BC/NAV TECH DAYS 2022

mibuso.com

10 YEAR ANNIVERSARY

www.bctechdays.com

Git Under the Hood

Vjekoslav Babić
Vjeko.com

Why Git?

Why Git?

I mean, it's 2022, not 2005...

BC/NAV TECH DAYS 2022
mibuso.com

BC/NAV TECH DAYS 2022
mibuso.com

What is Git?

BC/NAV TECH DAYS 2022
mibuso.com

stupid content tracker

NAME

git - the stupid content tracker

SYNOPSIS

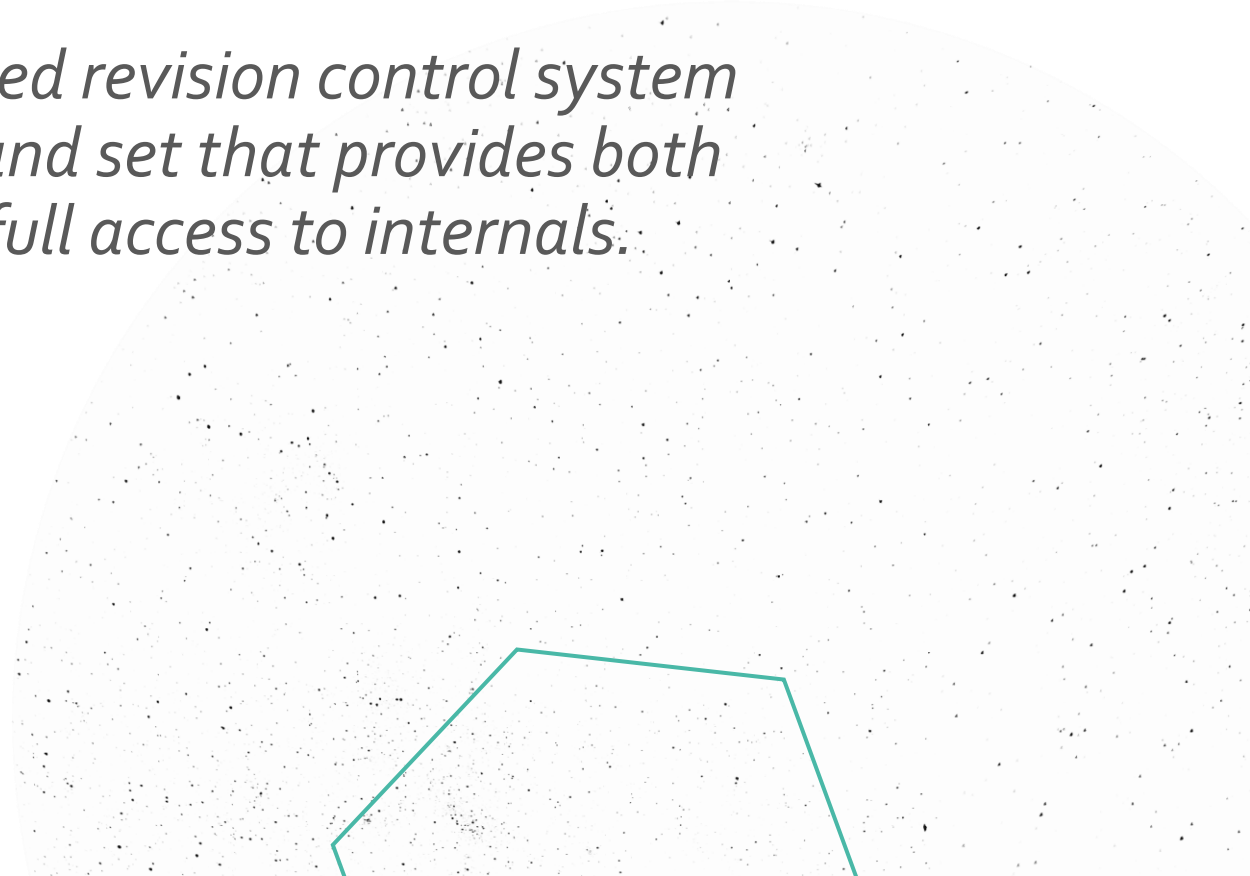
```
git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
    [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
    [-p|--paginate|-P|--no-pager] [--no-replace-objects] [--bare]
    [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
    [--super-prefix=<path>] [--config-env=<name>=<envvar>]
    <command> [<args>]
```

DESCRIPTION

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

See gittutorial[7] to get started, then see giteveryday[7] for a useful minimum set of commands. The Git User's Manual has a more in-depth introduction.

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.



*Git is a fast, scalable, distributed revision control system with an unusually rich **command set** that provides both high-level operations and full access to internals.*

Git is a **command line tool**.

What is the **most common thing** Git does?

Git is a dictionary

Git is a dictionary

```
var xyz: Dictionary of [Text, Boolean];
```

Git is a dictionary

```
var git: Dictionary of [?, ?];
```

Inside the dictionary

Value: Any **sequence of bytes**

```
57 68 65 72 65
27 73 20 57 61
6C 64 6F 3F 0A
```

Key: **SHA-1**

```
526405142abb3ed4745b9f66da0226172d03703d
```

Inside the dictionary

Value: Any **sequence of bytes**

```
57 68 65 72 65
27 73 20 57 61
6C 64 6F 3F 0A
```

Key: **SHA-1**

```
526405142abb3ed4745b9f66da0226172d03703d
```

```
git hash-object
```

Inside the dictionary

SHA-1

Same input **always** gives the same output

Git is a dictionary

```
git hash-object
```

Git is a **persisted** dictionary

```
git hash-object -w
```

git hash-object -w

content

```
Where’s Waldo?\x0a
```

git hash-object -w

header	content
blob 15\u0000	Where’s Waldo?\x0a

git hash-object -w

header	content
blob 15\u0000	Where’s Waldo?\x0a

hash
526405142abb3ed4745b9f66da0226172d03703d

git hash-object -w

compressed object

```
78 01 4b ca c9 4f 52 30 34 65 08 cf 48 2d 4a 55
2f 56 08 4f cc 49 c9 b7 e7 02 00 56 99 07 1b
```

hash

```
526405142abb3ed4745b9f66da0226172d03703d
```

git hash-object -w

compressed object

```
78 01 4b ca c9 4f 52 30 34 65 08 cf 48 2d 4a 55
2f 56 08 4f cc 49 c9 b7 e7 02 00 56 99 07 1b
```

52

6405142abb3ed4745b9f66da0226172d03703d

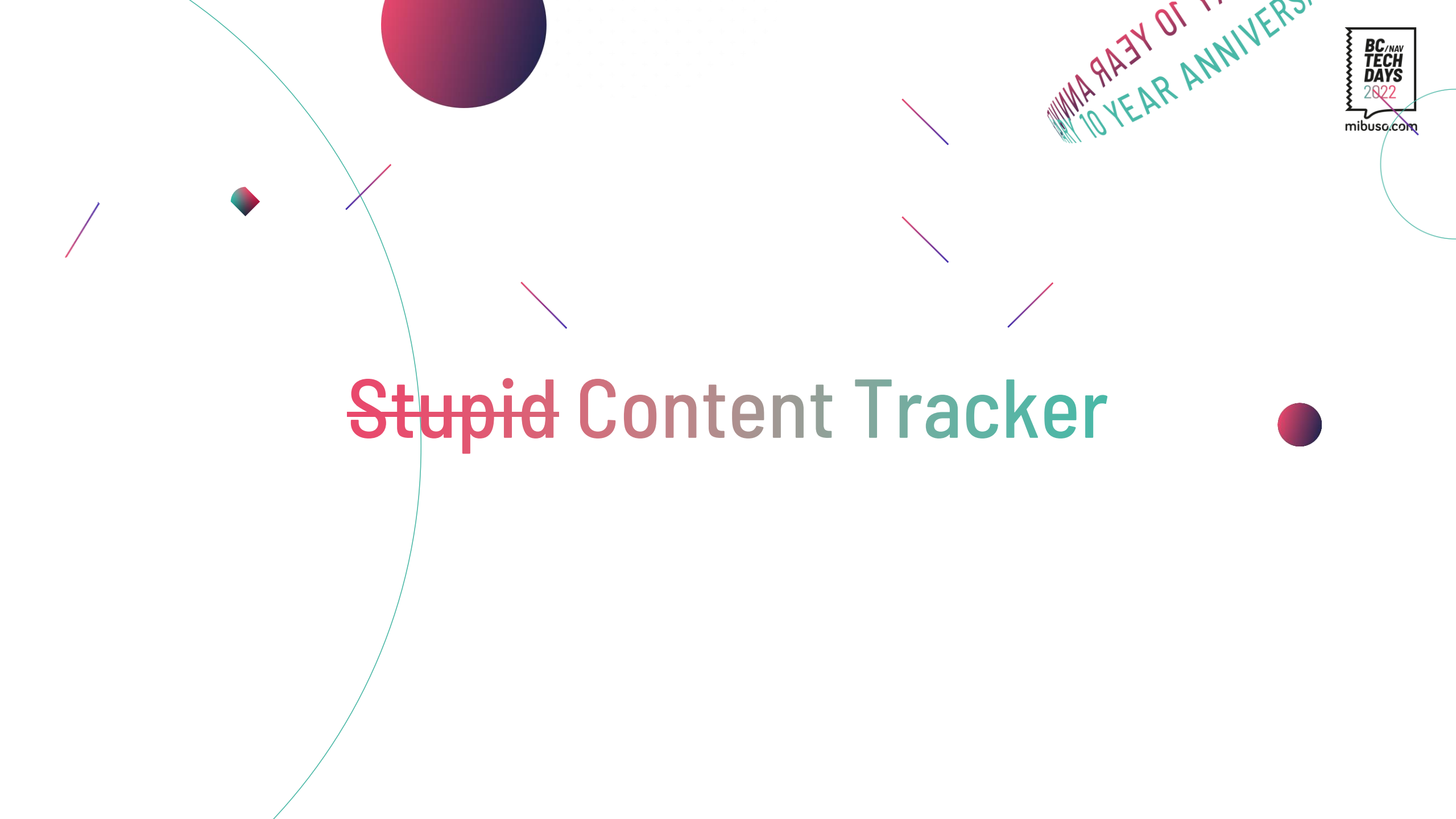
git hash-object -w

52

6405142abb3ed4745b9f66da0226172d03703d

A few persisted dictionary takeaways

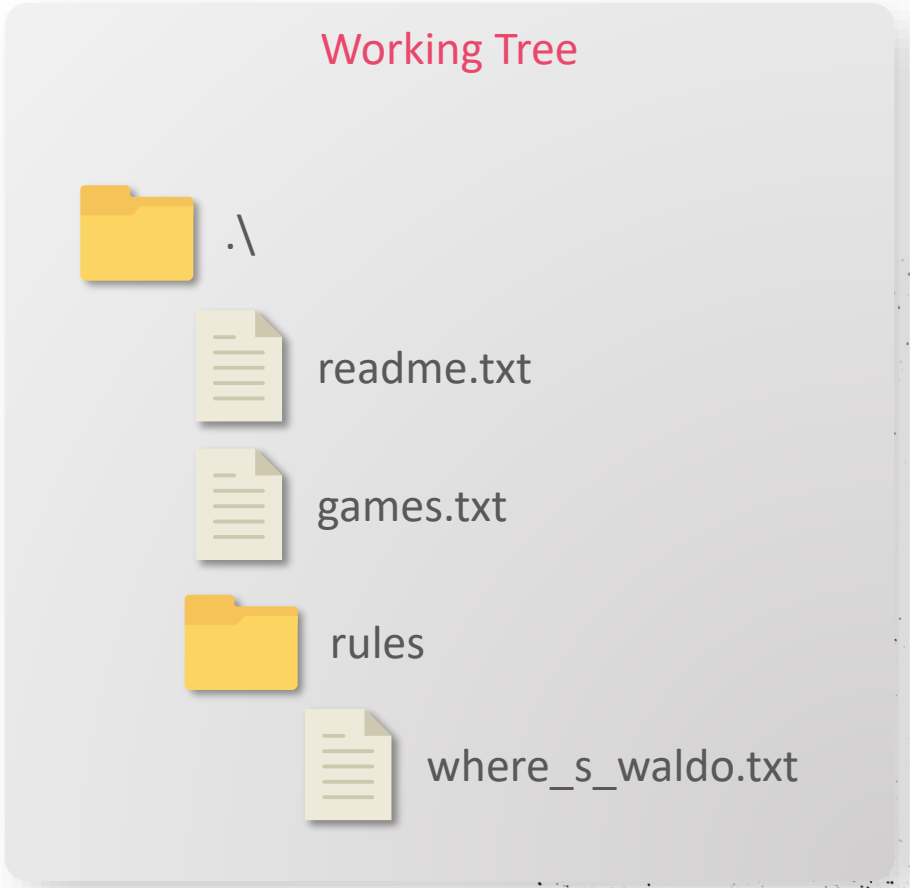
- Git works like a **dictionary**
- Value is any **data**, in **binary format**
- Key is **SHA-1**
- SHA-1 is **identity**, it identifies content **globally**
- Objects in the dictionary are **immutable**



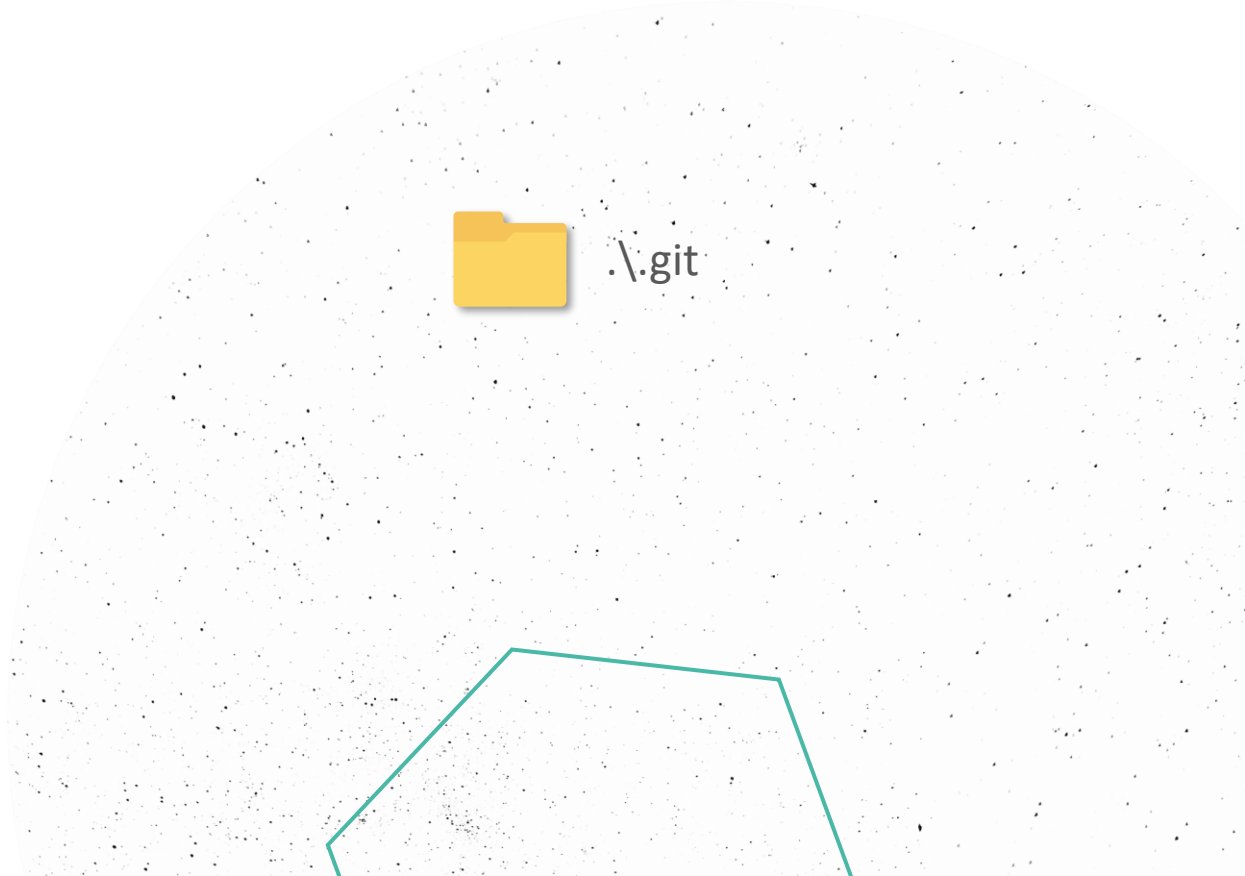
~~Stupid~~ Content Tracker

Git does not **monitor** source code

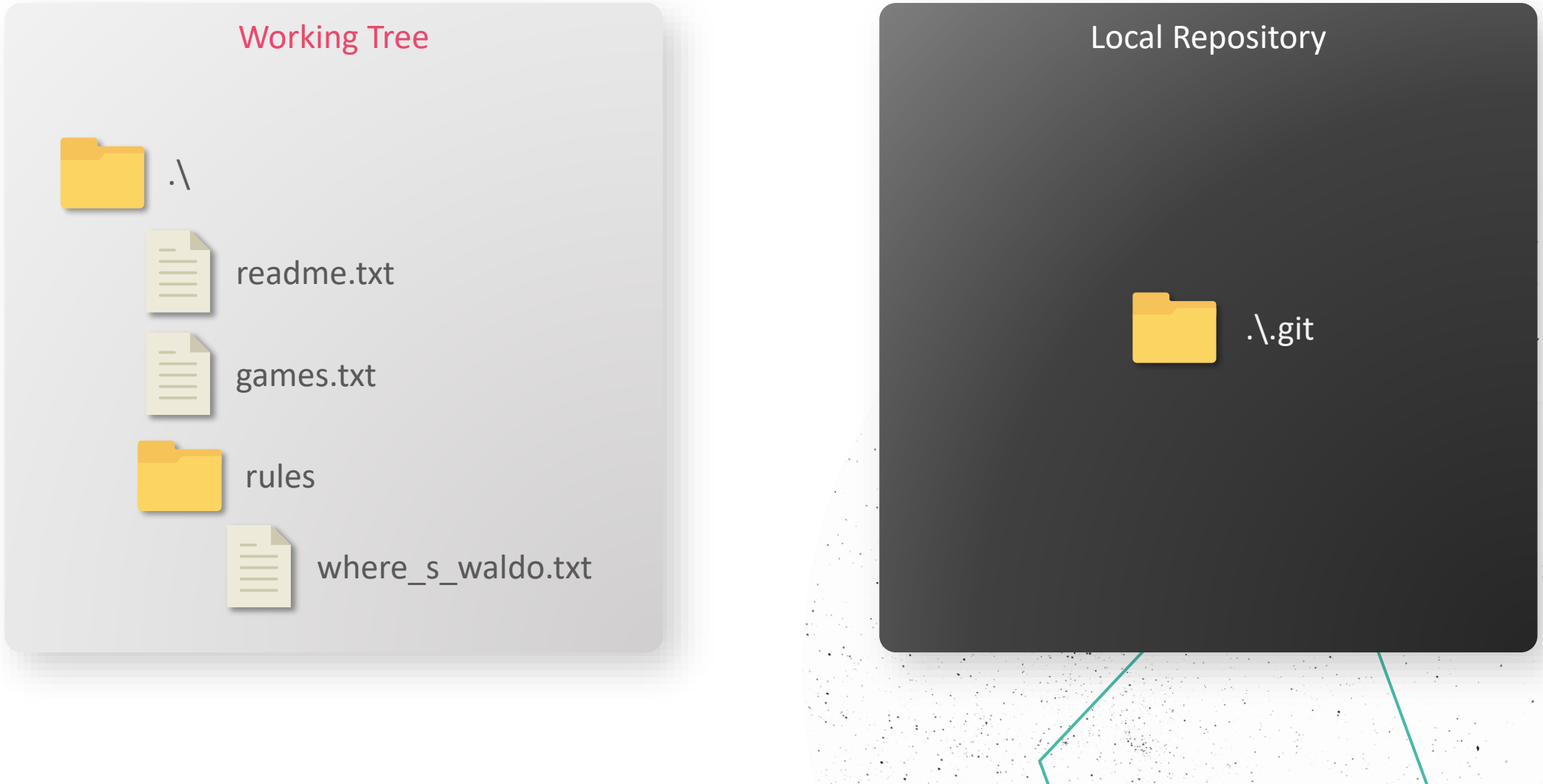
Three areas of Git



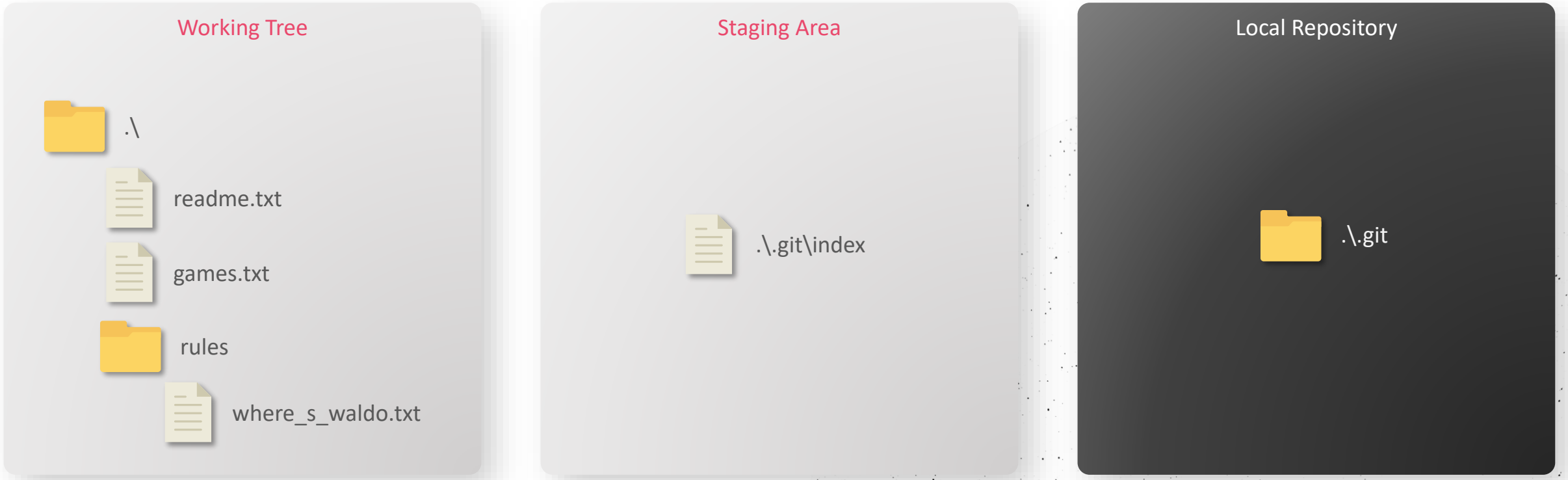
Three areas of Git



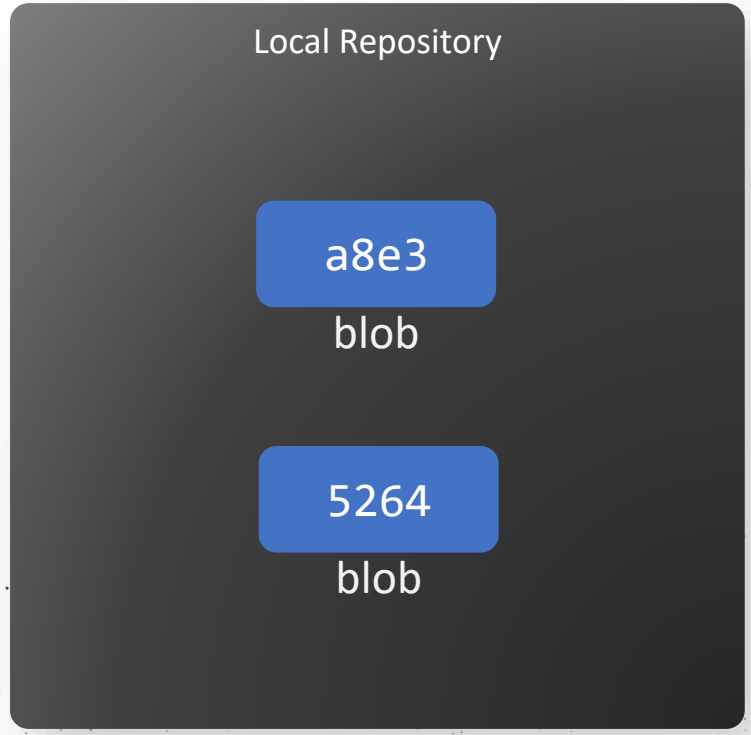
Three areas of Git



Three areas of Git

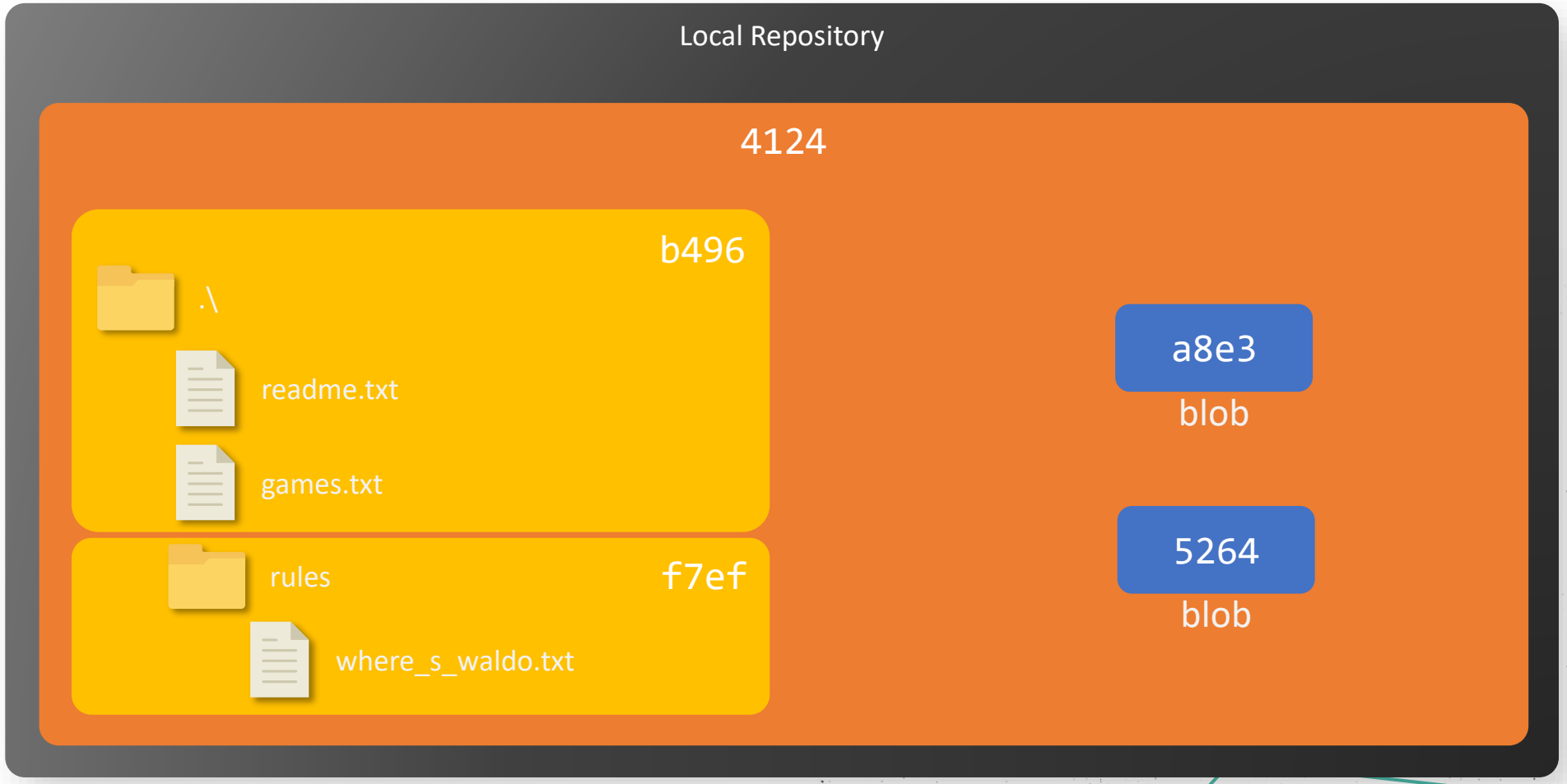


What happens during staging?

