

# An introduction to modern tools for collaborative science

Luca Heltai <[luca.heltai@sissa.it](mailto:luca.heltai@sissa.it)>

Mathematical Analysis, Modeling, and Applications ([math.sissa.it](http://math.sissa.it))  
Theoretical and Scientific Data Science ([datascience.sissa.it](http://datascience.sissa.it))



# Target Audience

- PhD Students
- Postdocs
- Researchers
- Anyone who needs order and control... :)

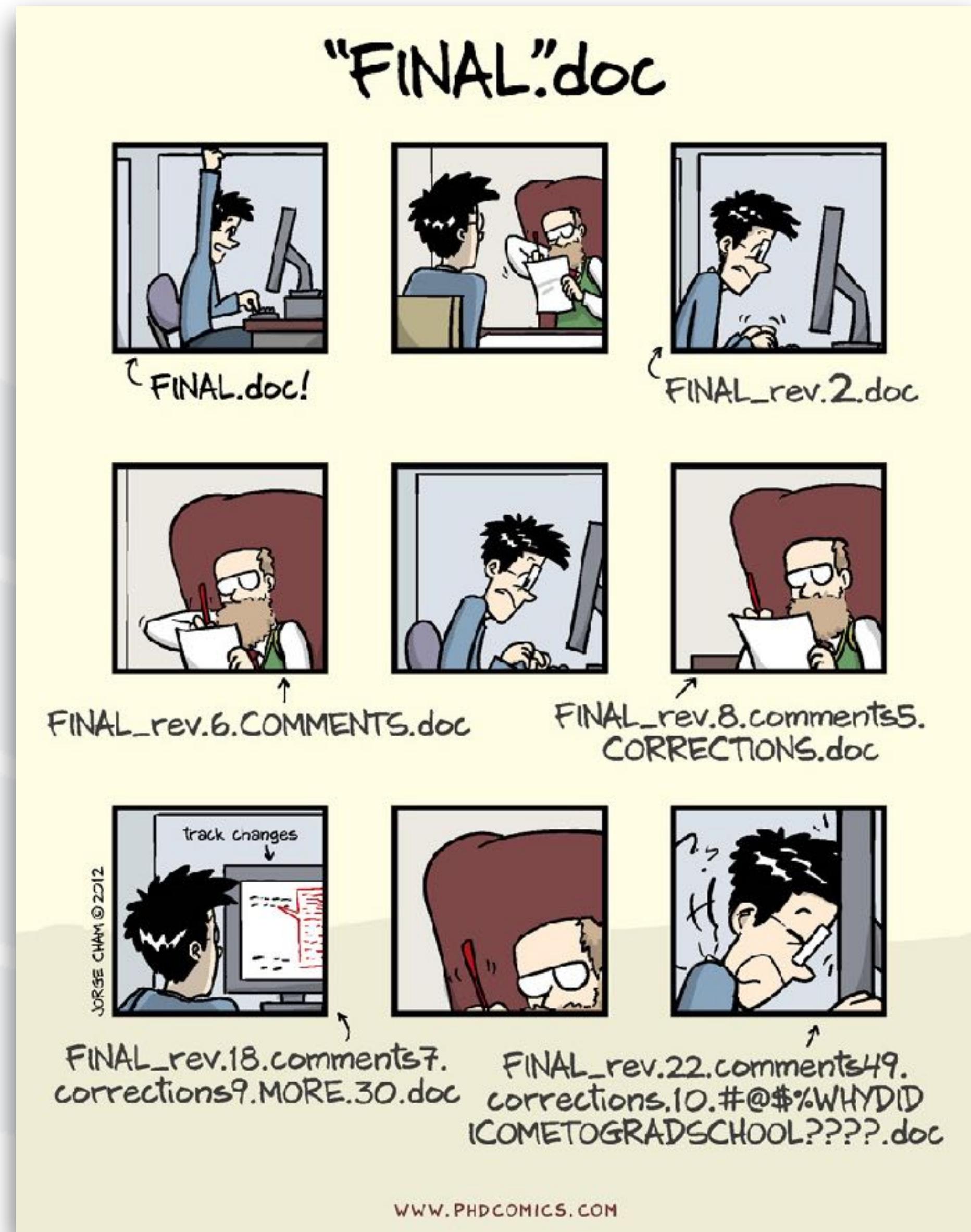


# Syllabus

- 1 Module, 12h (1.5 CFU)
  - Version control systems (with focus on git)
  - Testing systems
  - Continuous integration systems
  - Container systems



