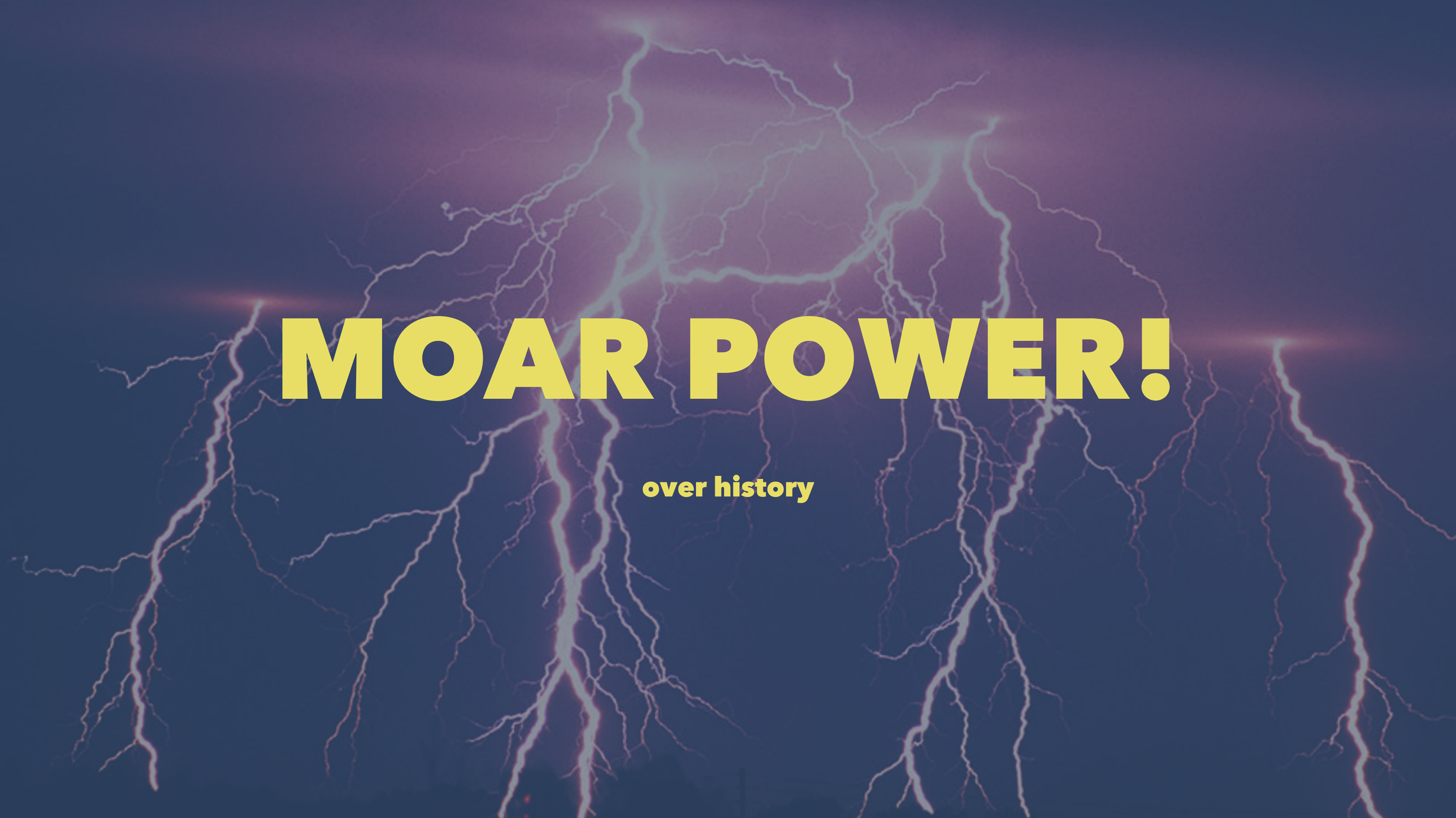
```
$ git rebase --abort
...

$ git rebase --interactive dev
???
```

Avoid rebasing pushed commits!