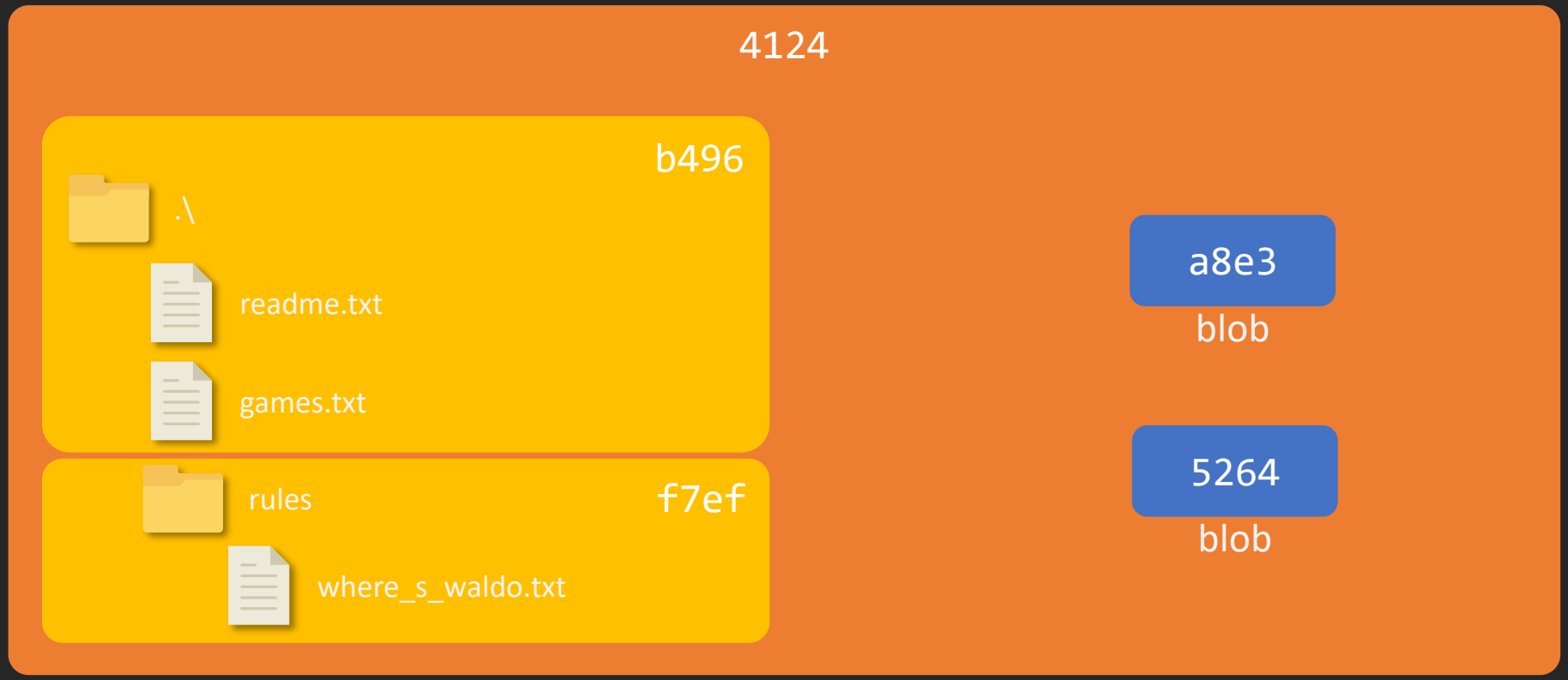


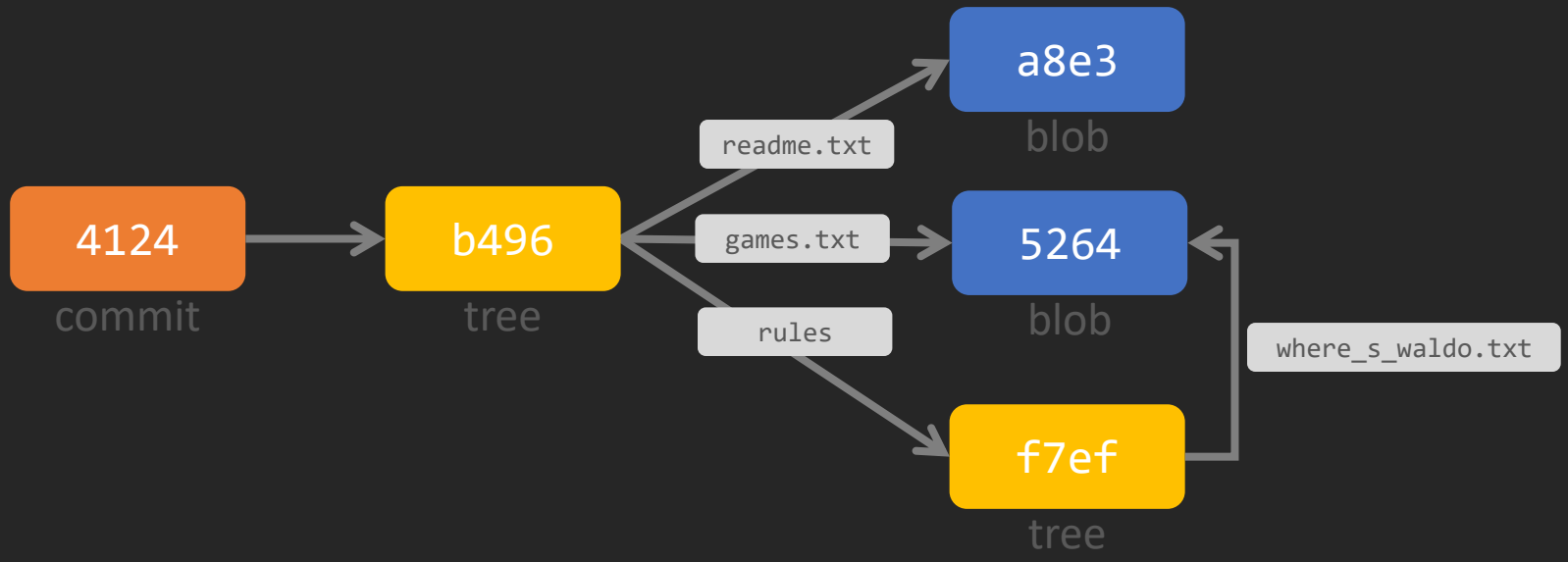

```
git add .\readme.txt .\games.txt .\rules\where_s_waldo.txt
```

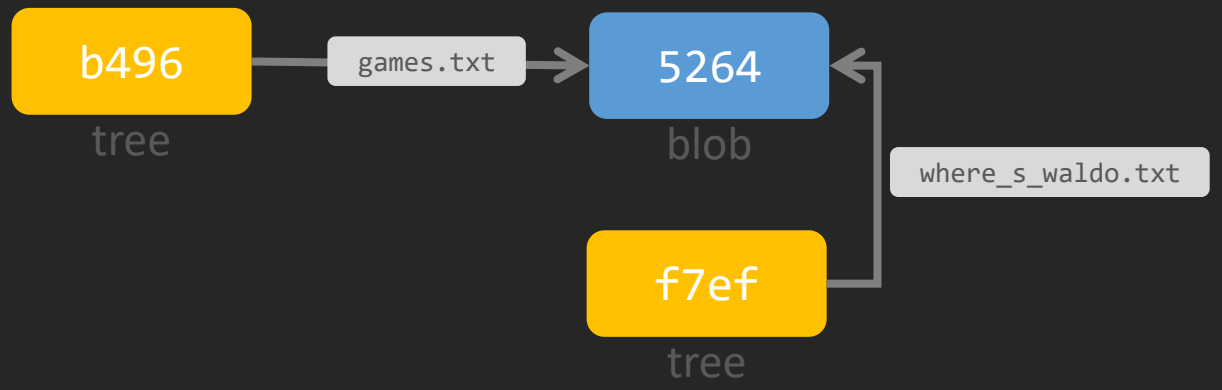
What happens during committing?




```
git commit -m "First commit"
```



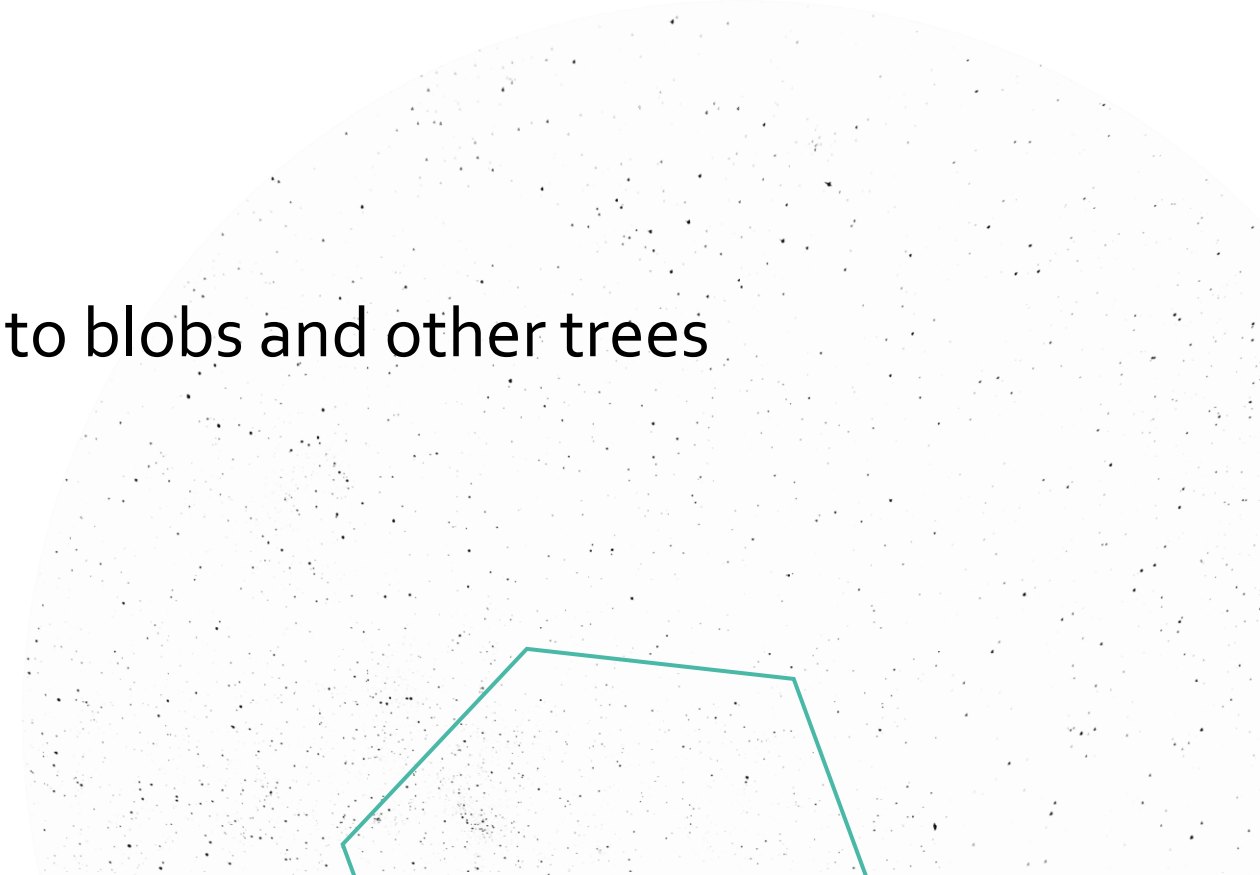



b496
tree
games.txt
5264
blob
where_s_waldo.txt
f7ef
tree

A few ~~stupid~~ content tracker takeaways

BC/NAV TECH DAYS 2022
mibuso.com

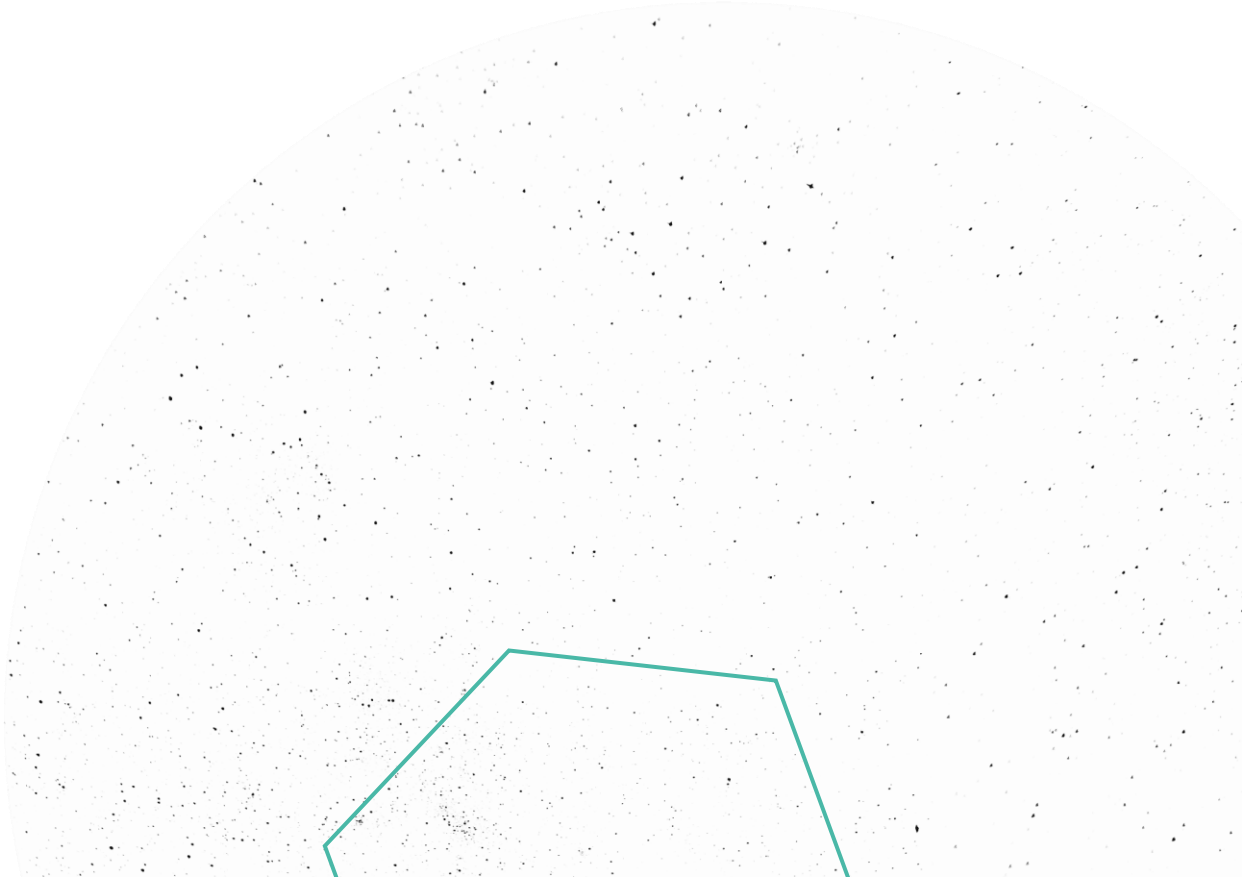
- Index is **blueprint for next commit**
- Commit creates **commit object**
- Commit object **points to a tree**
- Trees are catalogs of **named pointers** to blobs and other trees
- Git **hates duplication**



Revision Control System

What is a revision control system

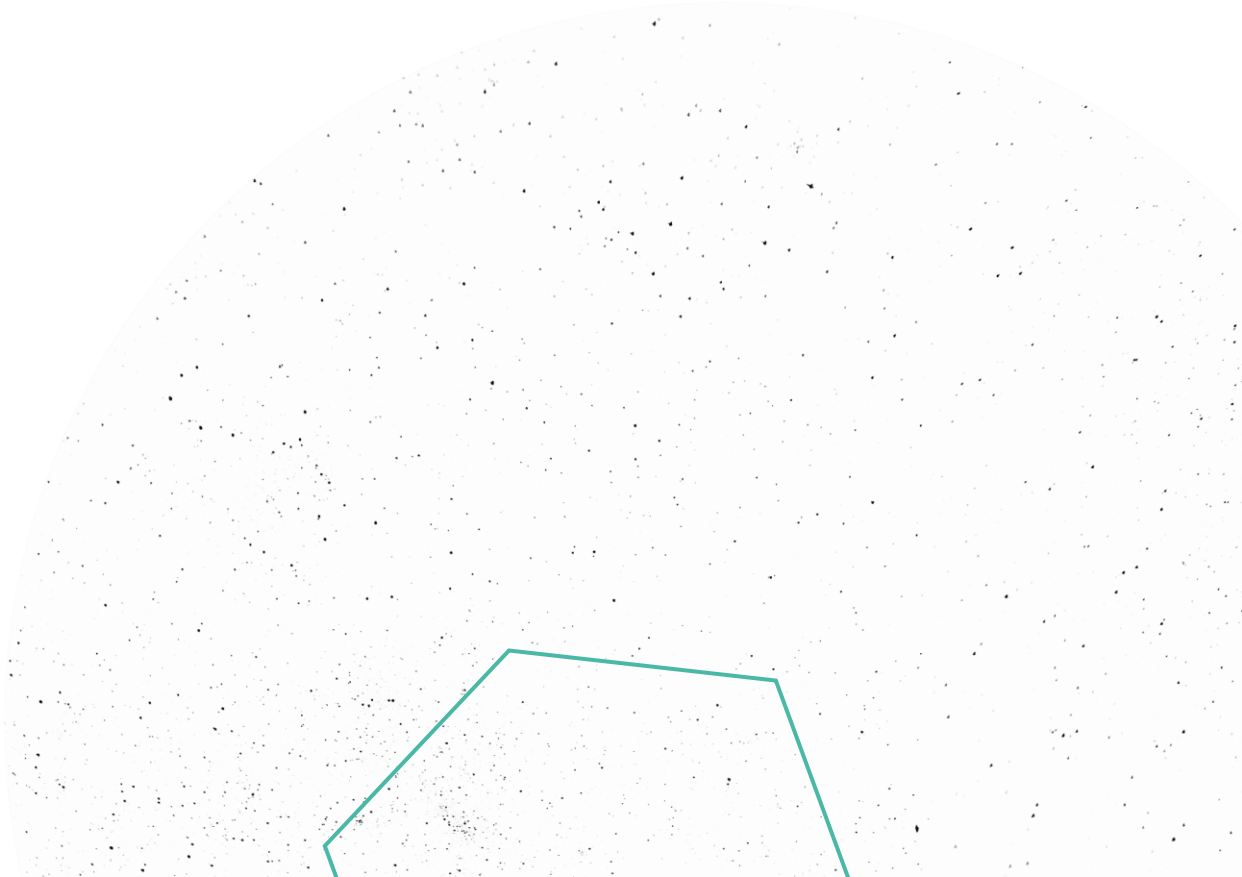
- Manages changes of data over time
- Identifies differences between revisions
- Allows isolation of changes
- Allows integration of changes

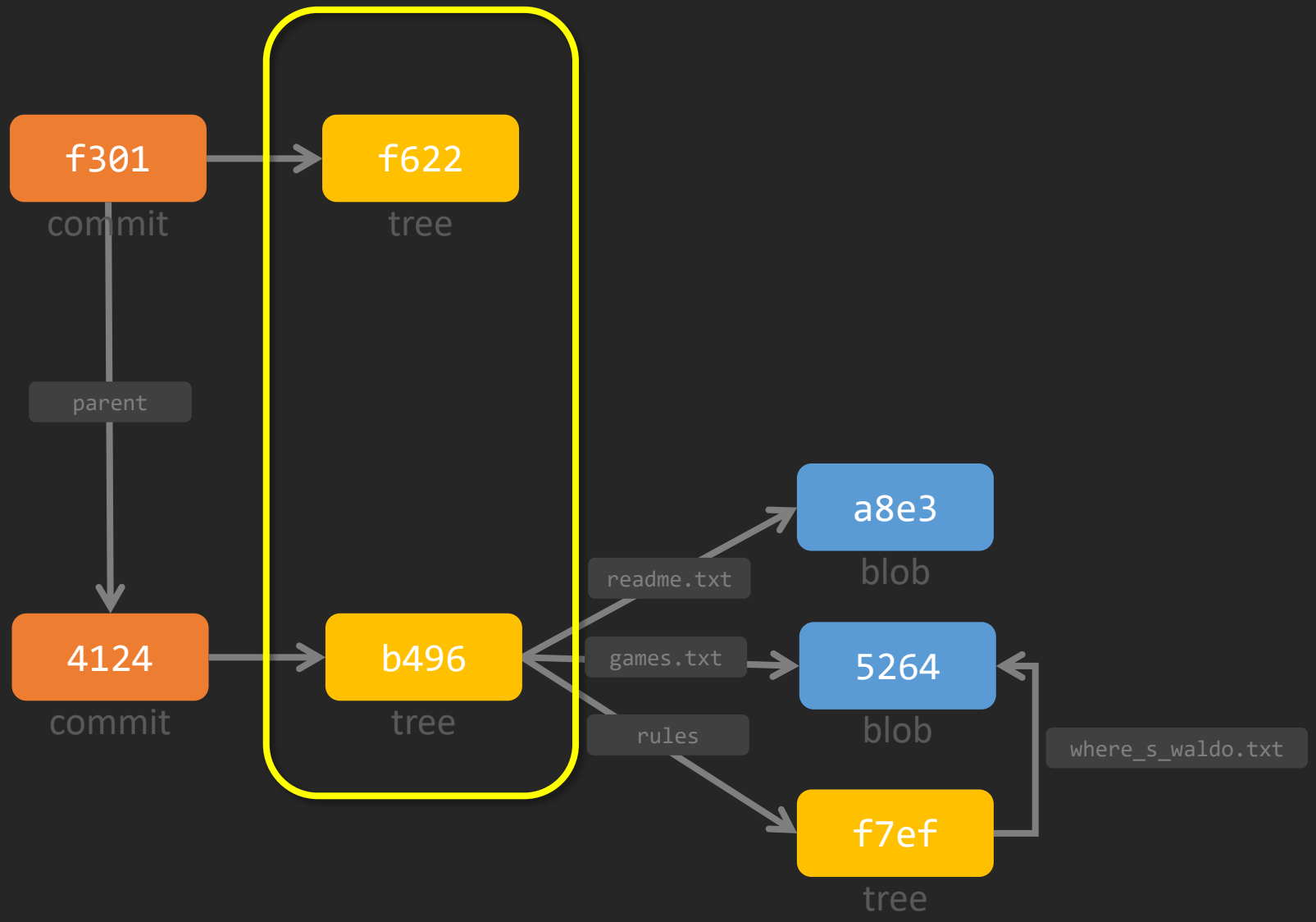


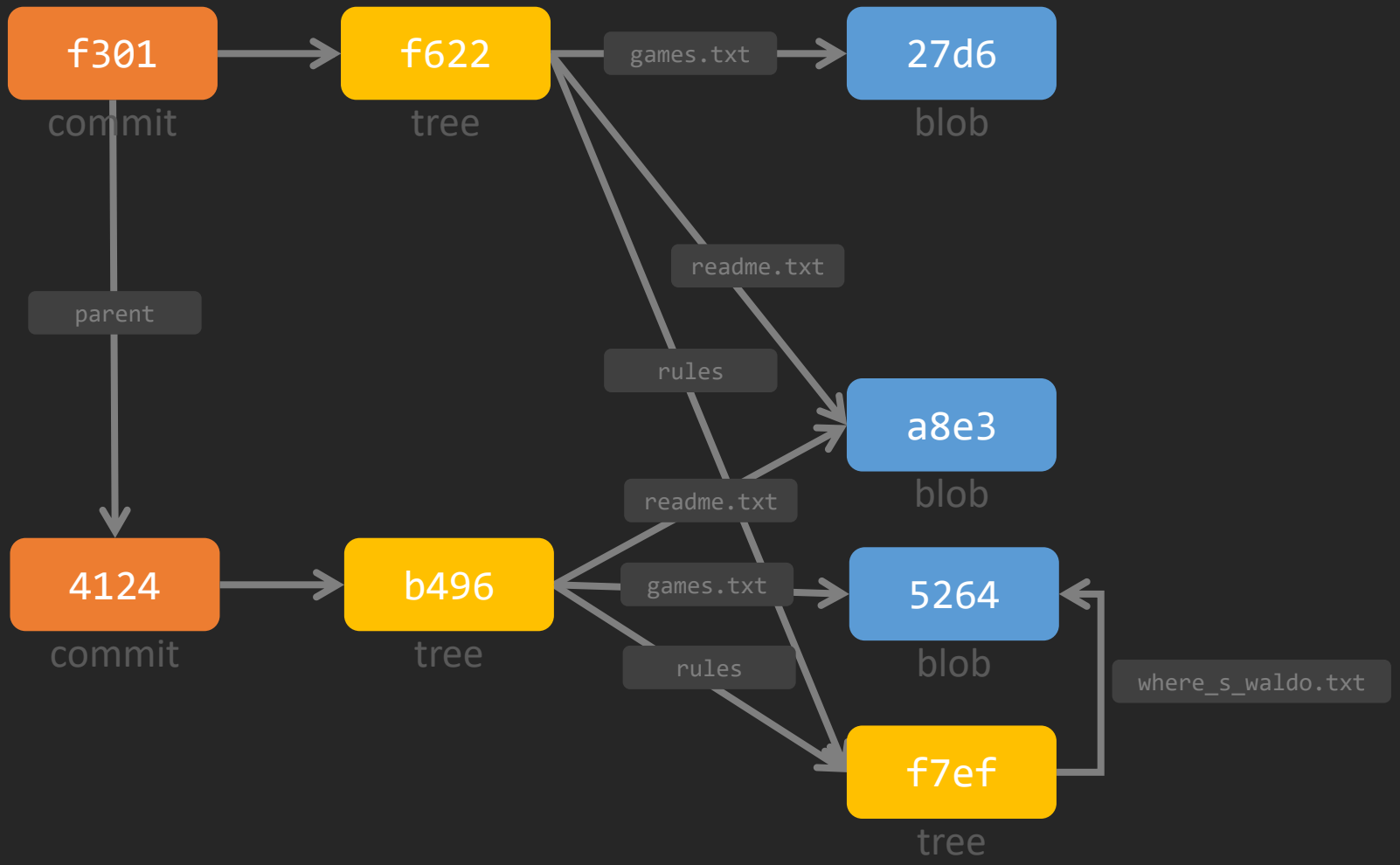
Git as a revision control system

BC/NAV TECH DAYS 2022
mibuso.com

- Manages history of commits
- Identifies differences between commits
- Allows isolation through branches
- Allows integration of branches







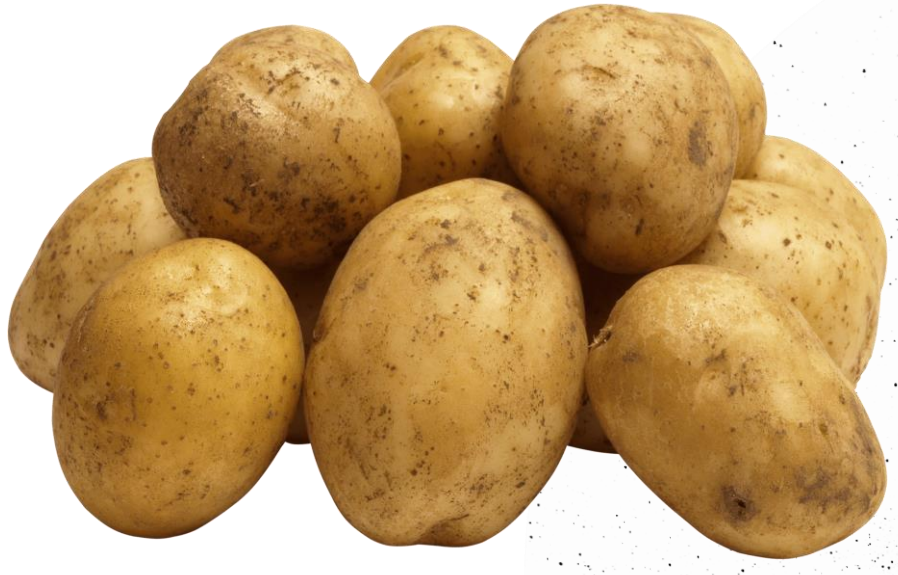
Common misconceptions about Git

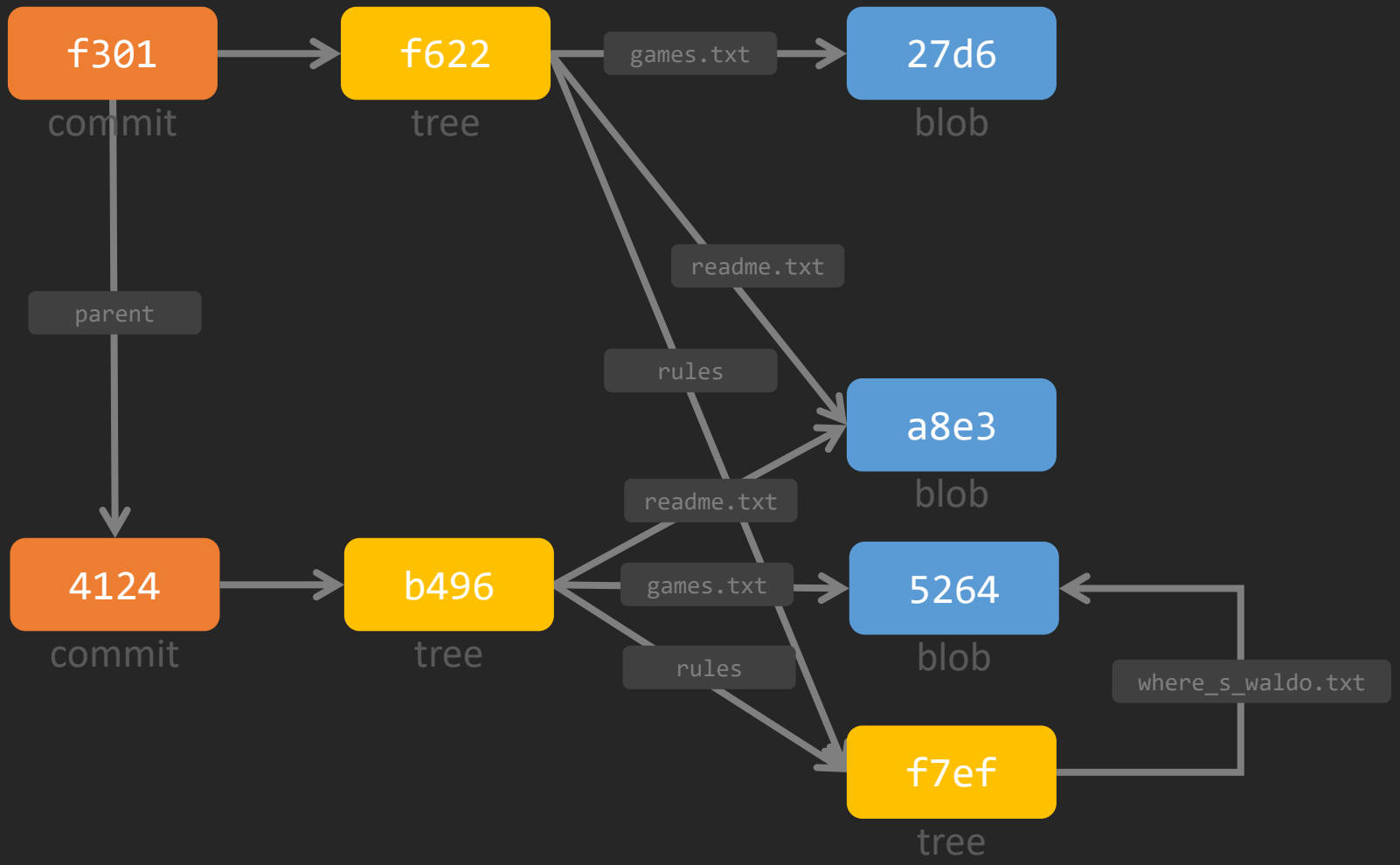
- Git stores deltas
- Git tracks changes

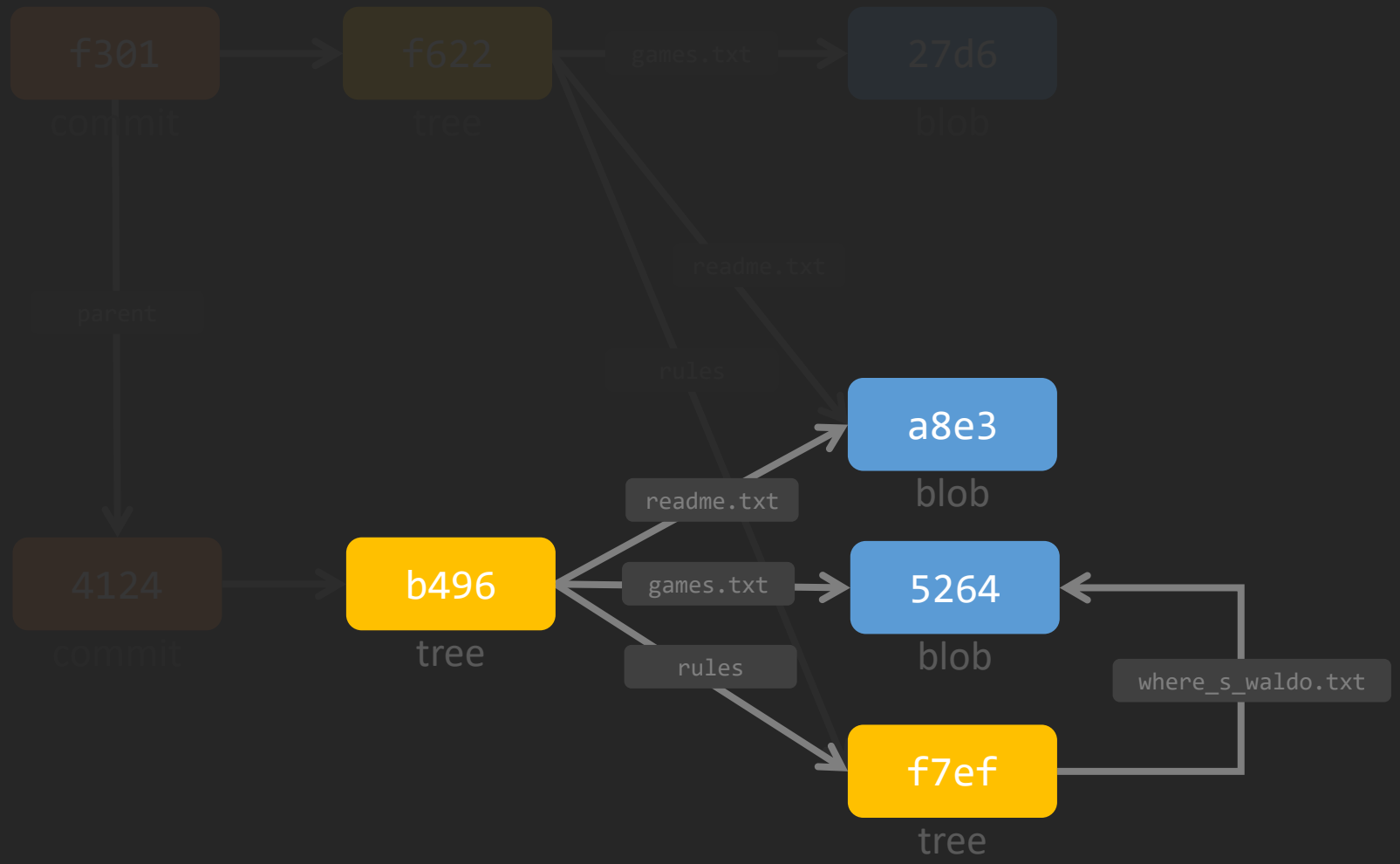
Common misconceptions about Git

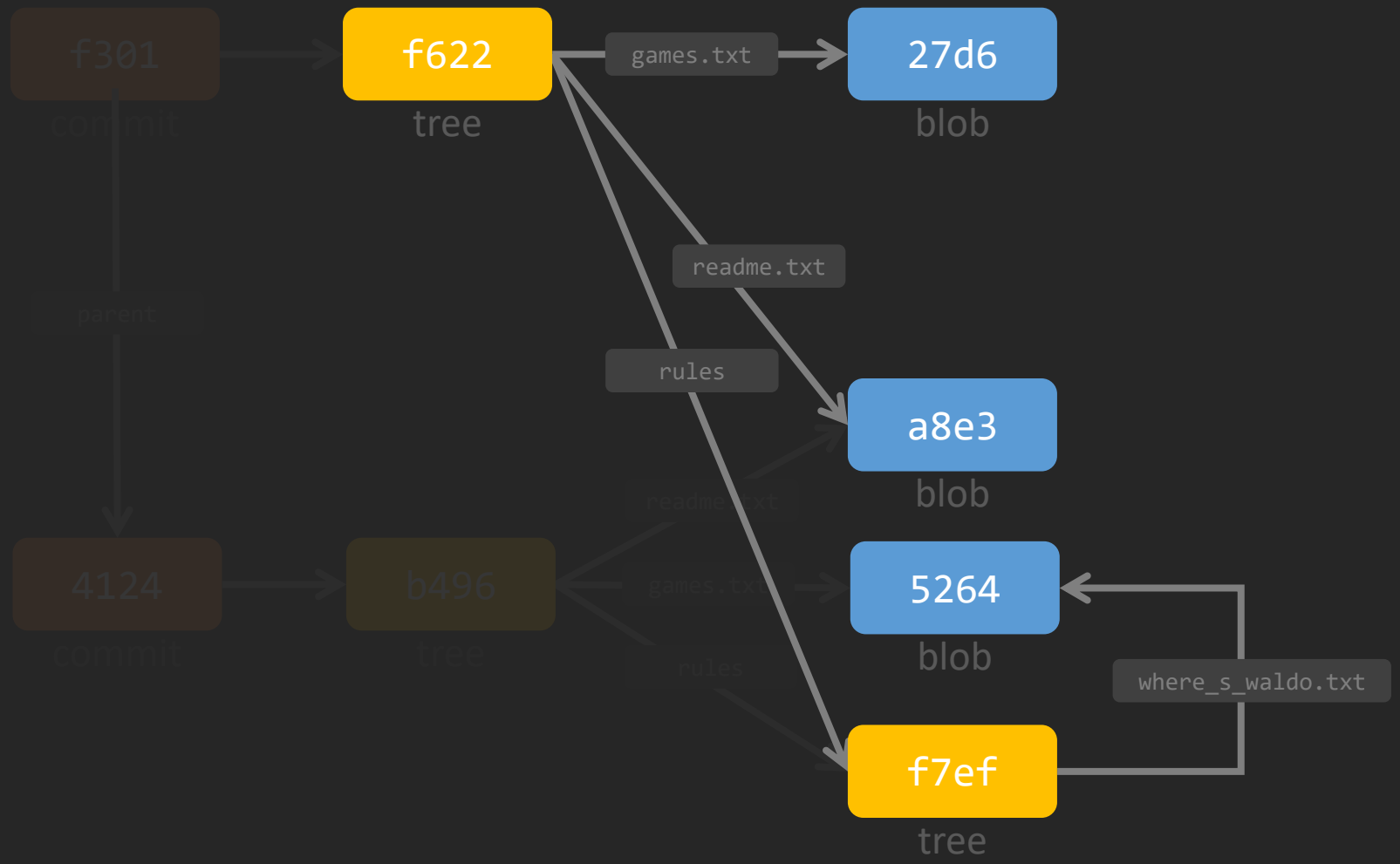
BC/NAV TECH DAYS 2022
mibuso.com

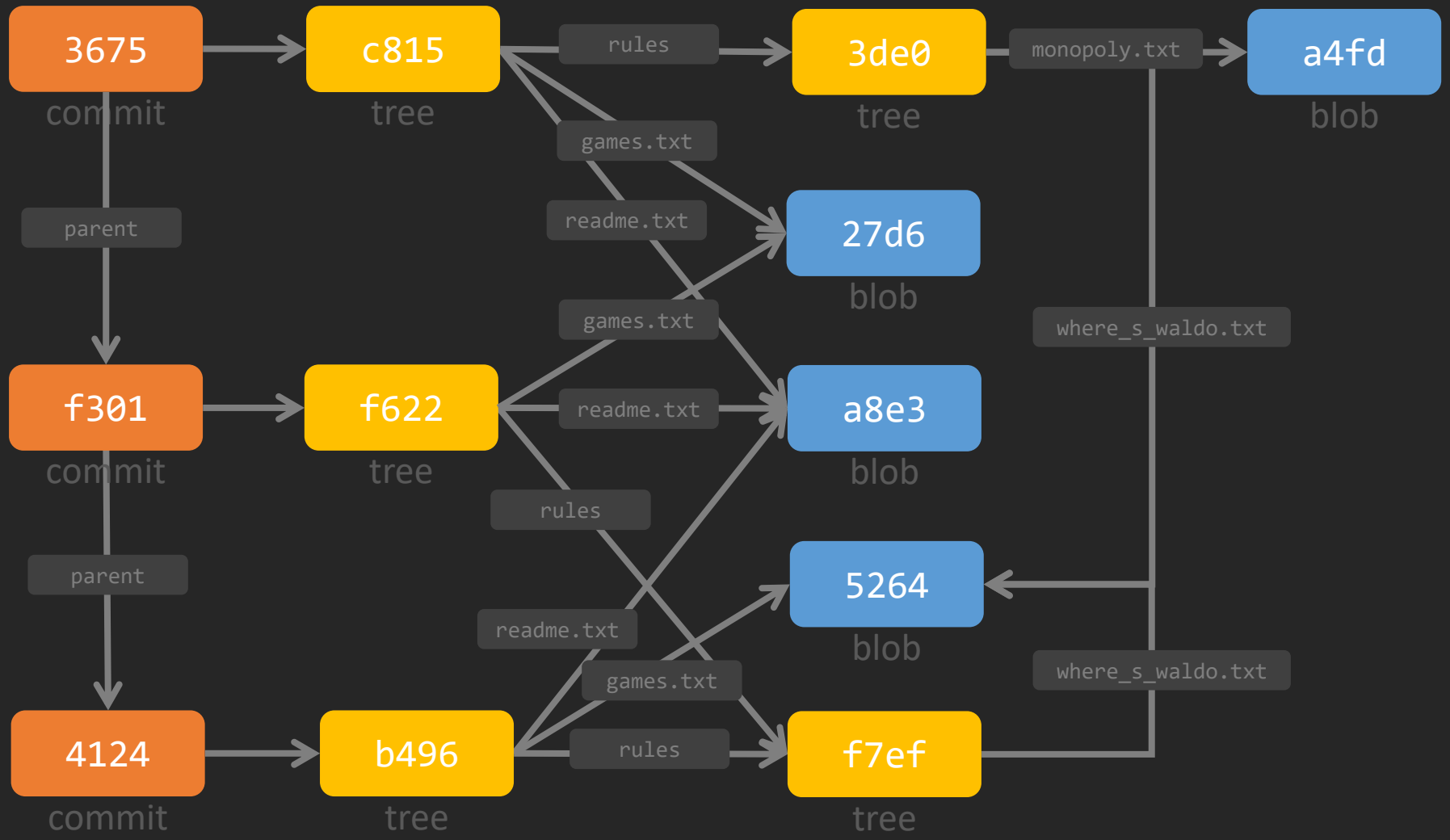
- Git stores ~~deltas~~ ***snapshots***
- Git tracks ~~changes~~ ***content***

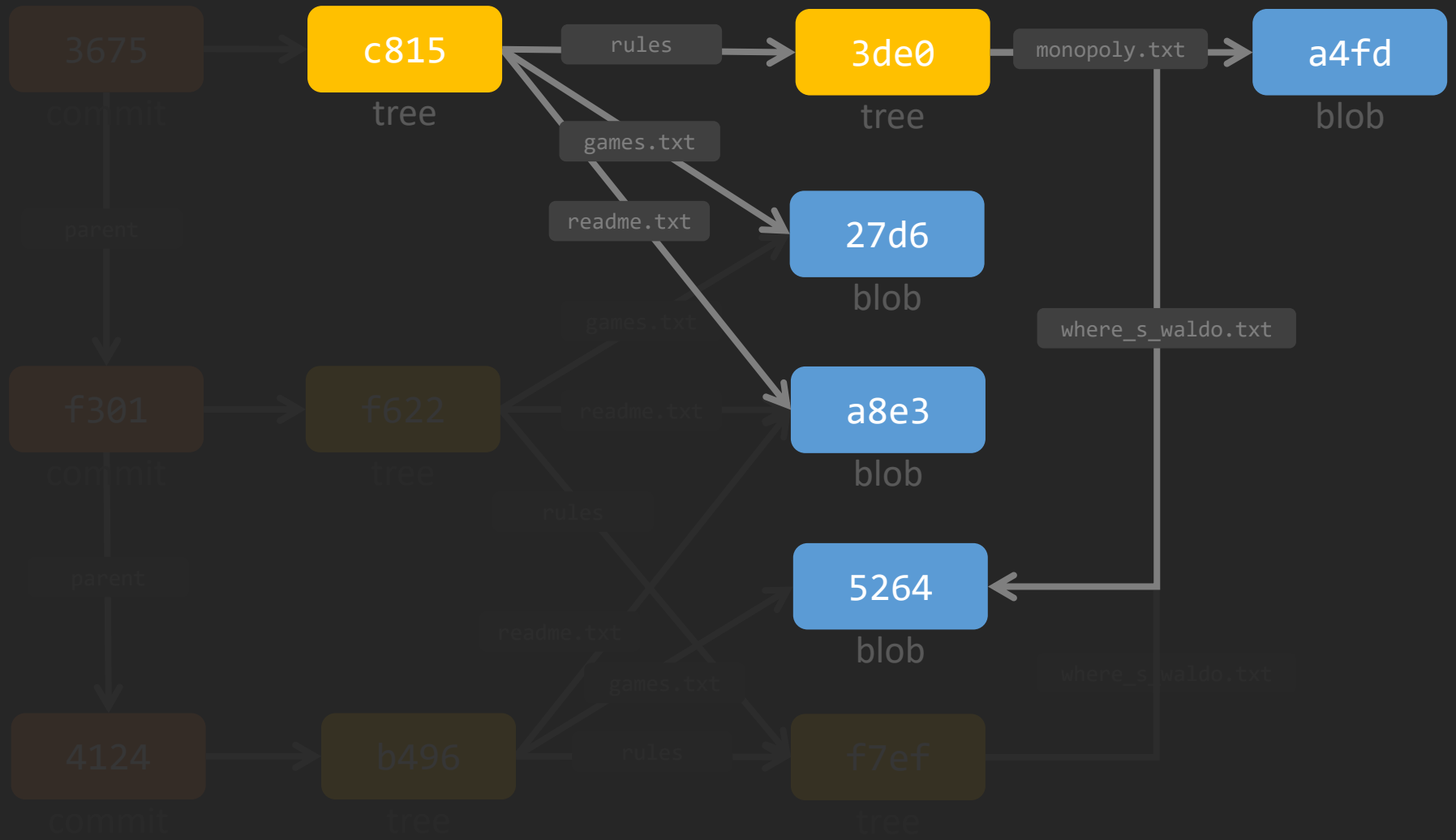






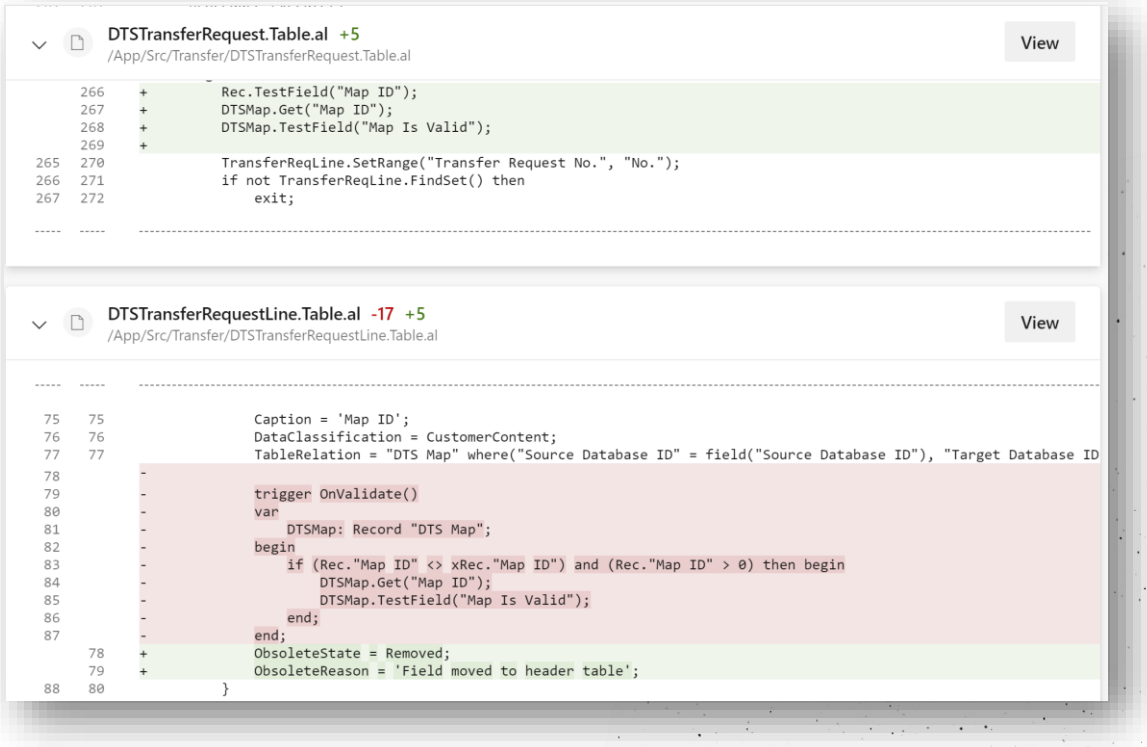






But… *how?*

- You ***know*** that Git stores differences, you've seen this:



- So, Git ***does*** track changes, doesn't it?

Git vs. Git tools

- Humans are ***bad*** at handling snapshots
- Humans are ***good*** at understanding change
- Git tools ***translate snapshots into changes***
- All Git tools present history of Git snapshots as a series of changes.
  It's just ***easier for us*** to follow that.