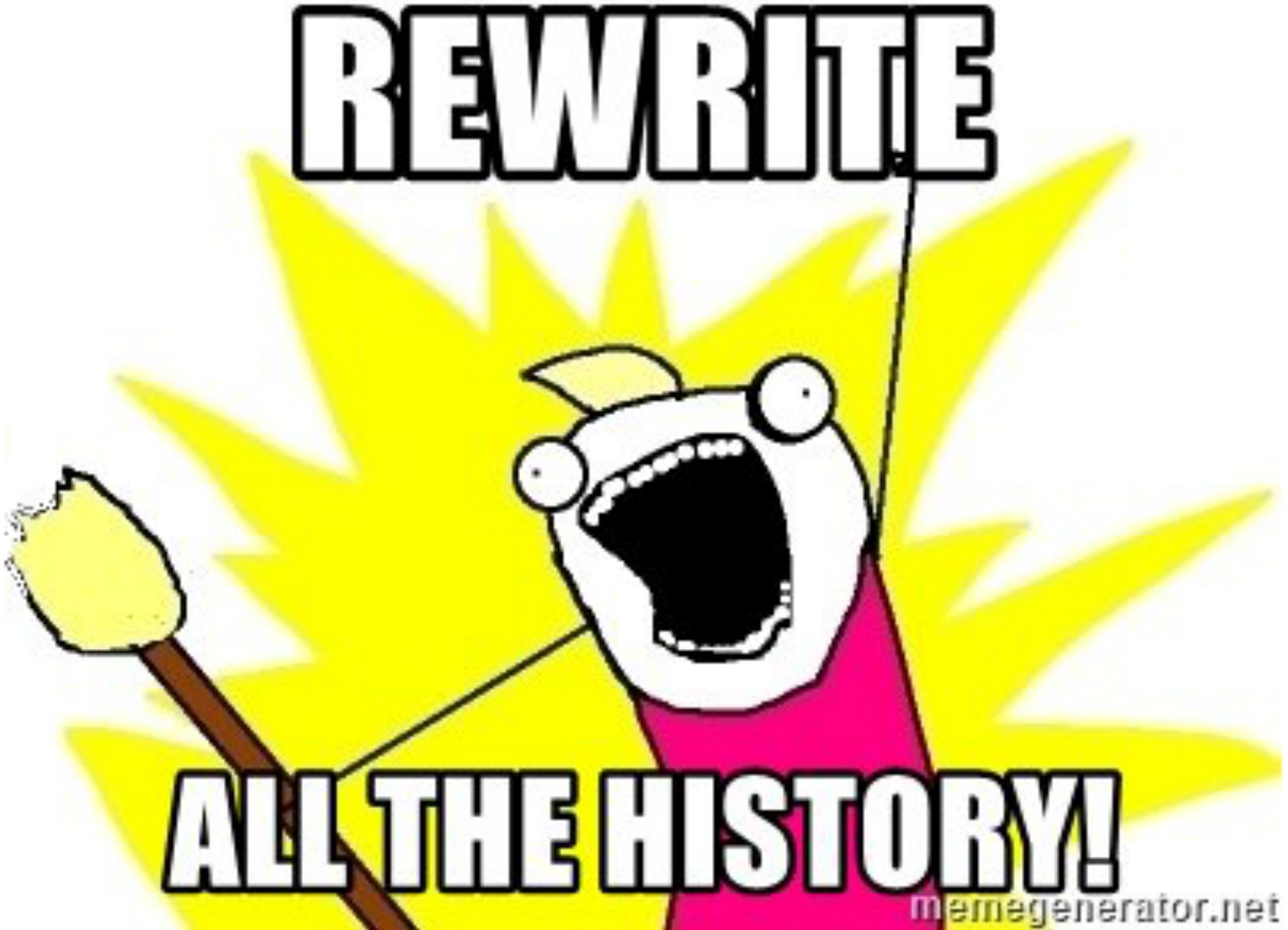
`rebase` rewrites history

with great power comes great responsibility


Changing parent
=
Changing SHA1
=
Chain reaction


MOAR POWER!
over history

git filter-branch


REWRITE
ALL THE HISTORY!
memegenerator.net

git filter-branch

Allows to rewrite

```
--env-filter <command>
--index-filter <command>
--msg-filter <command>
--parent-filter <command>
--subdirectory-filter <directory>
--tag-name-filter <command>
--tree-filter <command>
```

Honourable mentions

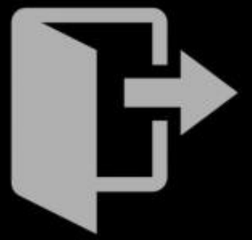
- `cherry-pick`: "copy" one+ commits onto the current branch
- `clean`: remove untracked files and folders
- `commit --amend`: include added changes in last commit
- `grep`: search pattern in tracked files
- `worktree`: checkout a reference into another worktree

[2] http://sethrobertson.github.io/GitBestPractices

IN CASE OF FIRE

Git Commit

Git Push



Git Out

Commit often
Perfect later
Publish once

Use shortlived branches

or at least merge upstream work regularly

Do not rewrite published history!


ORBO


Commit Messages

Commit Messages

```
<Topic|File>: <Short description> (50 characters sort limit)

<More detailed description> (bullet points are fine)
- problem (what is the issue with the current implementation)
- choosen approach with reasoning
- maybe rejected approaches
```

Use imperativ

`Add foo` instead of `Added foo`

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.


Thank you