`git diff`

- Works out and ***reports changes*** between two snapshots
- Takes ***massive*** advantage of SHA-1!
- Whenever you want to know what changed, Git tools call `git diff`

git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

BC/NAV TECH DAYS 2022
mibuso.com

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

BC/NAV TECH DAYS 2022
mibuso.com

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

git diff

BC/NAV TECH DAYS 2022
mibuso.com

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

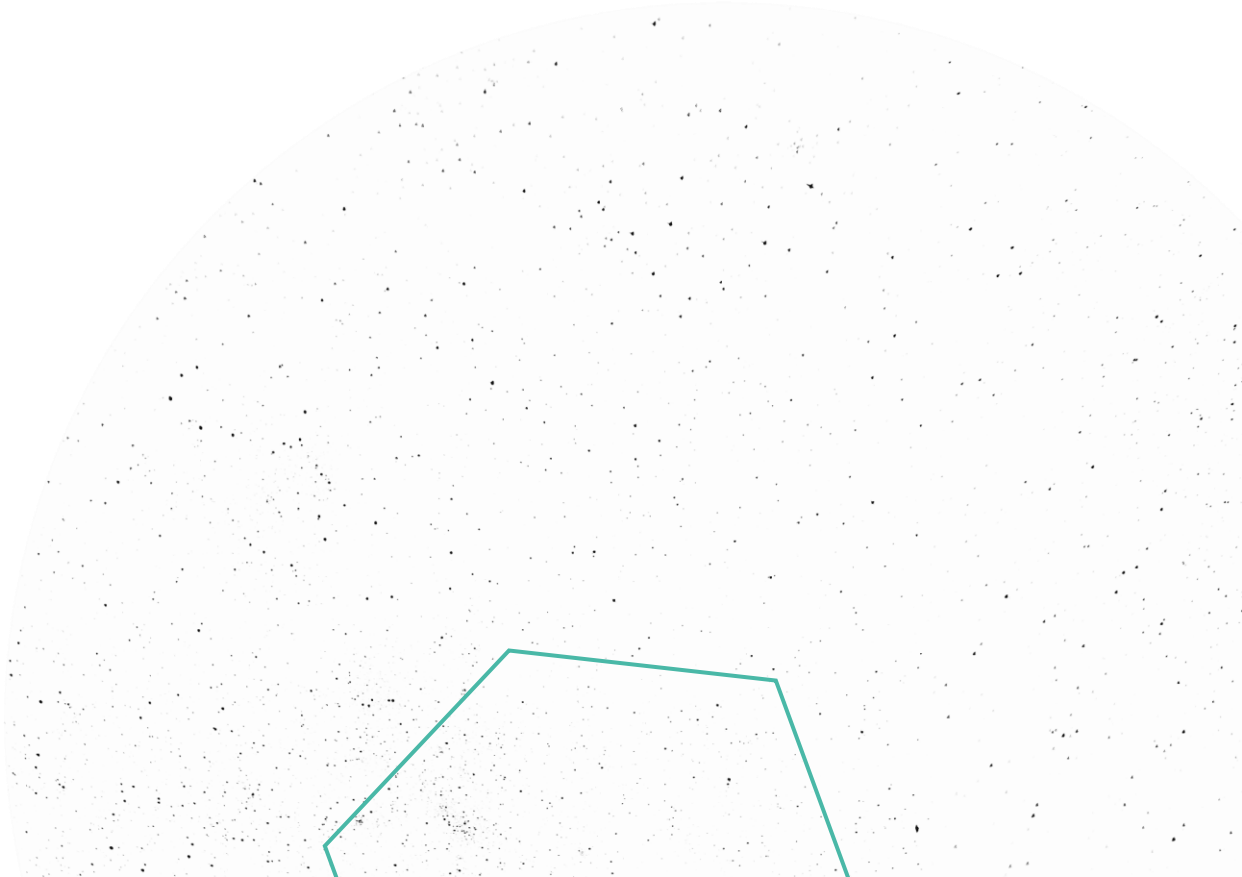
git diff

First Commit		1ead
Name	**Type**	**SHA-1**
.\	tree	c81f
.objidconfig	blob	74ee
app.json	blob	20b5
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Second	tree	a792
Second.Codeunit.al	blob	e191
Second.Table.al	blob	3010

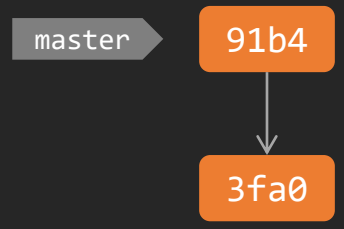
Second Commit		983b
Name	**Type**	**SHA-1**
.\	tree	3141
.objidconfig	blob	74ee
app.json	blob	6f0f
\src\First	tree	49d1
First.Page.al	blob	7c13
First.Table.al	blob	5119
\src\Third	tree	0e15
Third.Codeunit.al	blob	e191
Third.Page.al	blob	818e

Branches

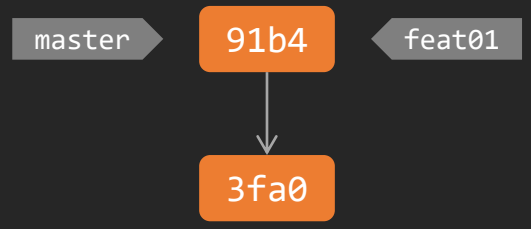
- Independent line of development
- Isolates work between developers
- In Git, they are pretty smart



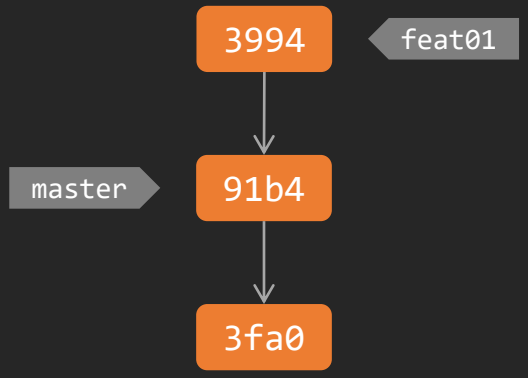
What is a branch?



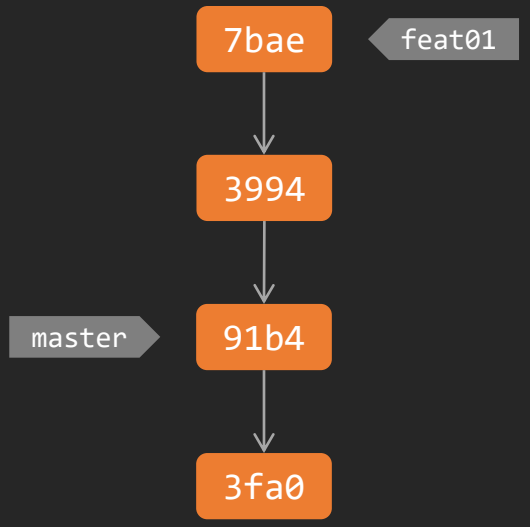
What is a branch?



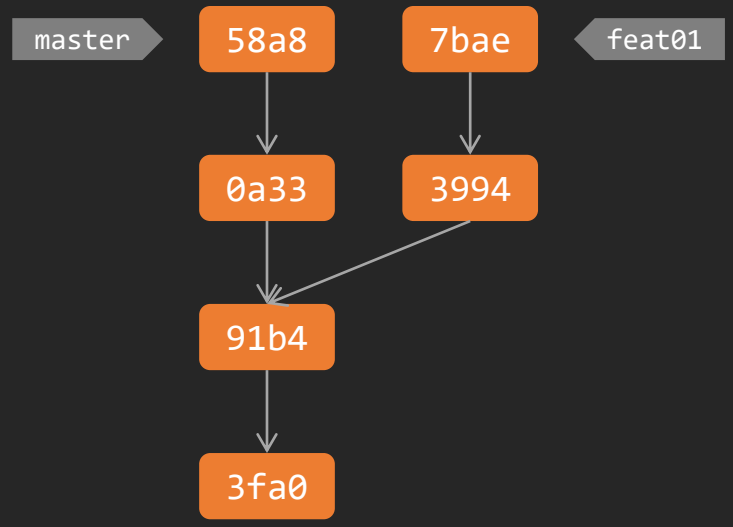
What is a branch?



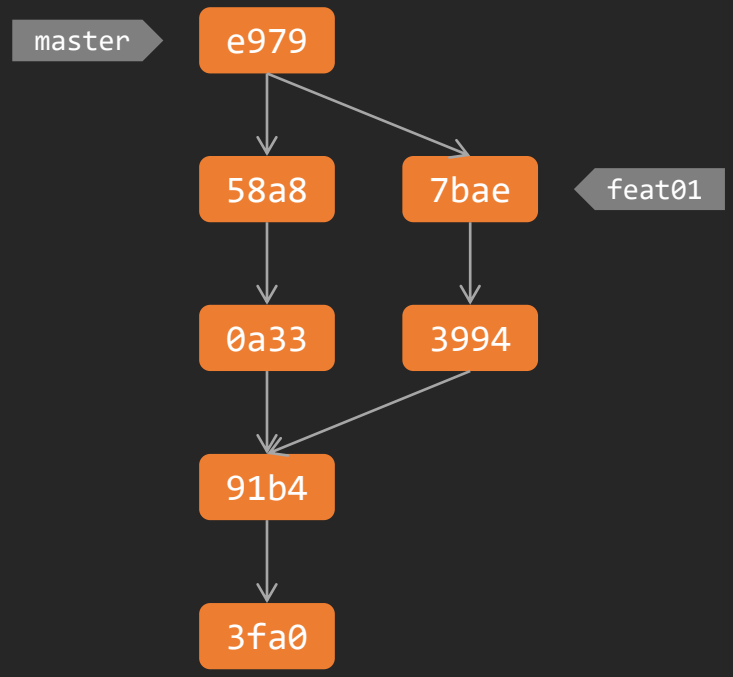
What is a branch?



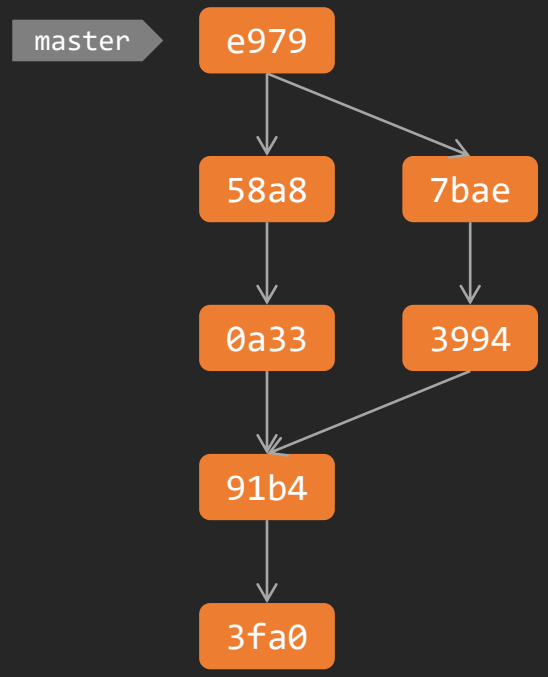
What is a branch?



What is a branch?

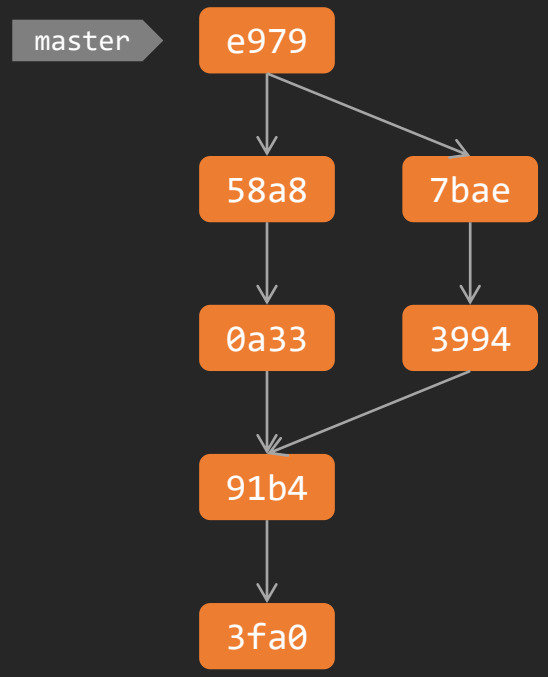


What is a branch?

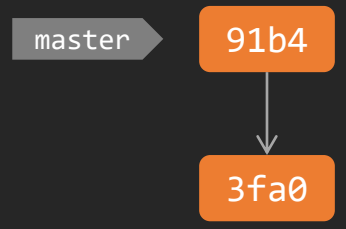


What is a branch?

Branch is **name for commit**

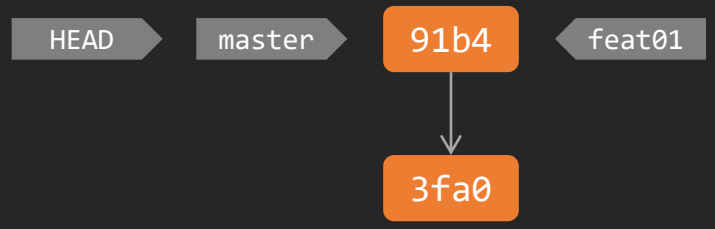


How does Git know which branch is current?

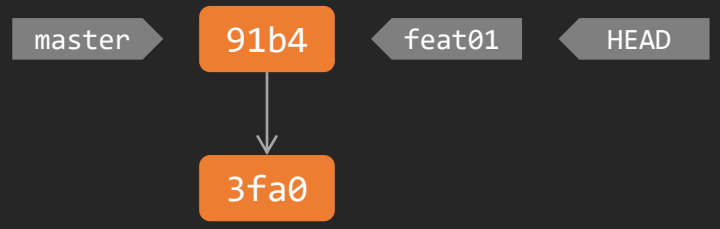


How does Git know which branch is current?

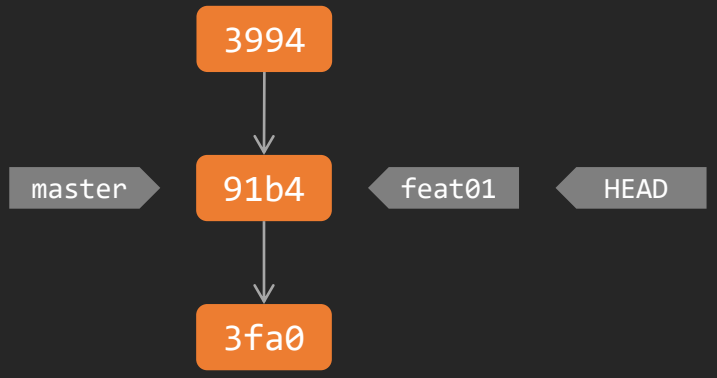
How does Git know which branch is current?



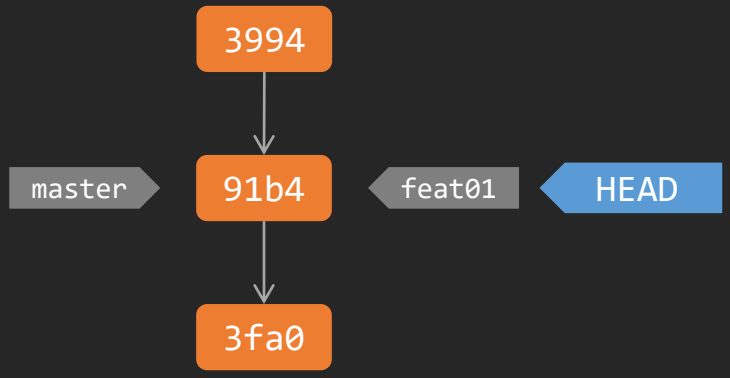
How does Git know which branch is current?



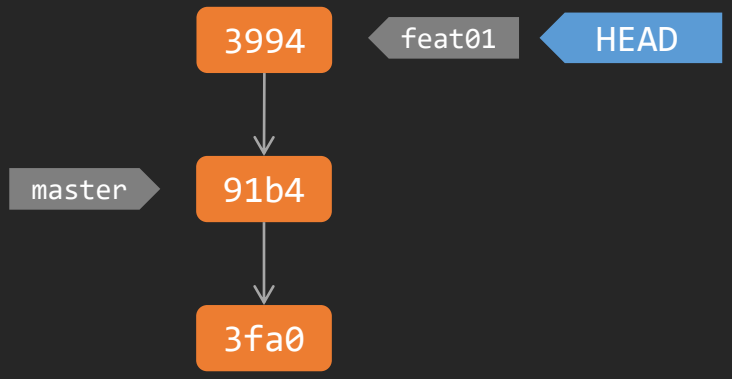
How does Git know which branch is current?



How does Git know which branch is current?

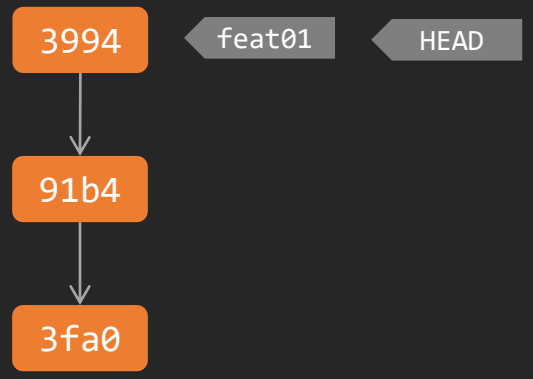


How does Git know which branch is current?

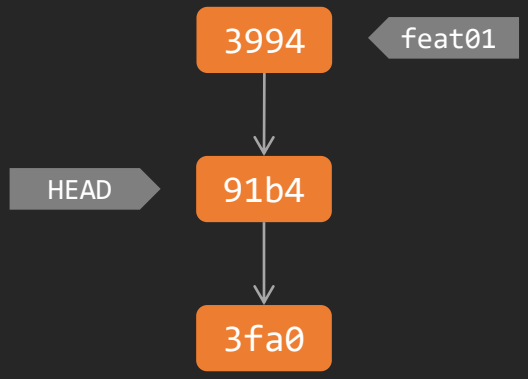


Detached head

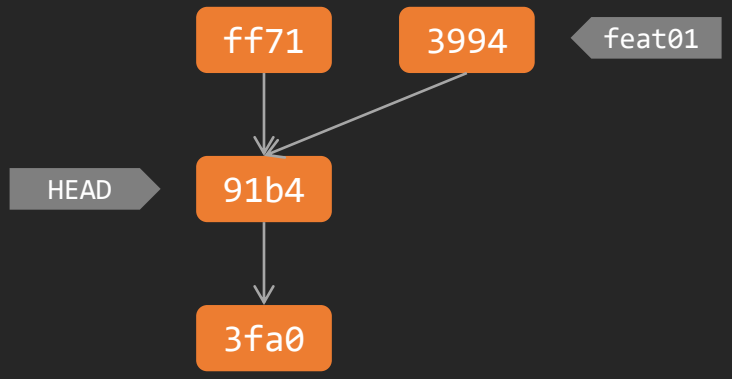
Detached head



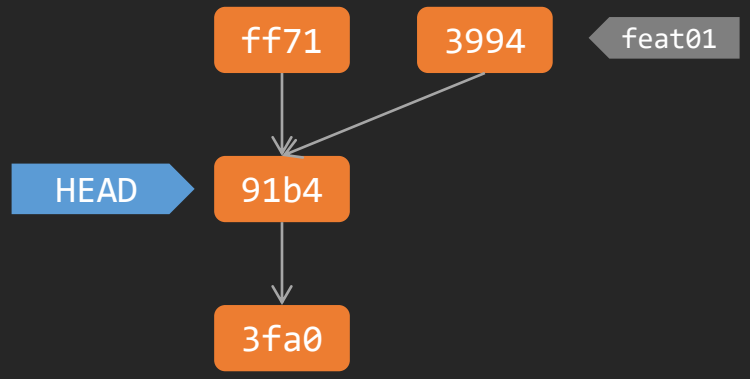
Detached head



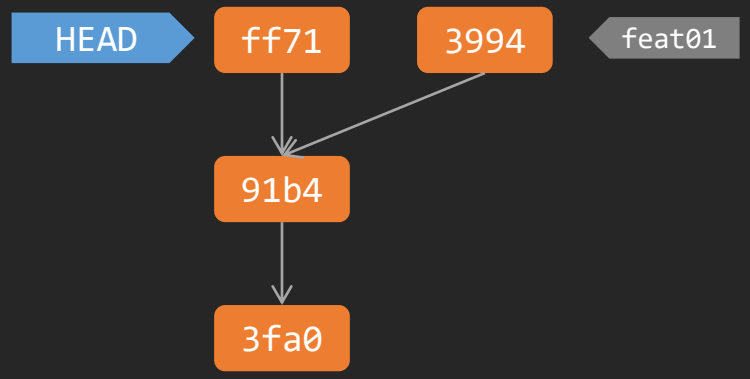
Detached head



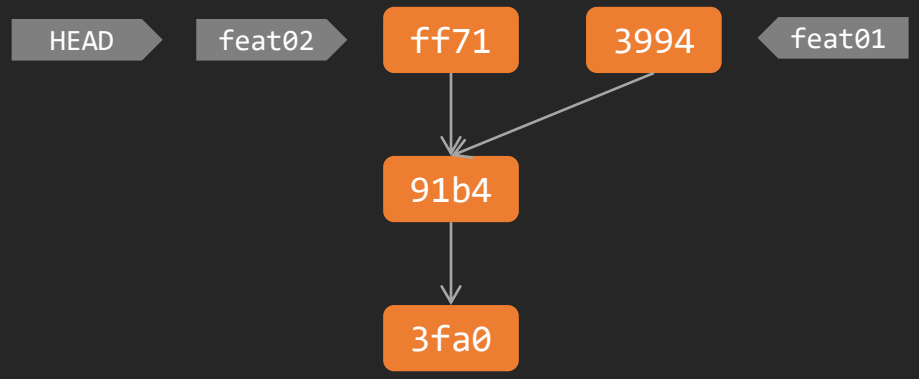
Detached head



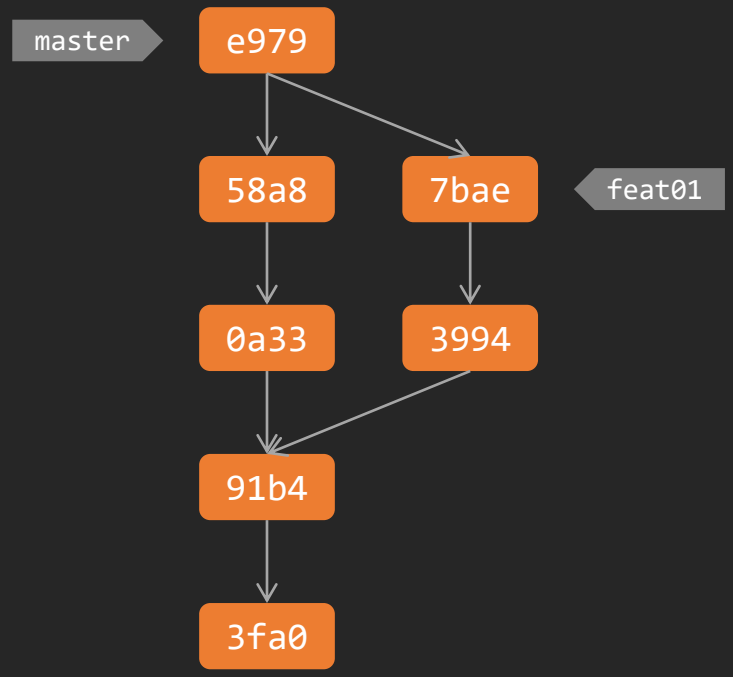
Detached head



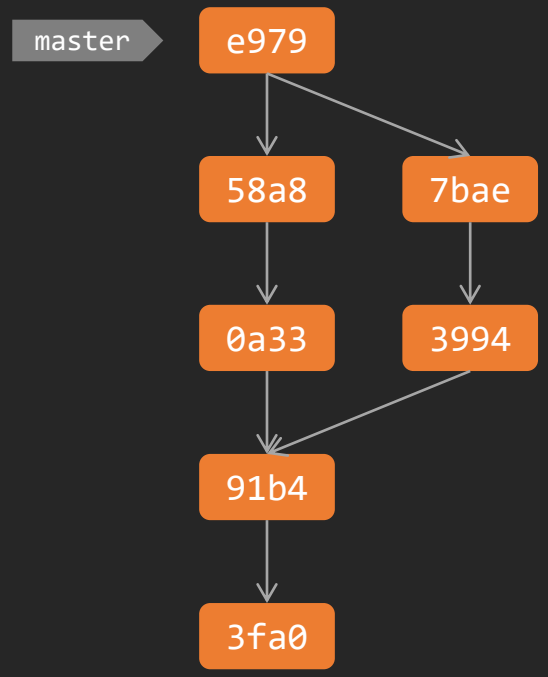
Detached head



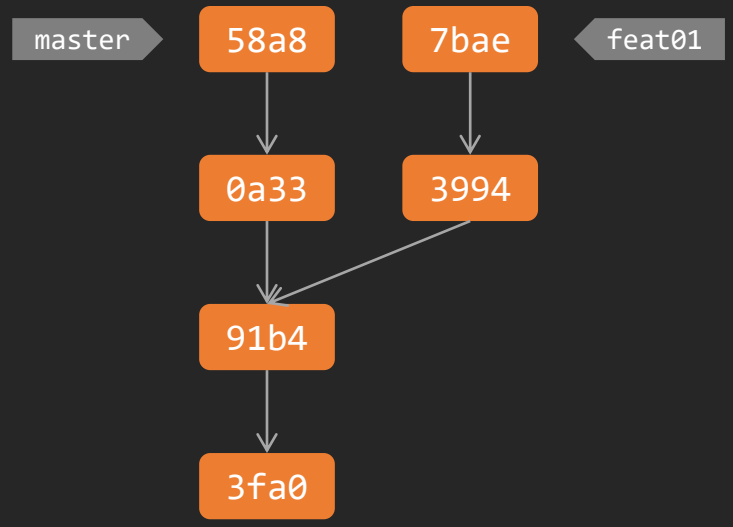
Deleting a branch



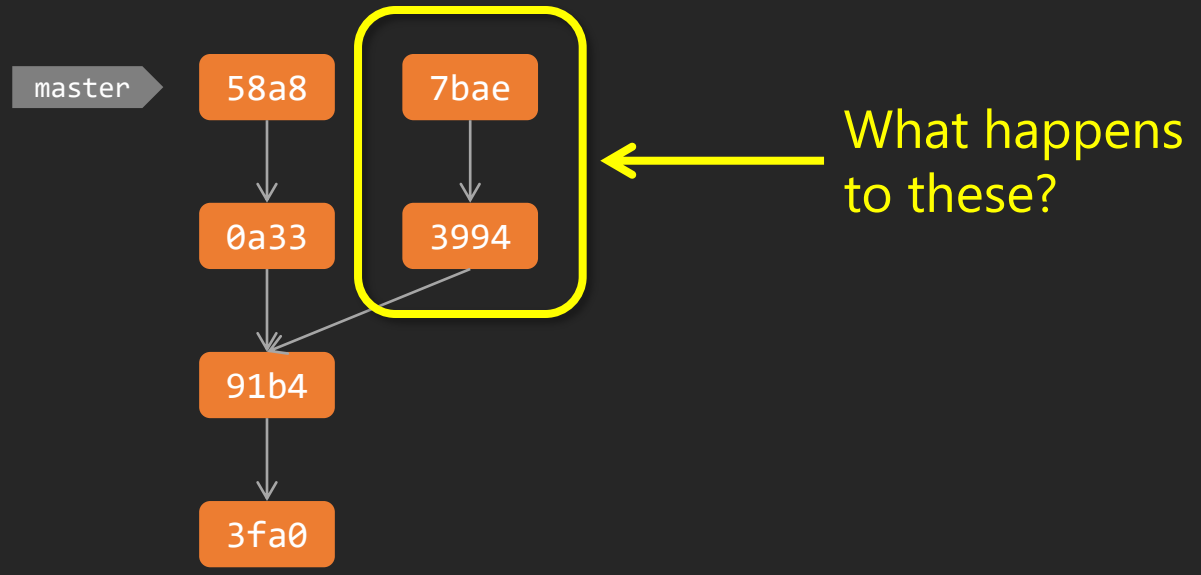
Deleting a branch



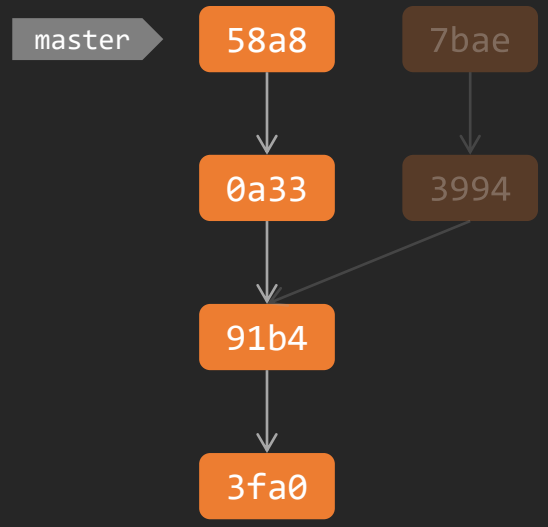
Deleting a branch



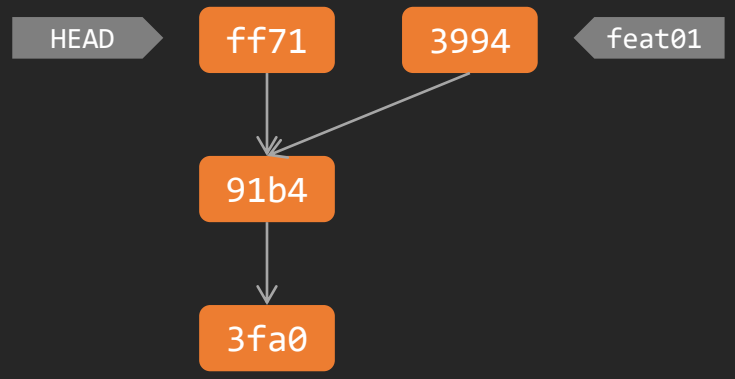
Deleting a branch



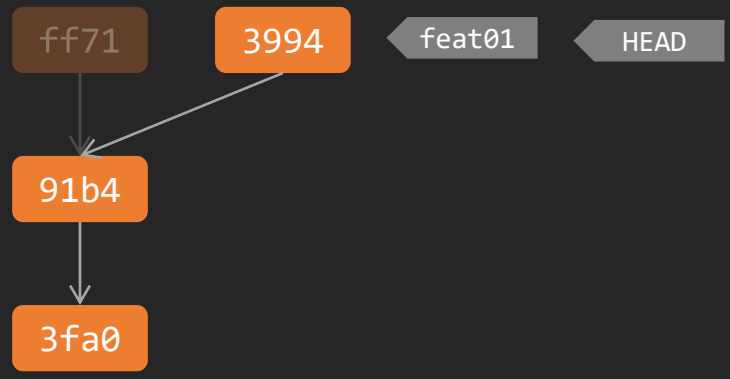
Deleting a branch



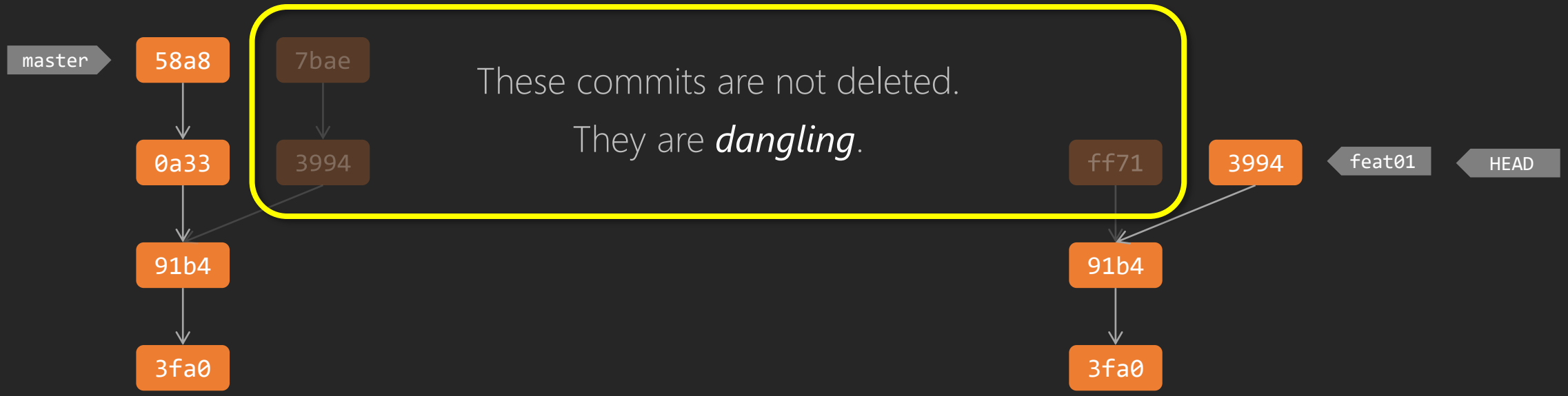
Abandoning commit from detached head



Abandoning commit from detached head

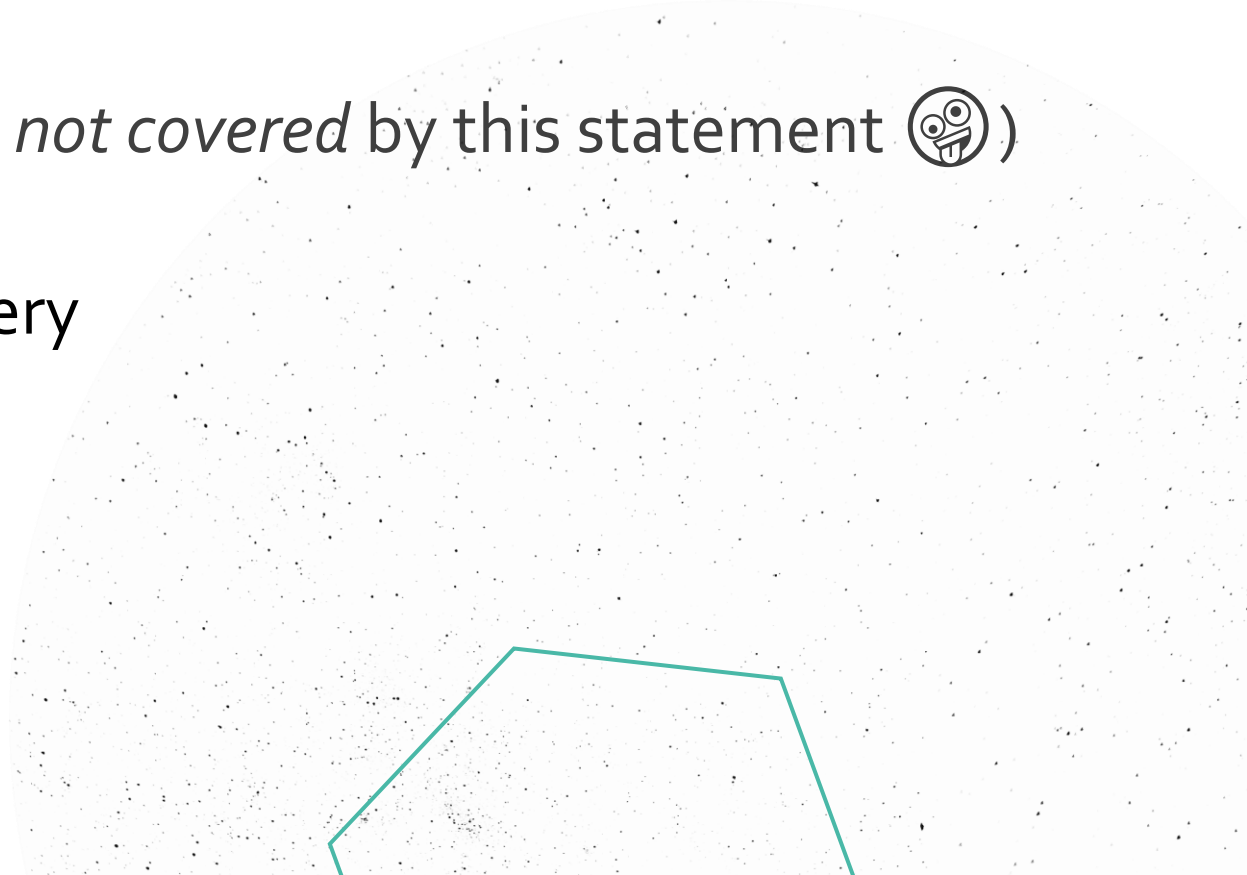


Dangling commits



Recovering commits

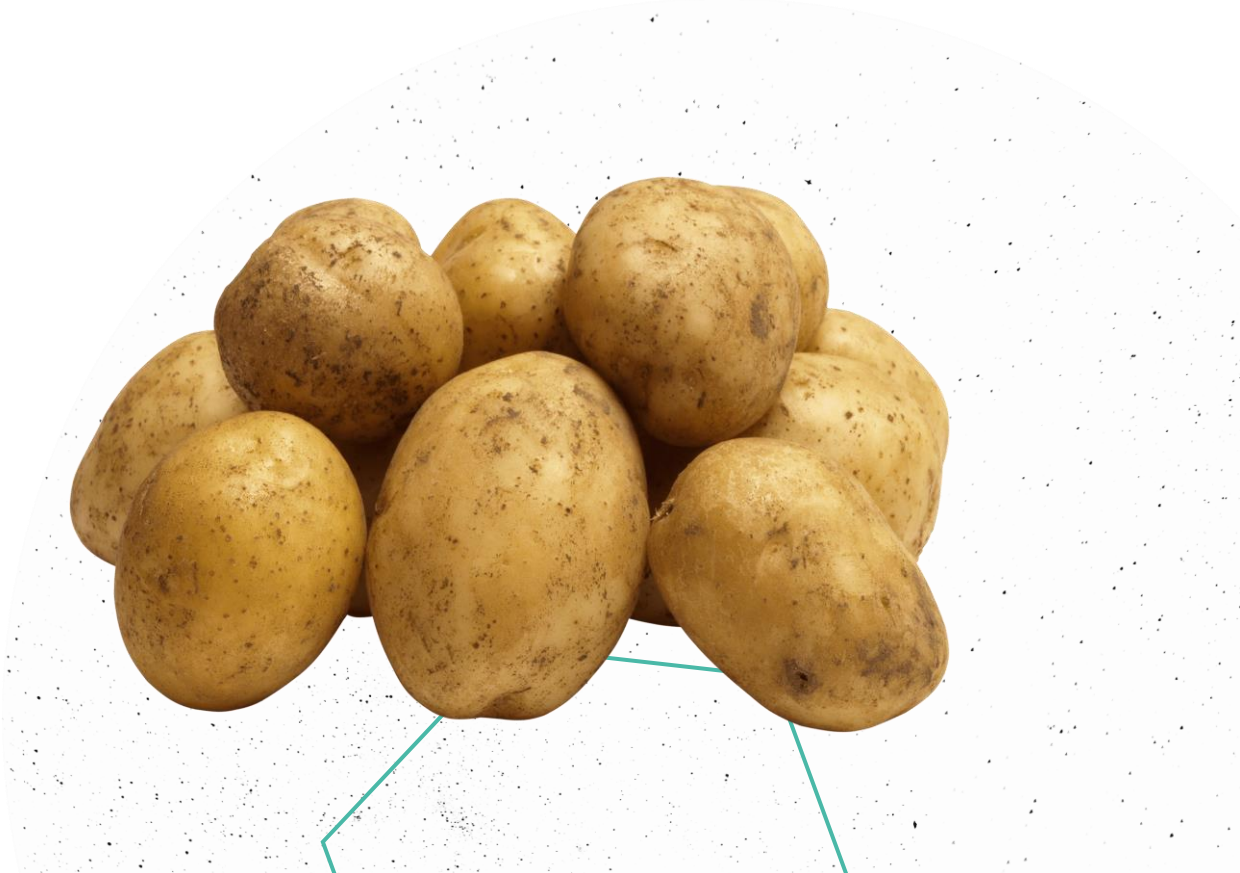
- Git **never loses** anything
  (at least not immediately 😅)
- You can **recover** from **all situations**
  (manual hacking through `.git` folder *not covered* by this statement 🤪)
- Git gives ample grace period for recovery



Reference log

- ***Not the same thing as log***

- Keeps track of ***changes*** to ***references***:
  - HEAD changes
  - Tips of branch changes



Recovering lost commits

BC/NAV TECH DAYS 2022
mibuso.com

- Step 1: Identify lost commits

```
git reflog
git fsck --lost-found
```

- Step 2: Create branch from (or detach head into) the commit

Garbage collection

- Reference log is kept for ***90 days*** (by default)
  - Any commit reachable from reference log ***is reachable***
- Dangling commits are kept for ***14 days*** (at least)
- Git runs garbage collection on:

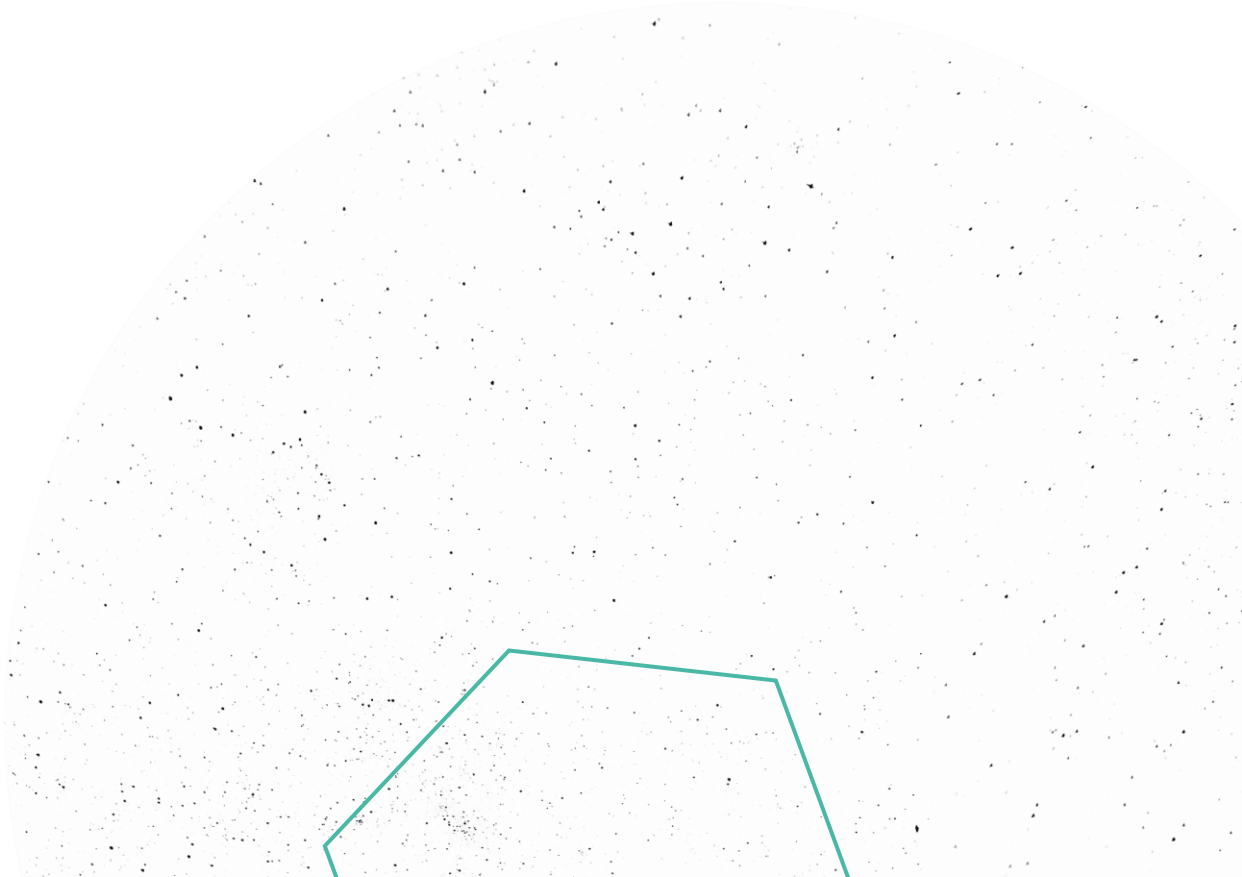
```
git fetch
git merge
git rebase
```

Rewriting history

- Git allows manipulating history

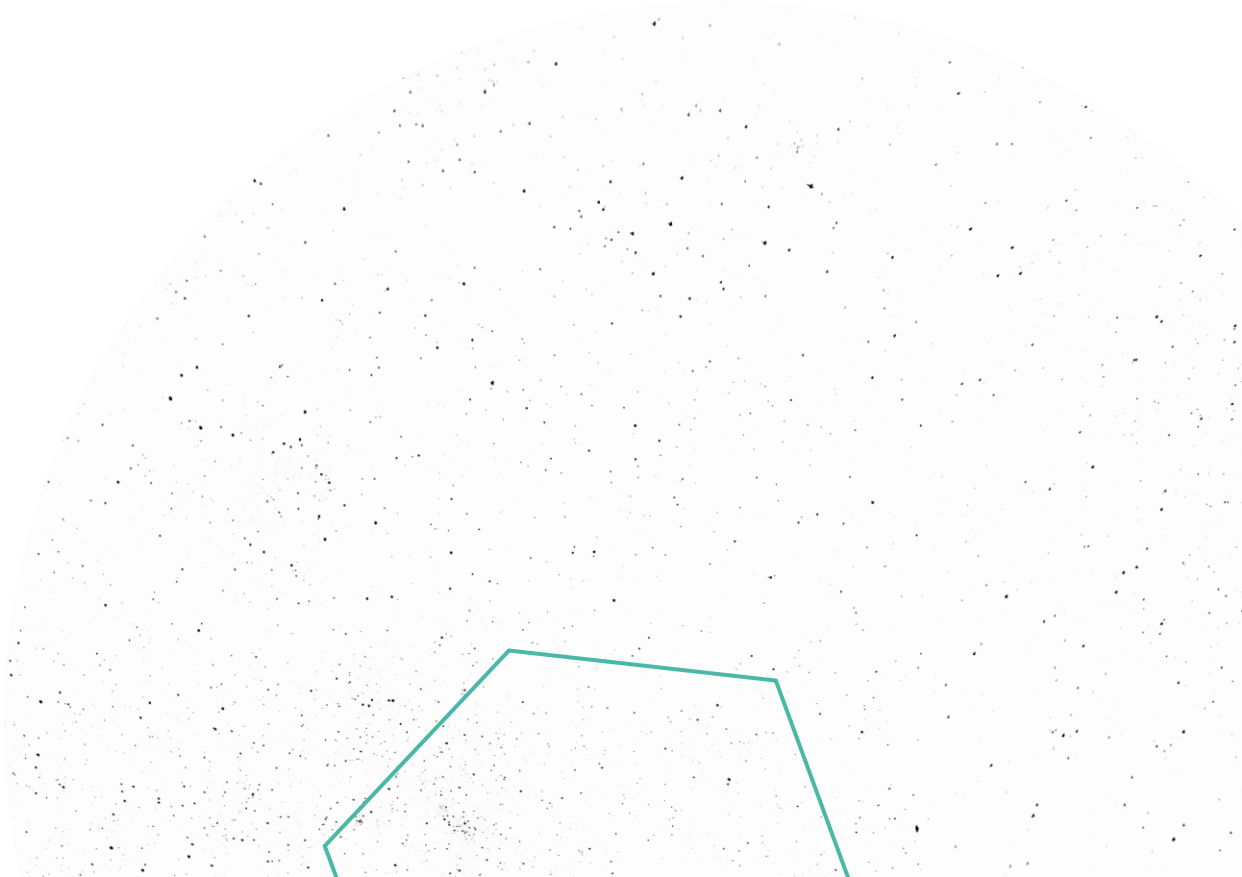
- Ways of manipulating history
  - Resetting
  - Amending
  - Interactive rebasing (rewriting)

- You cannot modify history
- You can only create alternate histories

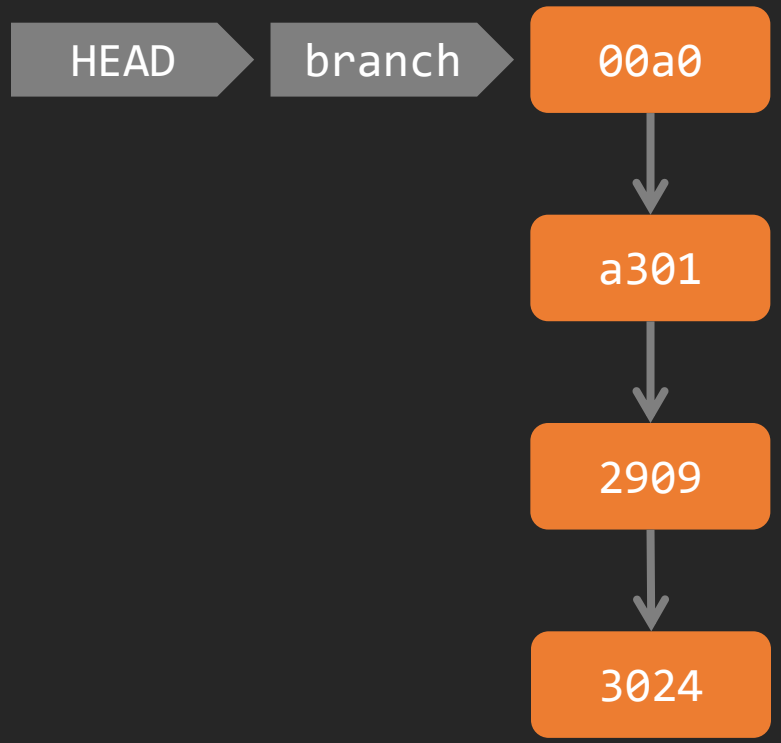


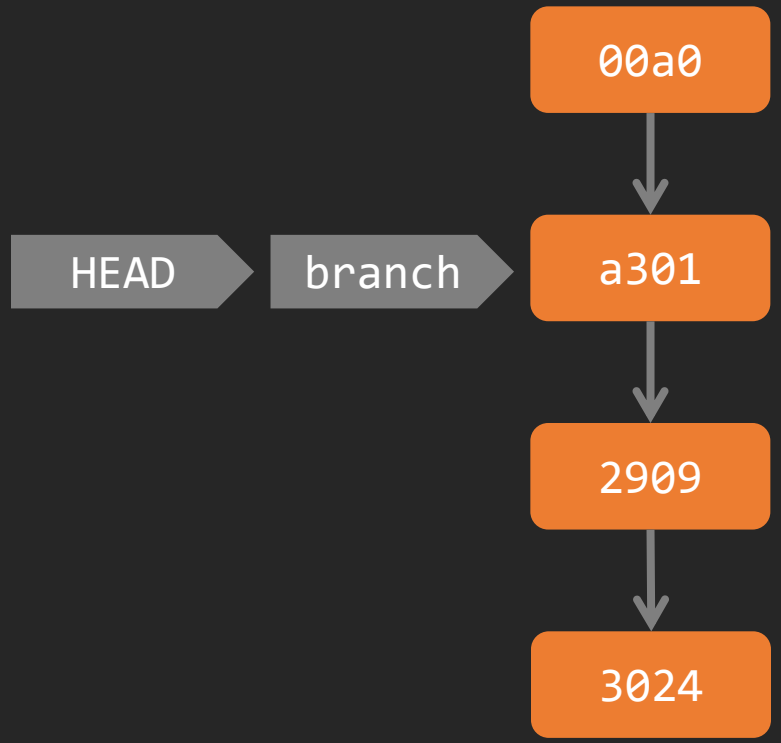
Resetting

- Think of it as sort of a `git undo`
- Moves HEAD pointer backwards
- Useful for:
  - Undoing commits
  - Correcting commits
  - Removing commits from history
  - Squashing

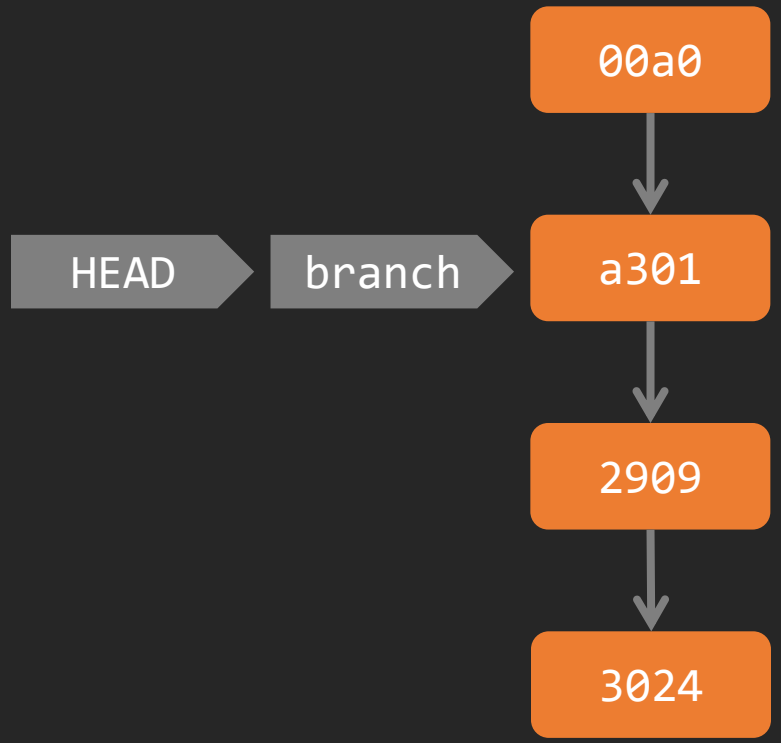


git reset

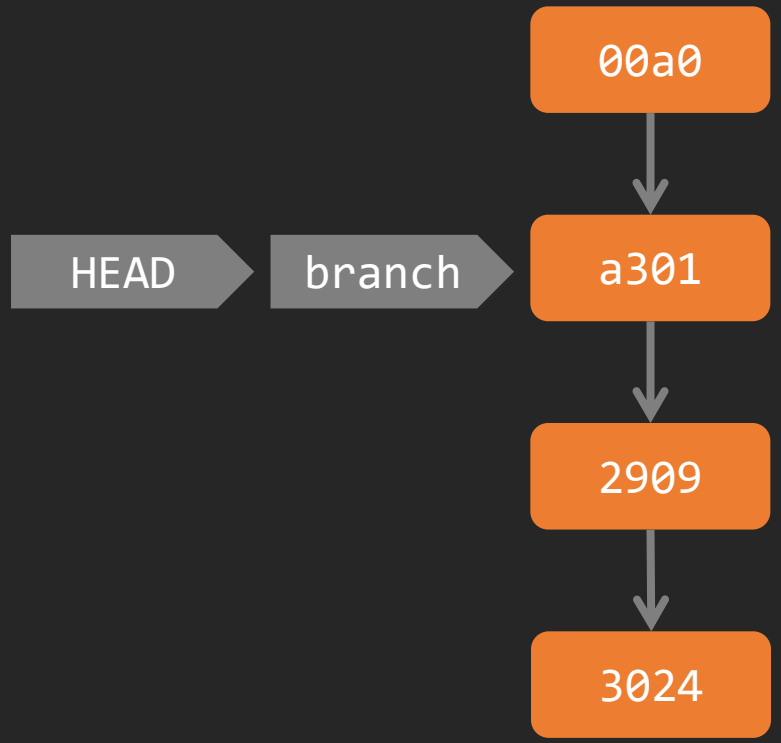

HEAD
branch
00a0
a301
2909
3024



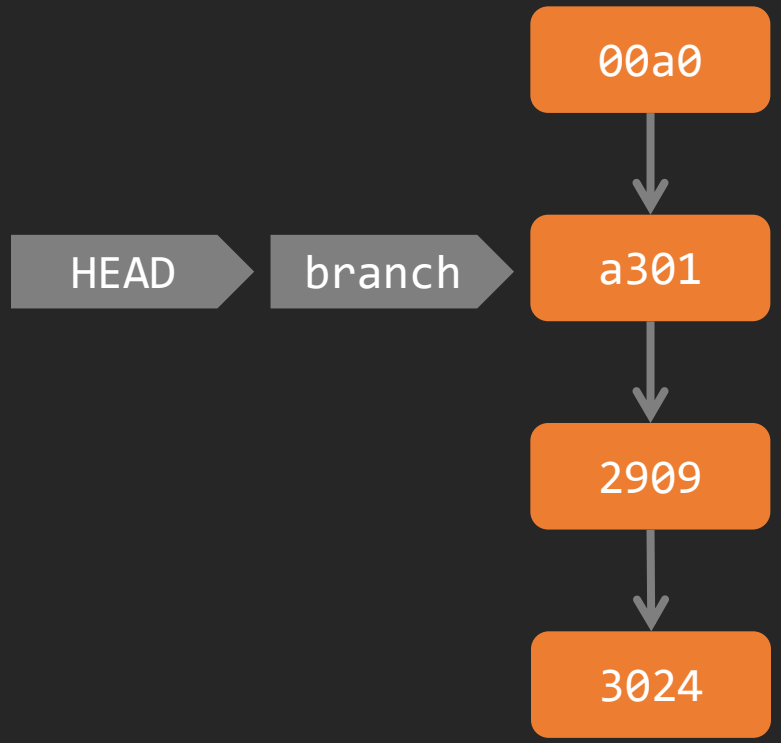

```
git reset a301
```



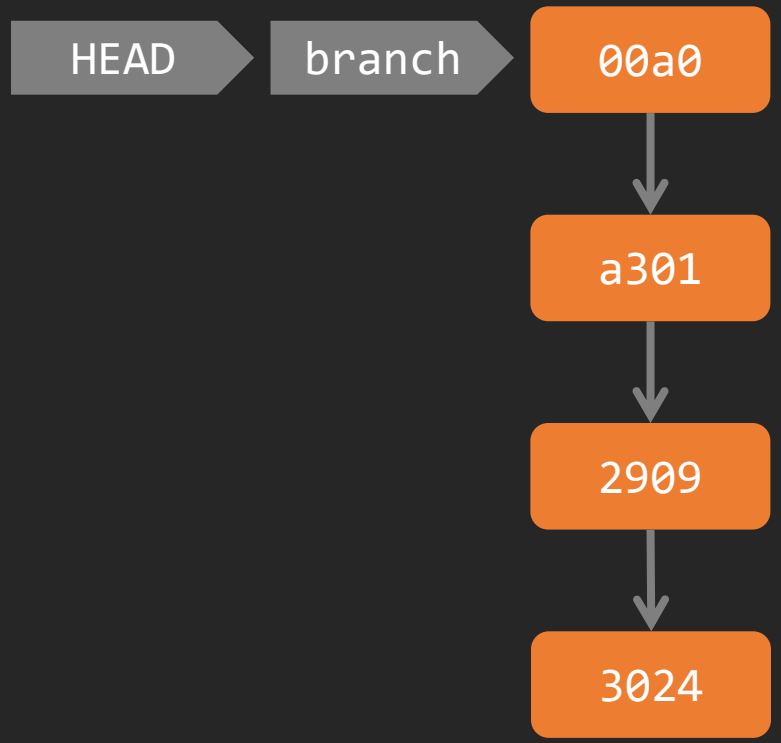

```
git reset HEAD~1
```

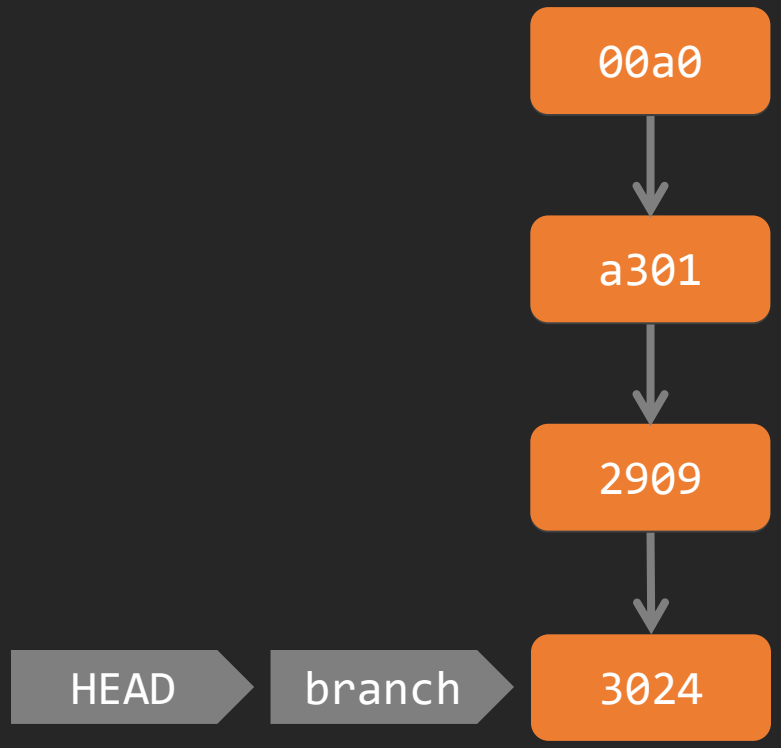



```
git reset HEAD~1
```

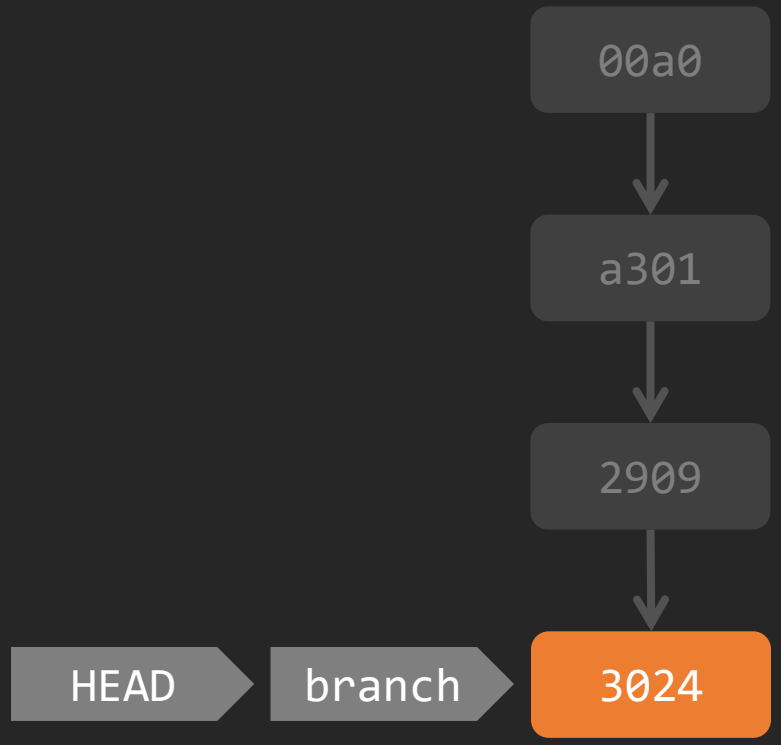



```
git reset HEAD~1
```


HEAD
branch
00a0
a301
2909
3024

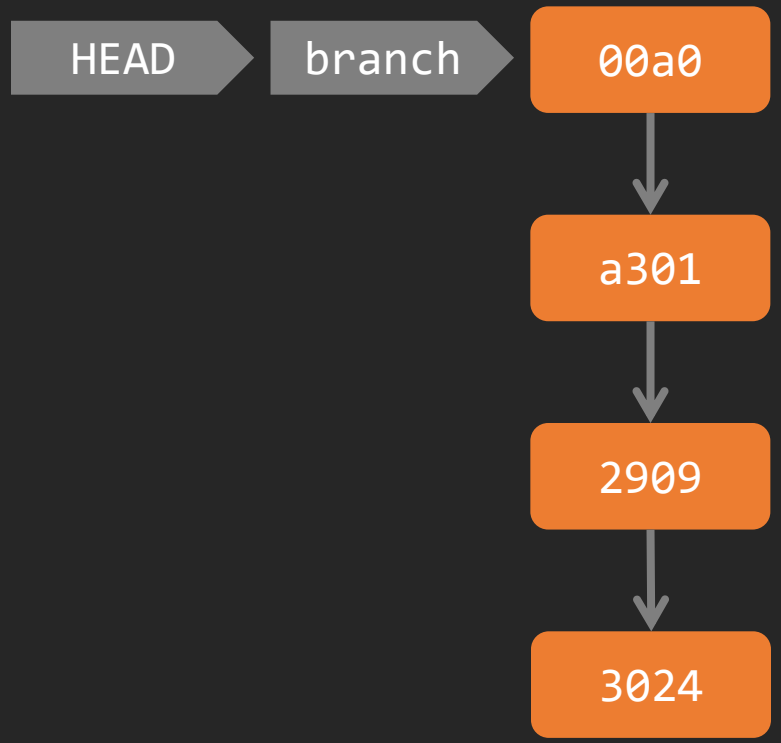


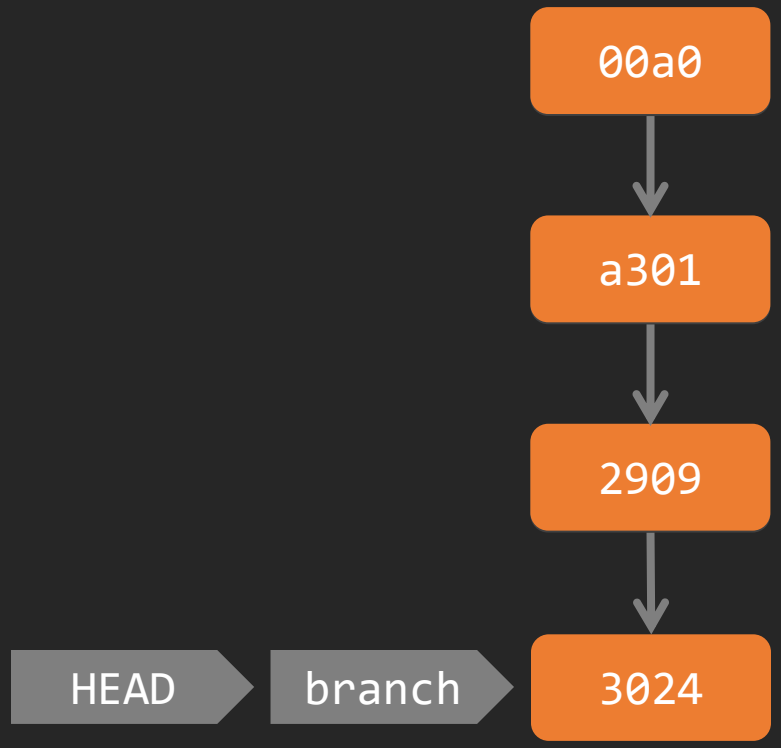

```
git reset HEAD~3
```



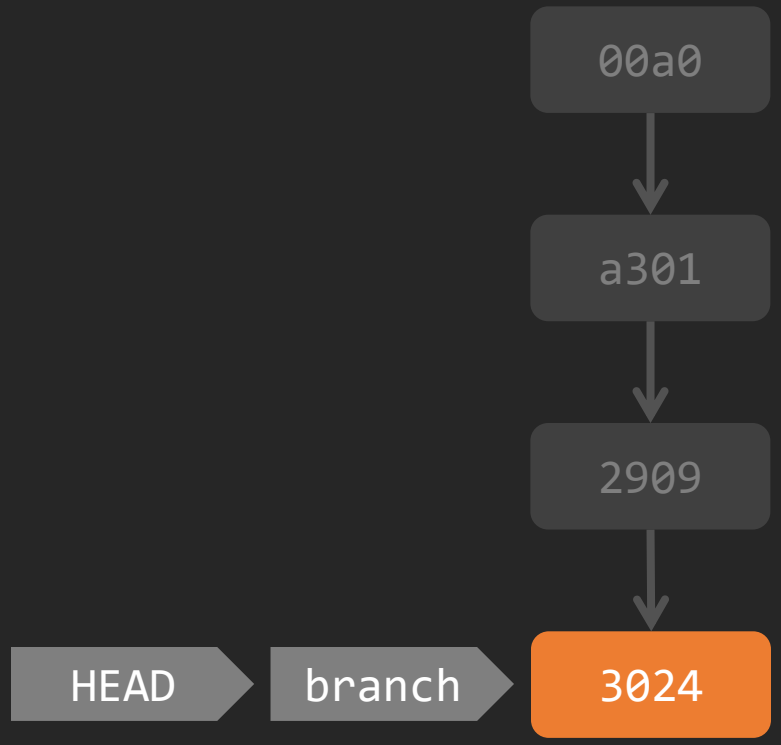

```
git reset HEAD~3
```

Working tree


HEAD
branch
00a0
a301
2909
3024

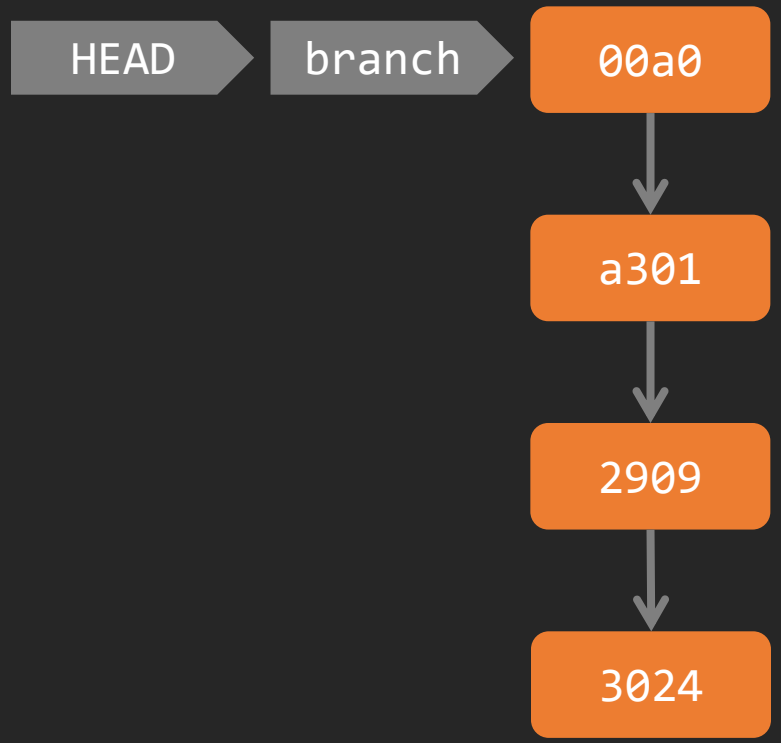


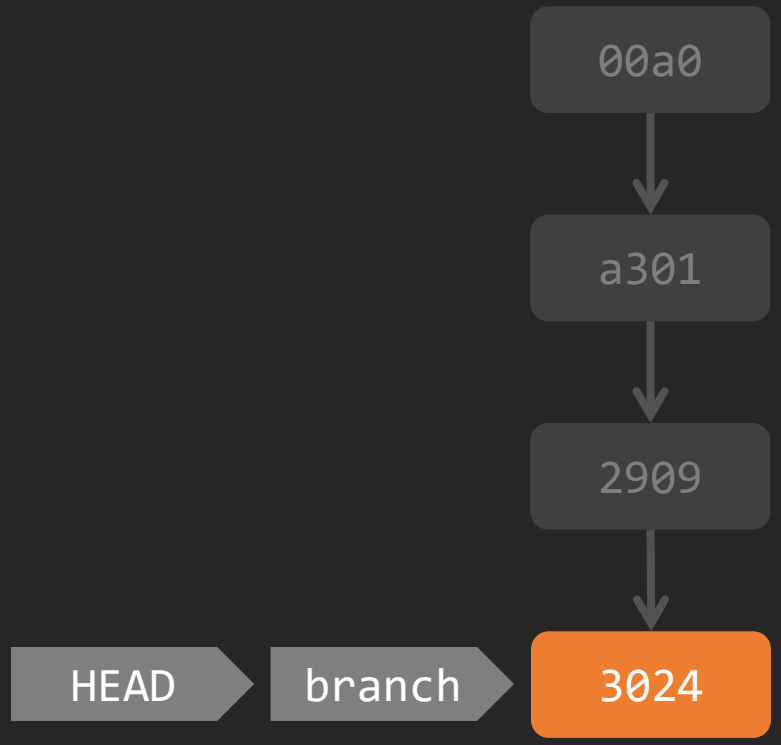

```
git reset HEAD~3 --soft
```




```
git reset HEAD~3 --soft
```

Staging area (index)

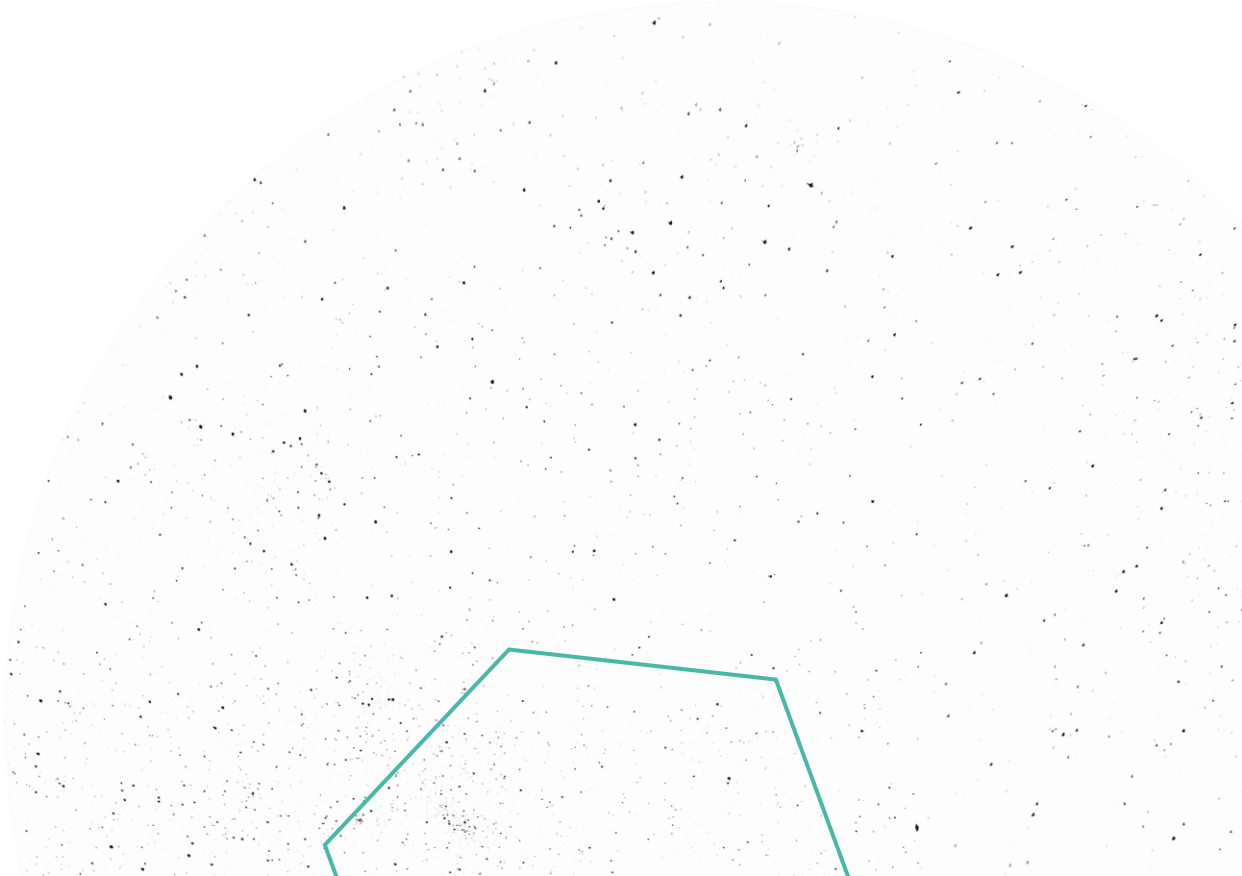

HEAD
branch
00a0
a301
2909
3024




```
git reset HEAD~3 --hard
```

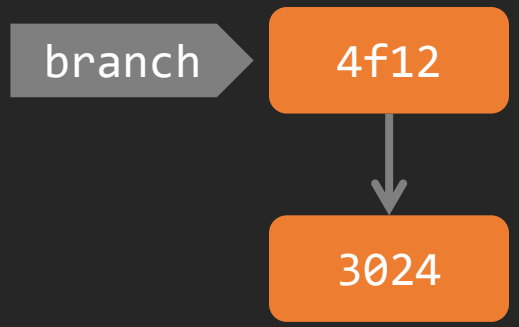
Amending

- Directly modifies last commit
- It can:
  - Add, remove, change contents of
  - Modify commit message
  - Modify some properties (e.g. author)
- Never actually changes last commit!
- Always creates new commit

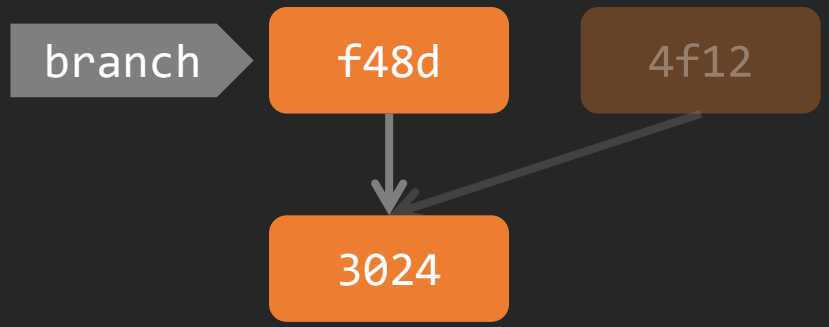


Amending

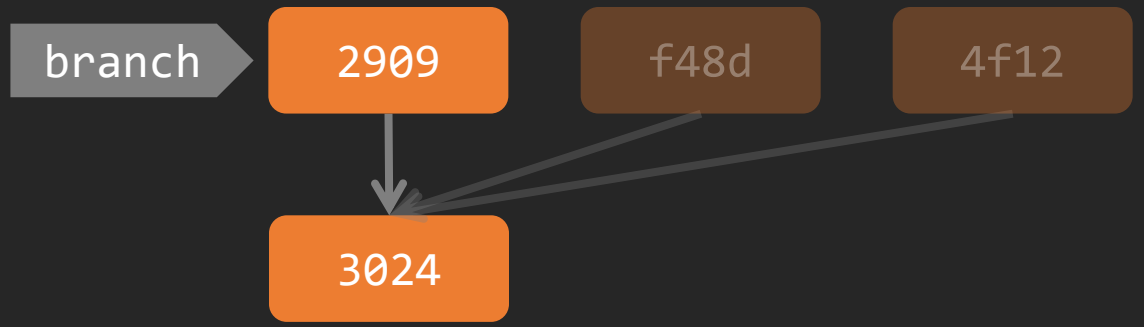
Amending



Amending



Amending

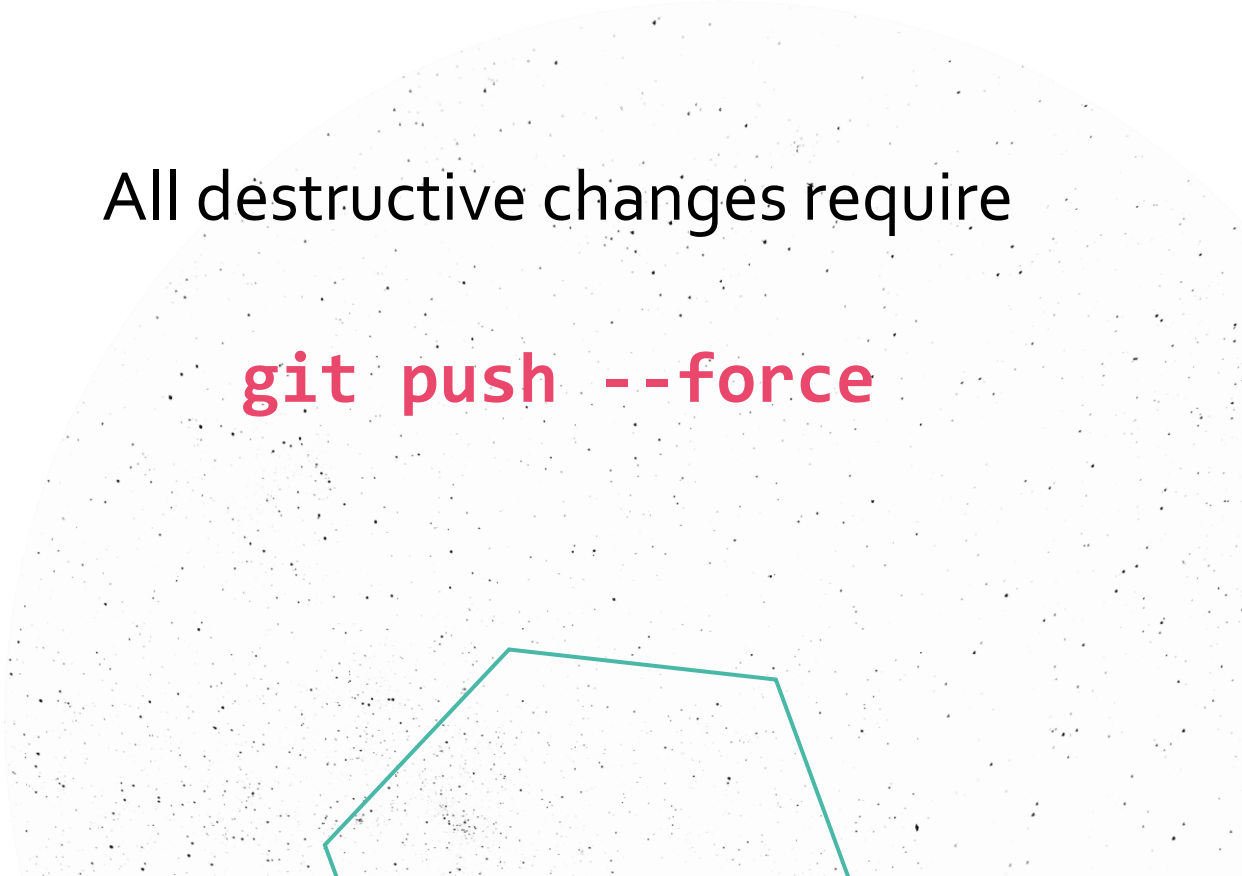


Important to know!

- **Only safe before** `git push`
- You cannot (just) push alternate histories
- Any destructive changes to histories of pushed branches are always rejected by remotes

All destructive changes require

`git push --force`



Important to know!

All destructive changes require

```
git push --force
```

A few revision control system takeaways

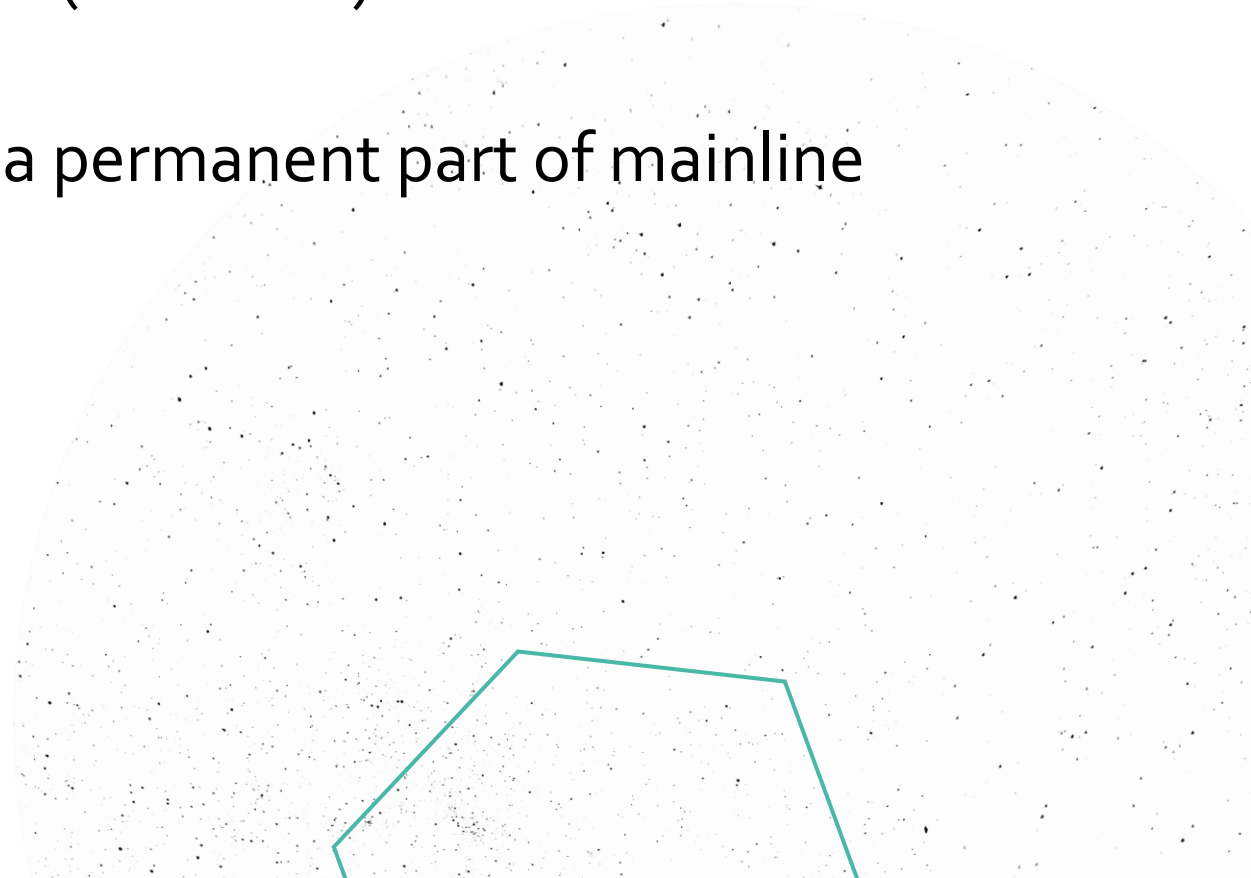
BC/NAV TECH DAYS 2022
mibuso.com

- Branches are just **pointers**
- Git **never loses anything**
- Lost commits **can be recovered**, often for 90 days
- History can be **freely rewritten** before pushing

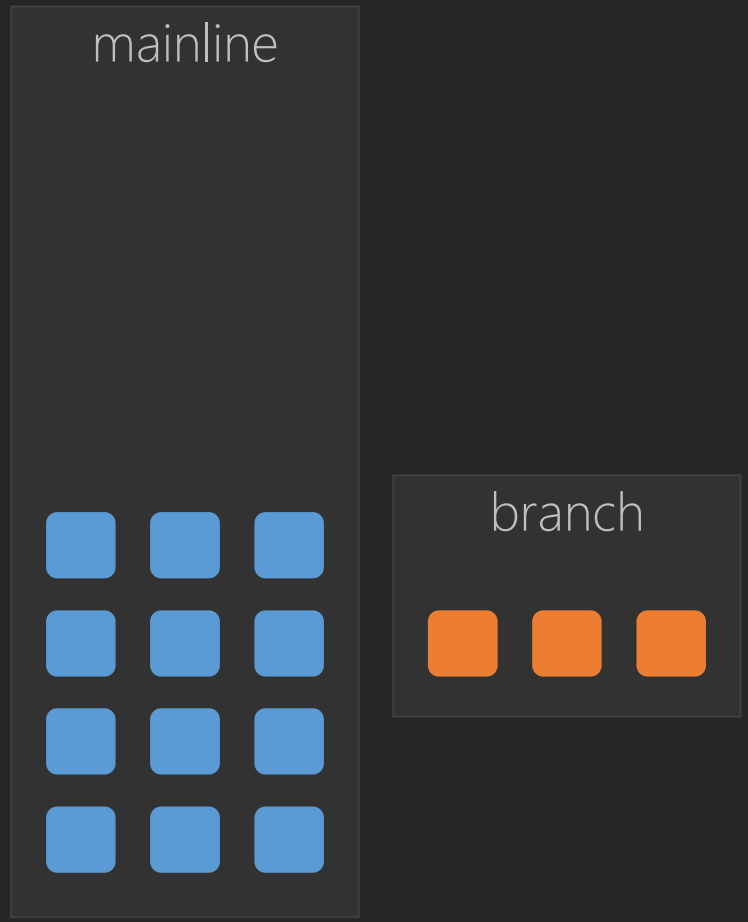
Integrating Changes

Integrating changes

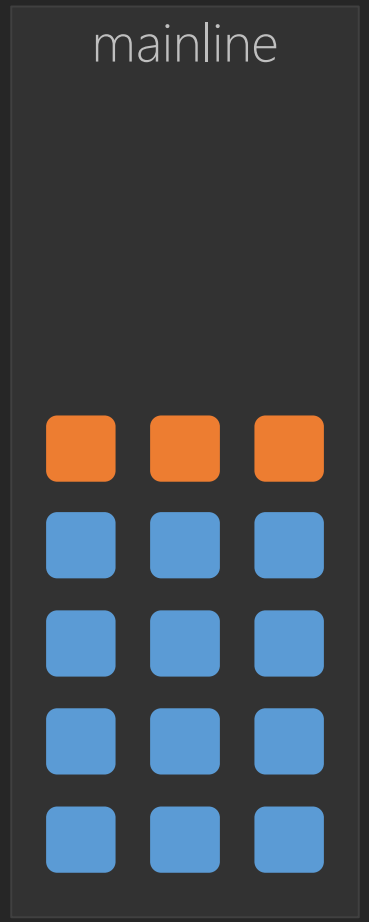
- Colloquially also known as ***"merging"***
- Process of combining separate histories (branches)
- Typically making contents of a branch a permanent part of mainline



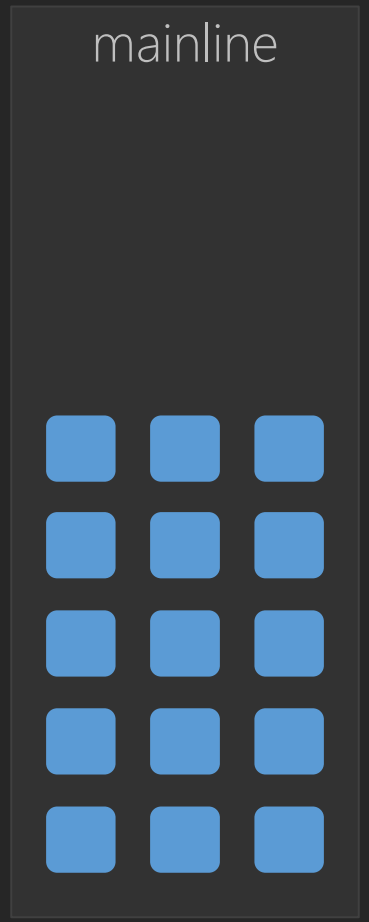
Integrating changes



Integrating changes



Integrating changes

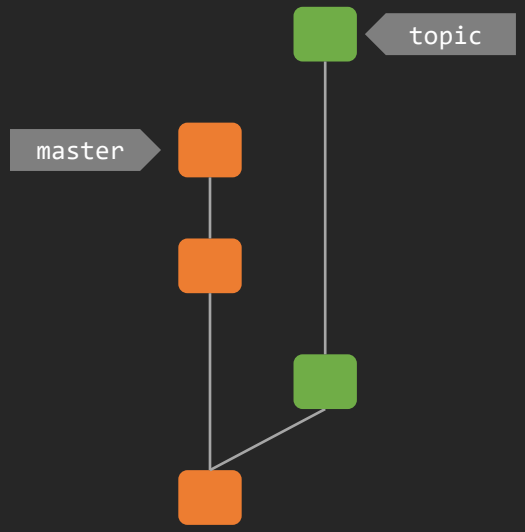


Integrating changes

Merge

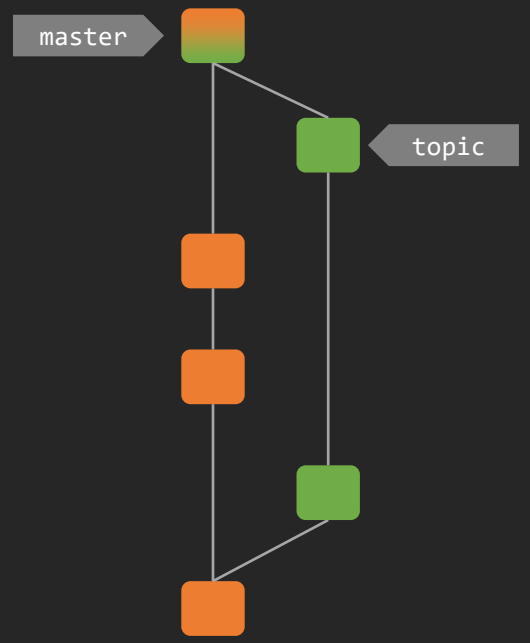
Integrating changes

Merge



Integrating changes

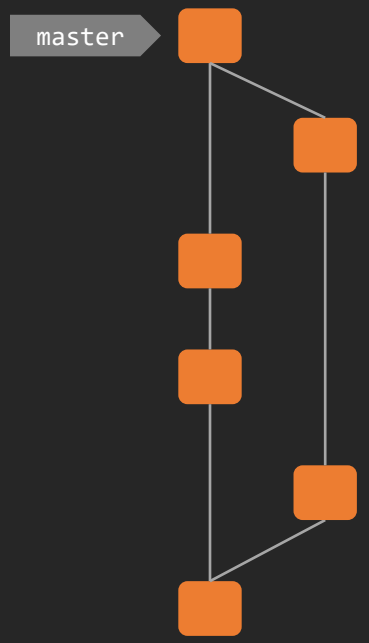
Merge




```
git checkout master
git merge topic
```

Integrating changes

Merge




```
git checkout master
git merge topic

git branch –d topic
```

Integrating changes

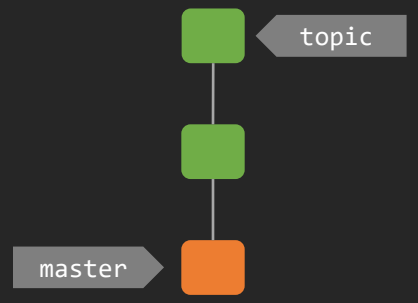
Merge

Fast-forward

Integrating changes

Merge

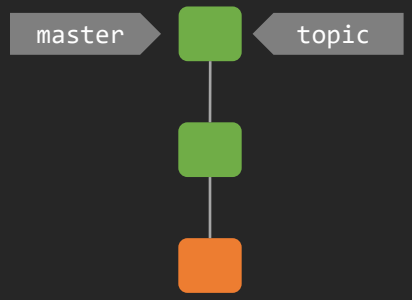
Fast-forward



Integrating changes

Merge

Fast-forward

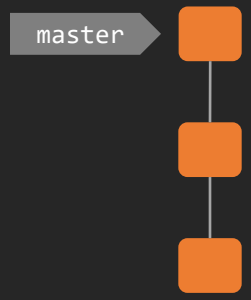



```
git checkout master
git merge topic
```

Integrating changes

Merge

Fast-forward



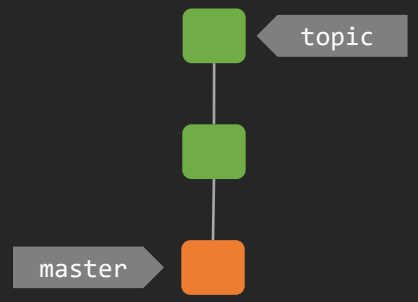

```
git checkout master
git merge topic

git branch –d topic
```

Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

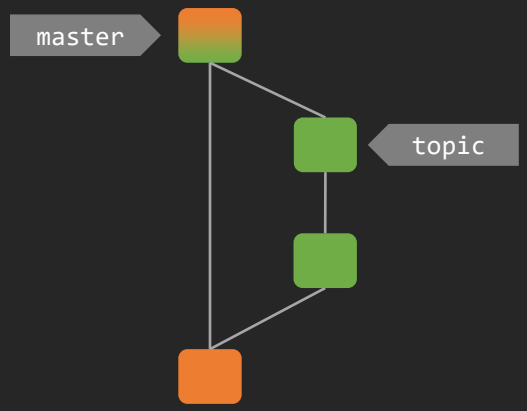


Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy



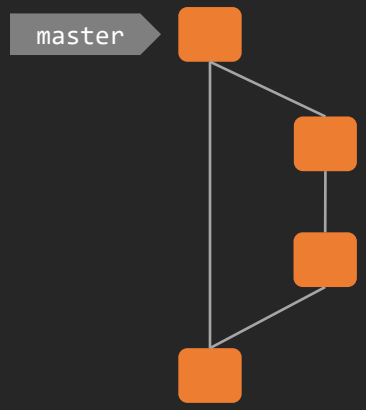

```
git checkout master
git merge topic --no-ff
```

Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy




```
git checkout master
git merge topic --no-ff

git branch –d topic
```

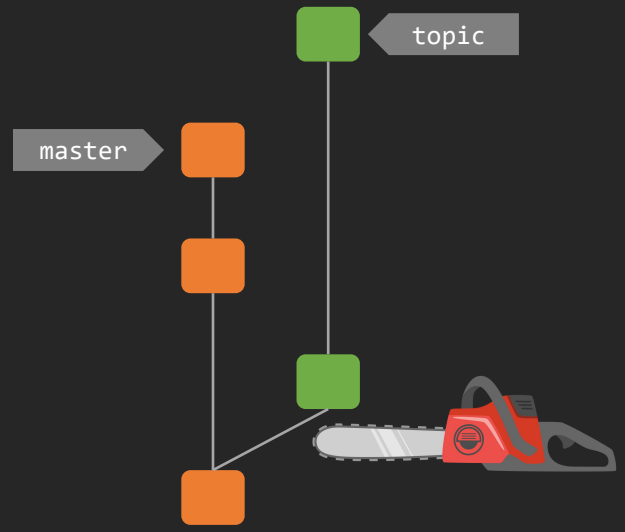
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase



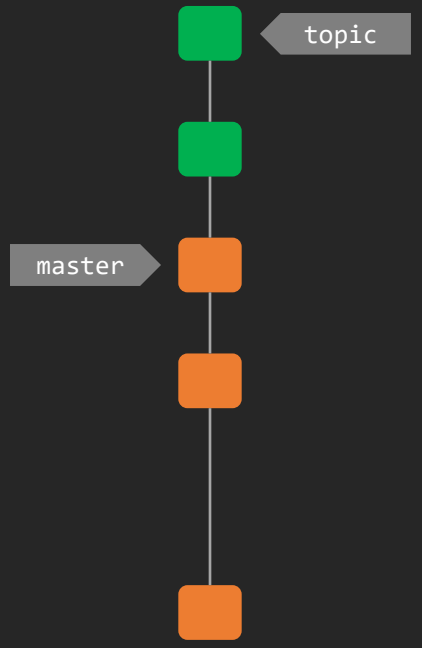
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase




```
git checkout topic
git rebase master
```

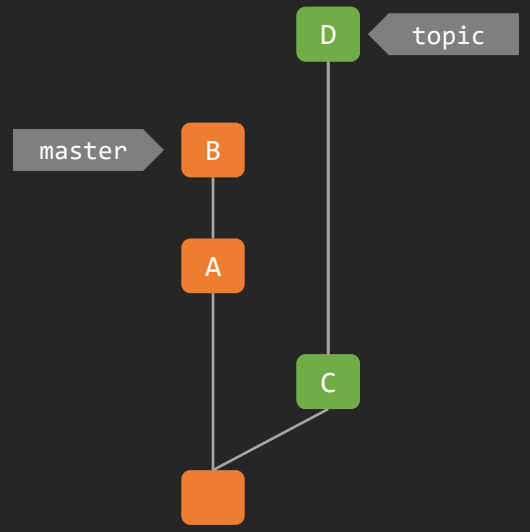
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase



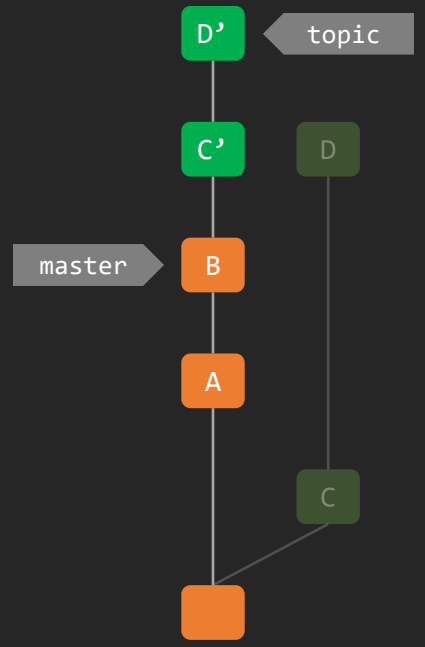
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase




```
git checkout topic
git rebase master
```

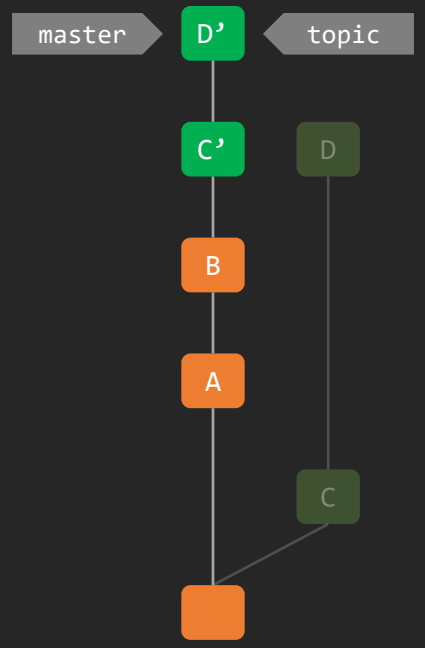
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase




```
git checkout topic
git rebase master

git checkout master
git merge topic
```

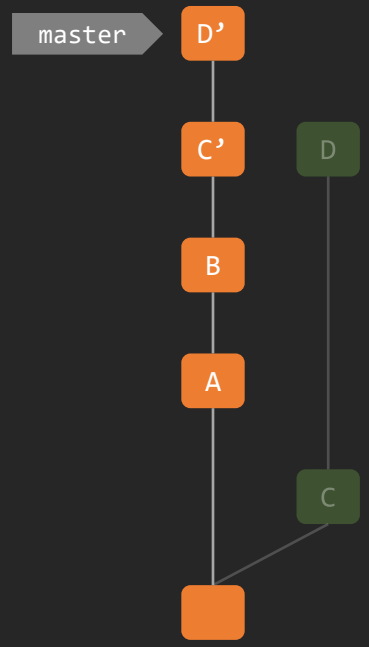
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase




```
git checkout topic
git rebase master

git checkout master
git merge topic

git branch –d topic
```

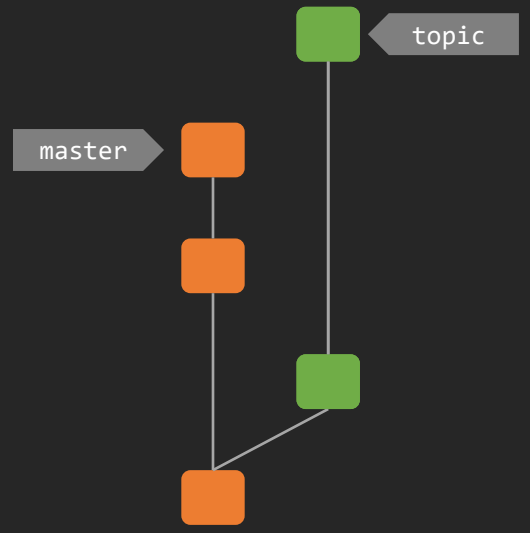
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase



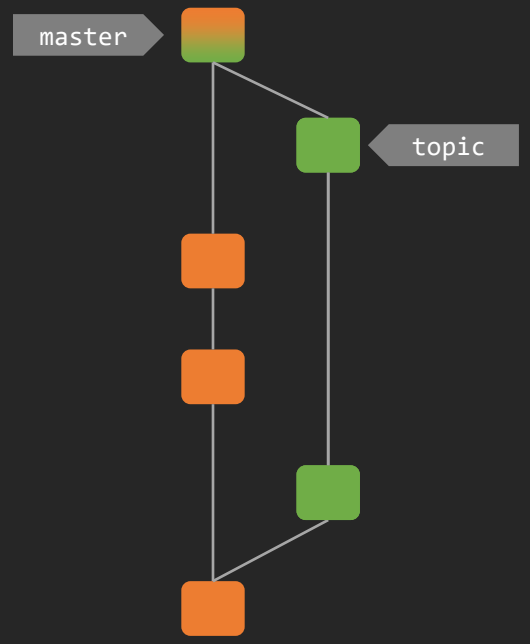
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase




```
git checkout master
git merge topic
```

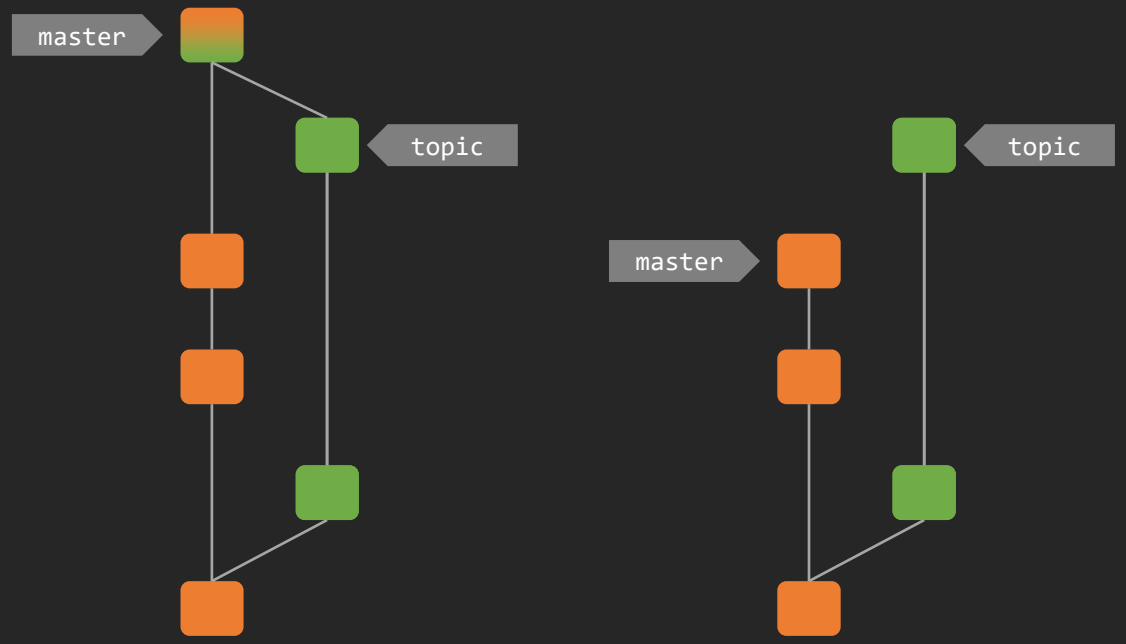
Integrating changes

Merge

Fast-forward

Merge
no fast-forward
policy

Rebase



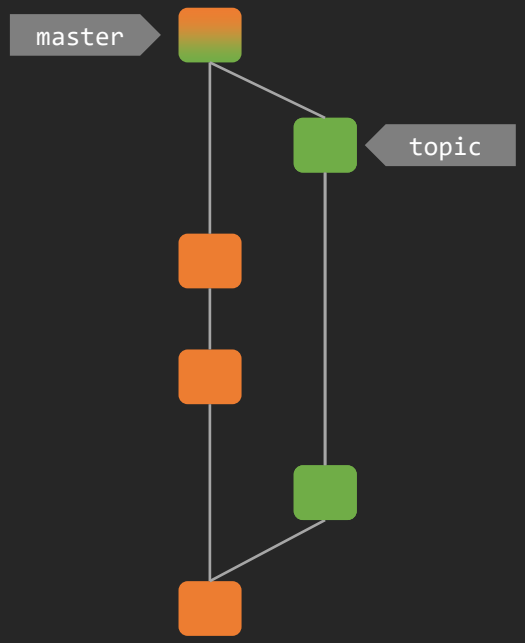
Integrating changes

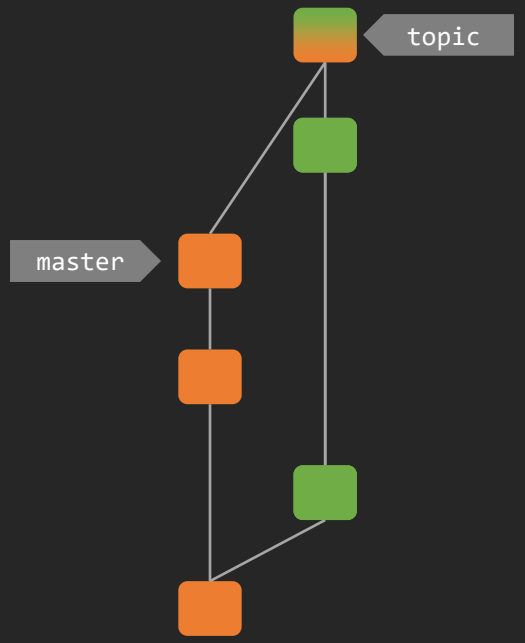
Merge

Fast-forward

Merge
no fast-forward
policy

Rebase






```
git checkout topic
git merge master
```

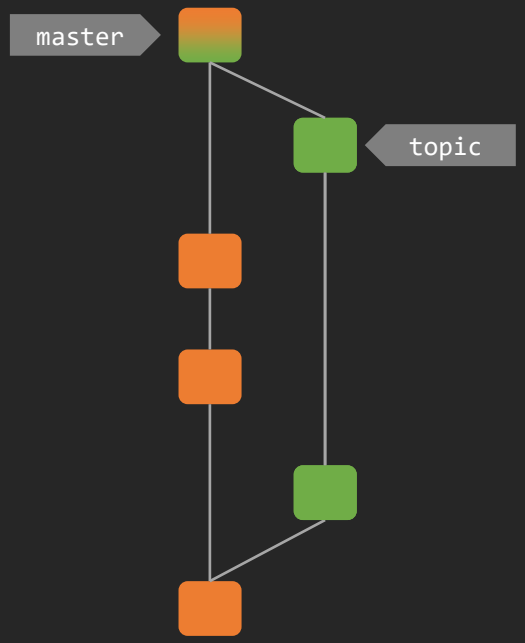
Integrating changes

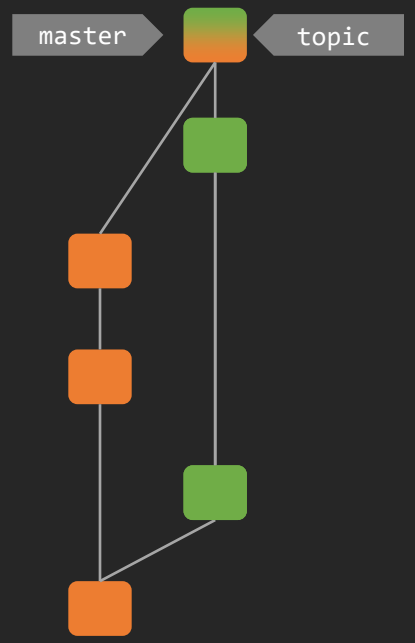
Merge

Fast-forward

Merge
no fast-forward
policy

Rebase






```
git checkout topic
git merge master

git checkout master
git merge topic
```

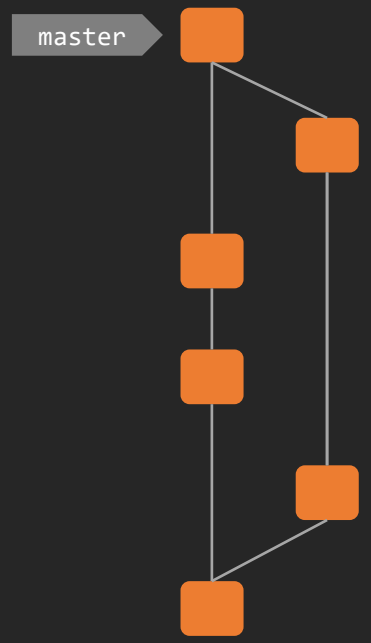
Integrating changes

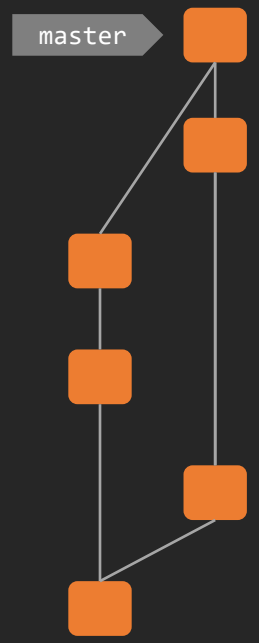
Merge

Fast-forward

Merge
no fast-forward
policy

Rebase





Integrating changes

Merge

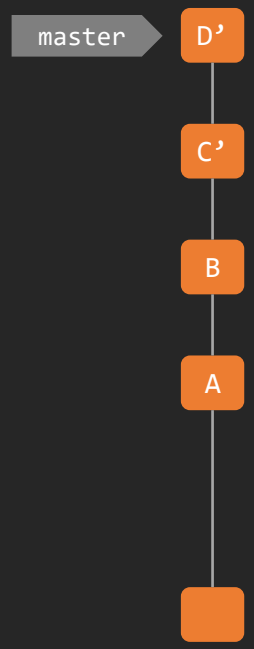
Fast-forward

Merge
no fast-forward
policy

Rebase

```
git checkout topic
git rebase master
```

```
git checkout master
git merge topic
```



Integrating changes

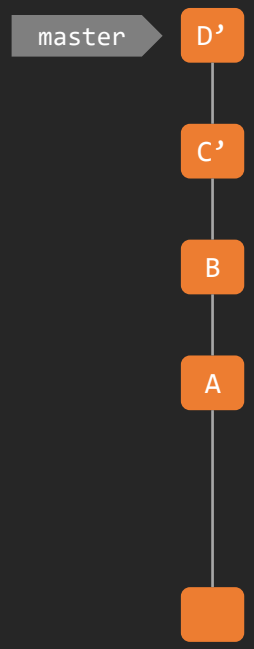
Merge

Fast-forward

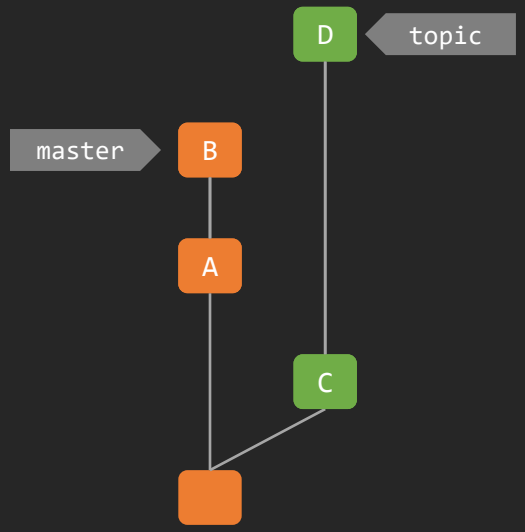
Merge
no fast-forward
policy

Rebase

```
git checkout topic
git rebase master
```




```
git checkout master
git merge topic
```



Integrating changes

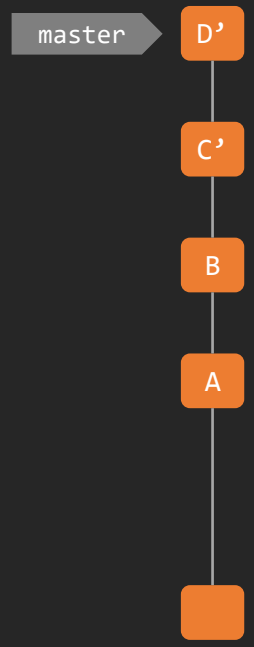
Merge

Fast-forward

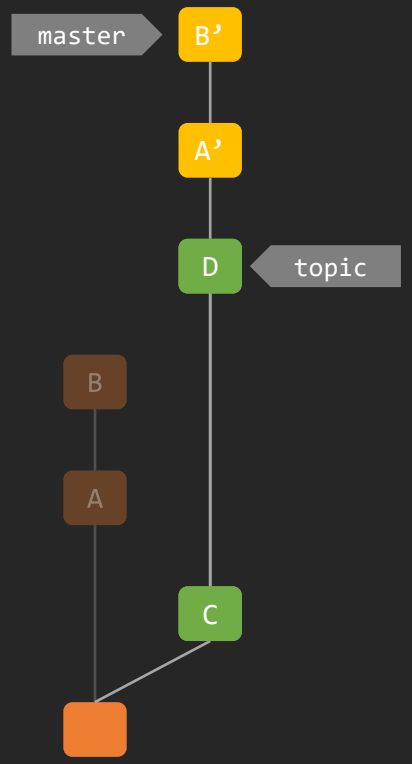
Merge
no fast-forward
policy

Rebase

```
git checkout topic
git rebase master
```




```
git checkout master
git merge topic
```




```
git checkout master
git rebase topic
```

Integrating changes

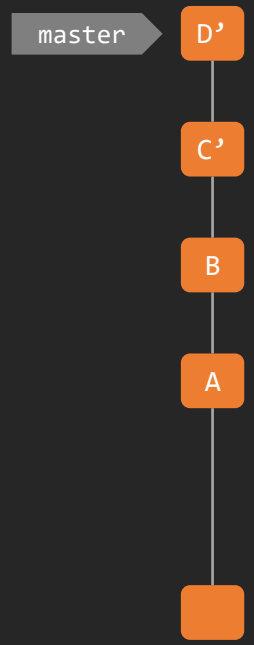
Merge

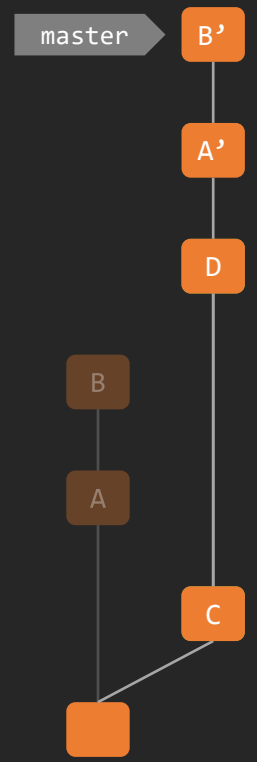
Fast-forward

Merge
no fast-forward
policy

Rebase

```
git checkout topic
git rebase master

git checkout master
git merge topic
```






```
git checkout master
git rebase topic

git branch -d topic
```

Integrating changes

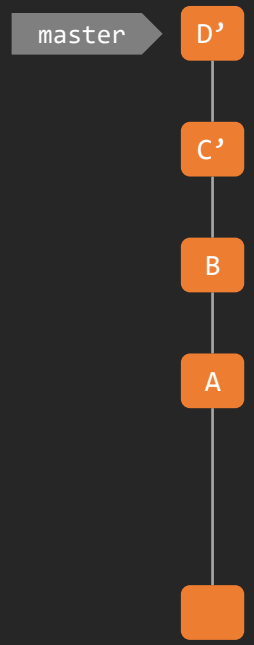
Merge

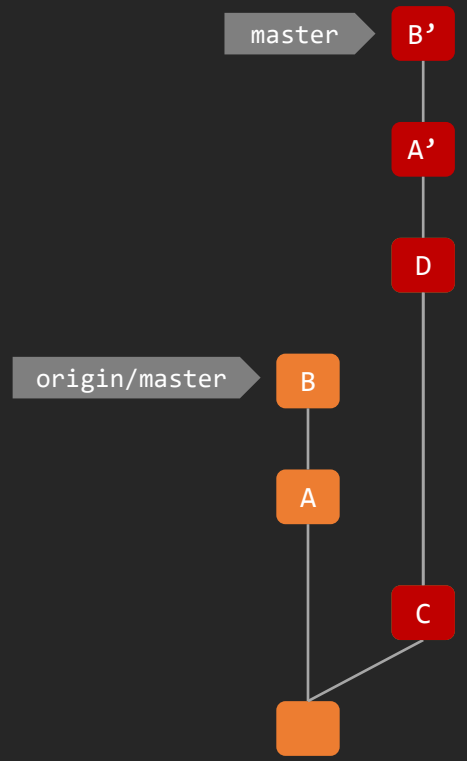
Fast-forward

Merge
no fast-forward
policy

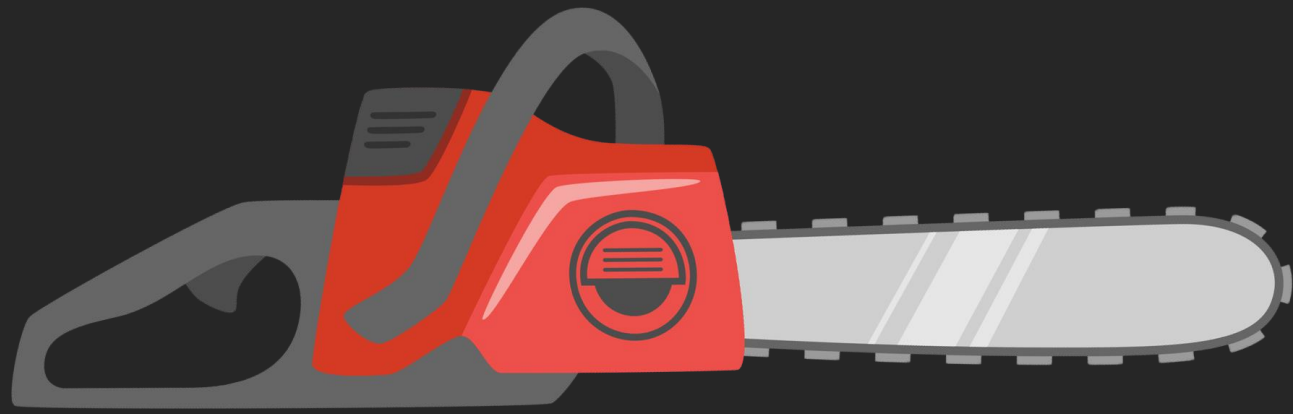
Rebase

```
git checkout topic
git rebase master

git checkout master
git merge topic
```






```
git checkout master
git rebase topic

git branch -d topic
```



BC/NAV TECH DAYS 2022
mibuso.com

Integrating changes

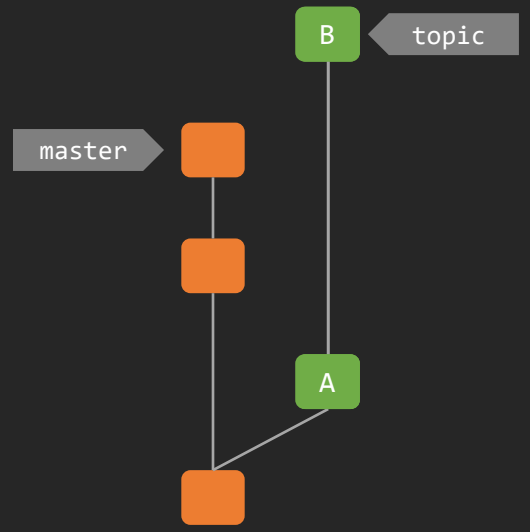
Merge

Fast-forward

Merge
no fast-forward
policy

Rebase

"Squash"



Integrating changes

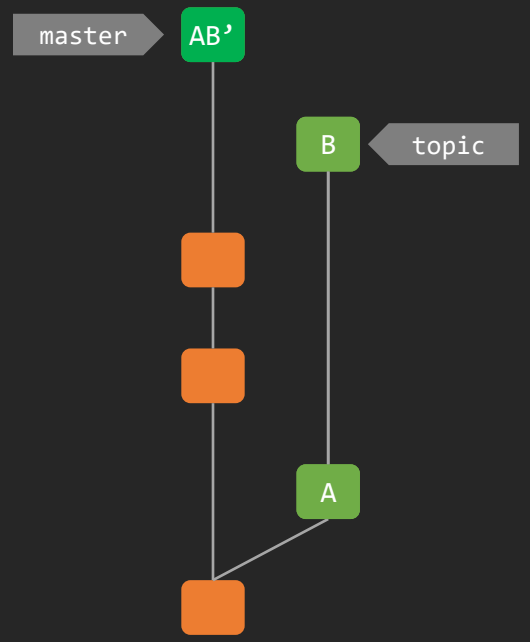
Merge

Fast-forward

Merge
no fast-forward
policy

Rebase

"Squash"




```
git checkout master
git merge --squash topic
```

Integrating changes

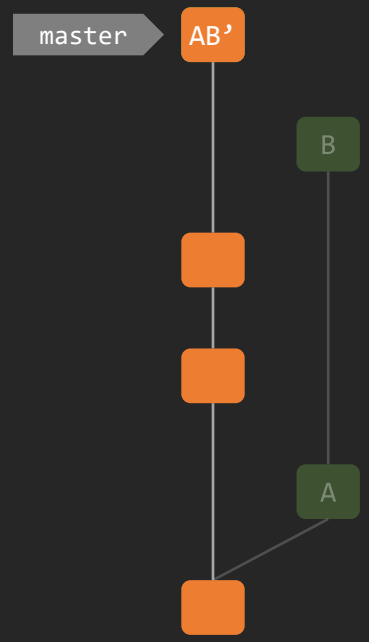
Merge

Fast-forward

Merge
no fast-forward
policy

Rebase

"Squash"




```
git checkout master
git merge --squash topic

git branch –D topic
```

Integrating changes

Merge

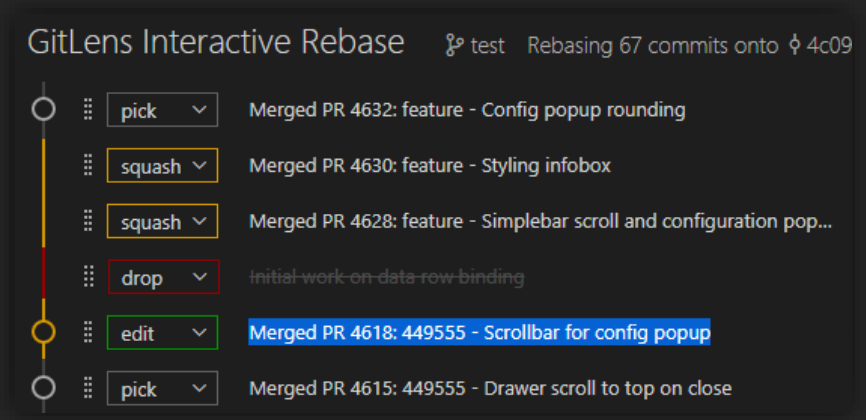
Fast-forward

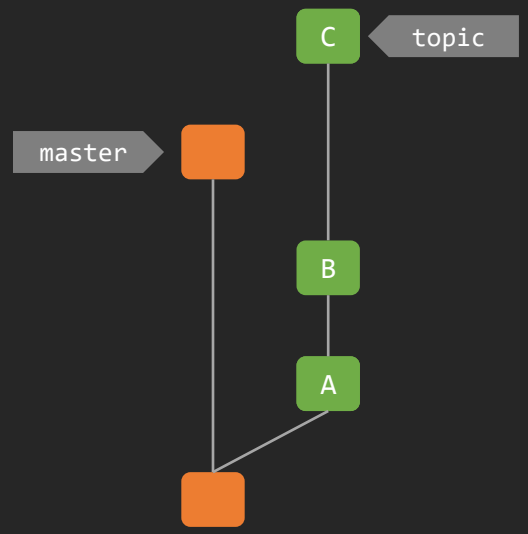
Merge
no fast-forward
policy

Rebase

"Squash"

Interactive
Rebase






```
git checkout topic
git rebase --interactive master
```

Integrating changes

Merge

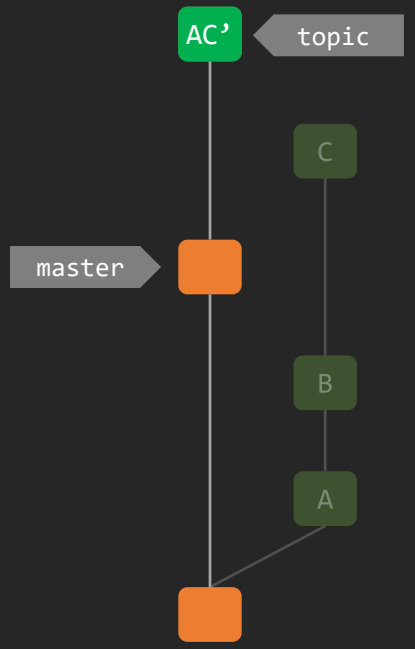
Fast-forward

Merge
no fast-forward
policy

Rebase

"Squash"

Interactive
Rebase




```
git checkout topic
git rebase --interactive master

git rebase --continue
```

Integrating changes

Merge

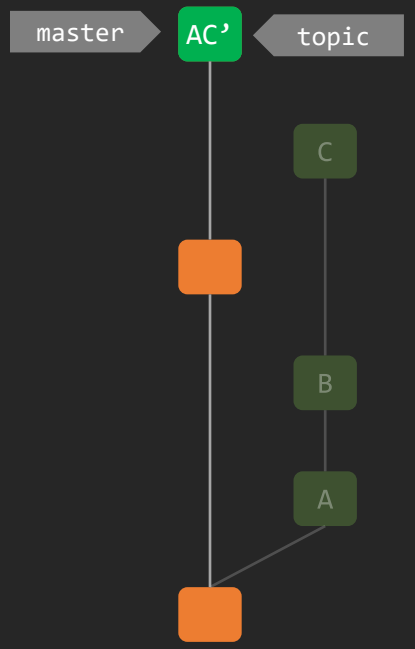
Fast-forward

Merge
no fast-forward
policy

Rebase

"Squash"

Interactive
Rebase




```
git checkout topic
git rebase --interactive master

git rebase --continue

git checkout master
git merge topic
```

Integrating changes

Merge

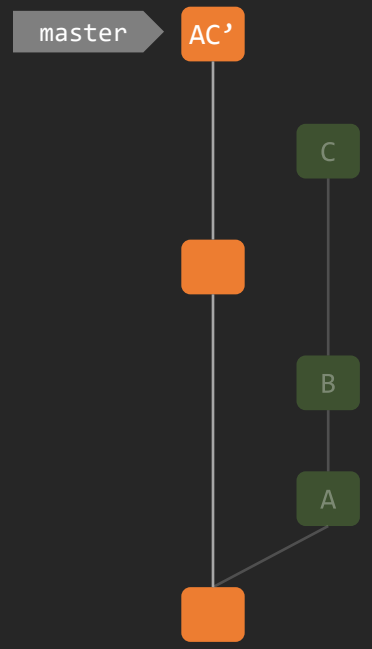
Fast-forward

Merge
no fast-forward
policy

Rebase

"Squash"

Interactive
Rebase




```
git checkout topic
git rebase --interactive master

git rebase --continue

git checkout master
git merge topic

git branch -d topic
```

Integrating changes

Merge

Fast-forward

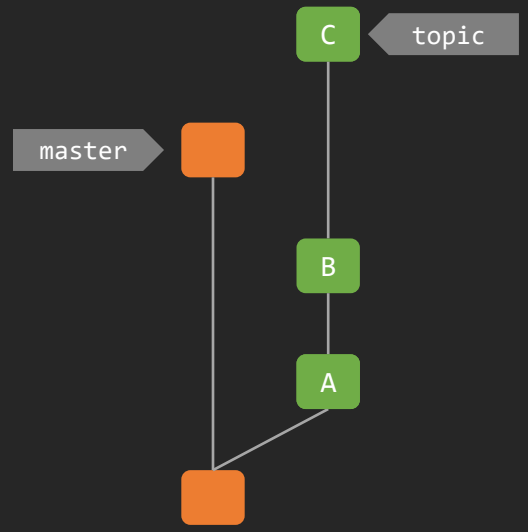
Merge
no fast-forward
policy

Rebase

"Squash"

Interactive
Rebase

Cherry-
picking



Integrating changes

Merge

Fast-forward

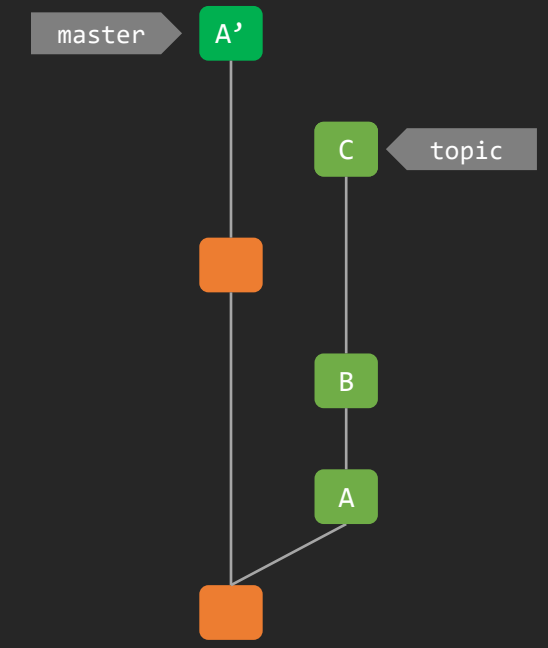
Merge
no fast-forward
policy

Rebase

"Squash"

Interactive
Rebase

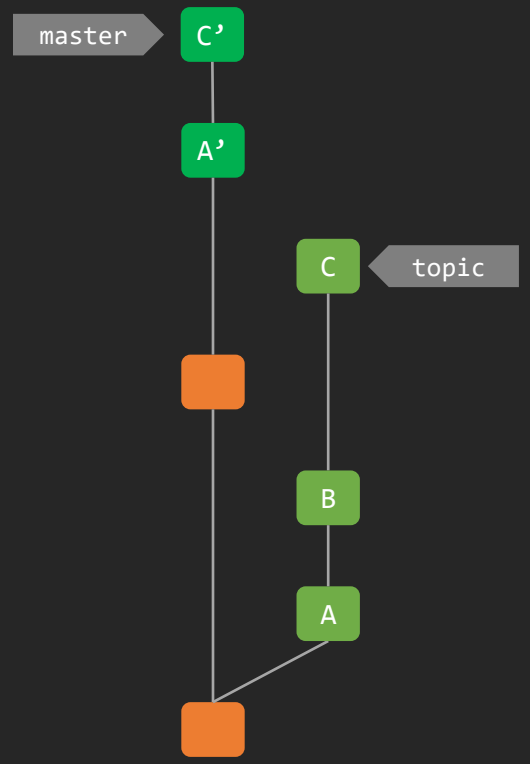
Cherry-
picking




```
git checkout master
git cherry-pick <sha1_a>
```

Integrating changes

Merge | Fast-forward | Merge no fast-forward policy | Rebase | "Squash" | Interactive Rebase | **Cherry-picking**




```
git checkout master
git cherry-pick <sha1_a>
git cherry-pick <sha1_c>
```

Any Questions?

BC/NAV TECH DAYS 2022
mibuso.com

Thank You!