# No-control version control





# Testing makes life easy

UNIT TESTS

MONKEYUSER.COM

SO YOU ASSUMED  
THAT THIS IS THE  
CORRECT BEHAVIOR



# An introduction to modern tools for collaborative science

## Lecture 0 - Introduction to Linux and Bash

Luca Heltai <[luca.heltai@sissa.it](mailto:luca.heltai@sissa.it)>

Mathematical Analysis, Modeling, and Applications ([math.sissa.it](http://math.sissa.it))

Theoretical and Scientific Data Science ([datascience.sissa.it](http://datascience.sissa.it))

# An introduction to modern tools for collaborative science

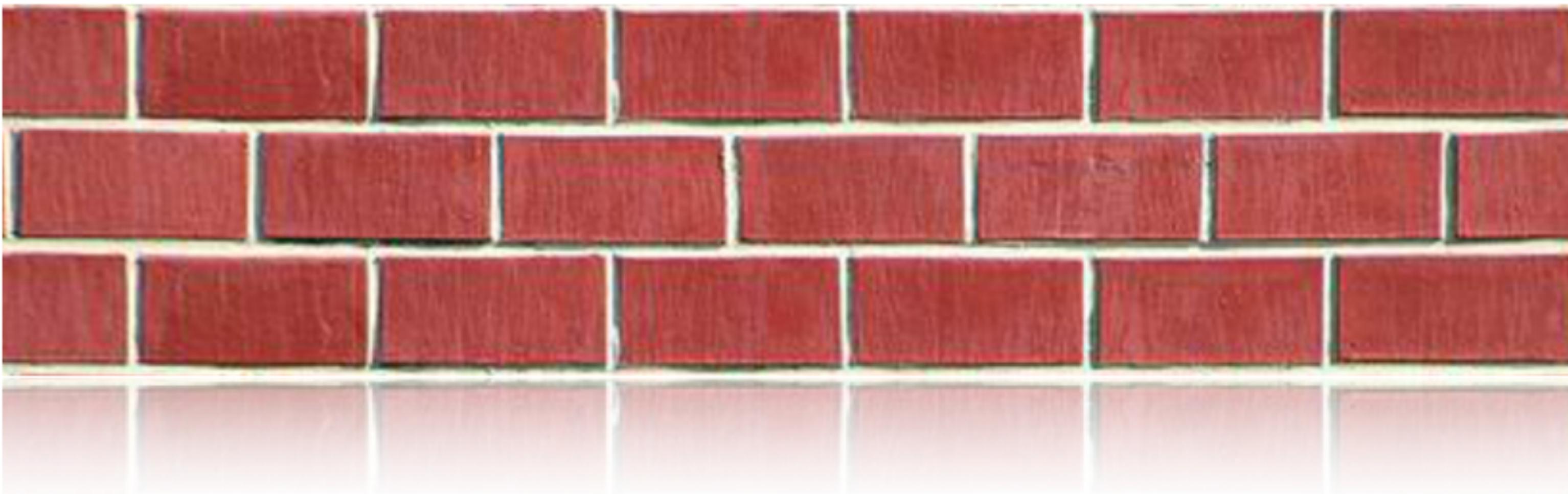
## Lecture 1 - Introduction to GIT

Luca Heltai <[luca.heltai@sissa.it](mailto:luca.heltai@sissa.it)>

Mathematical Analysis, Modeling, and Applications ([math.sissa.it](http://math.sissa.it))

Theoretical and Scientific Data Science ([datascience.sissa.it](http://datascience.sissa.it))

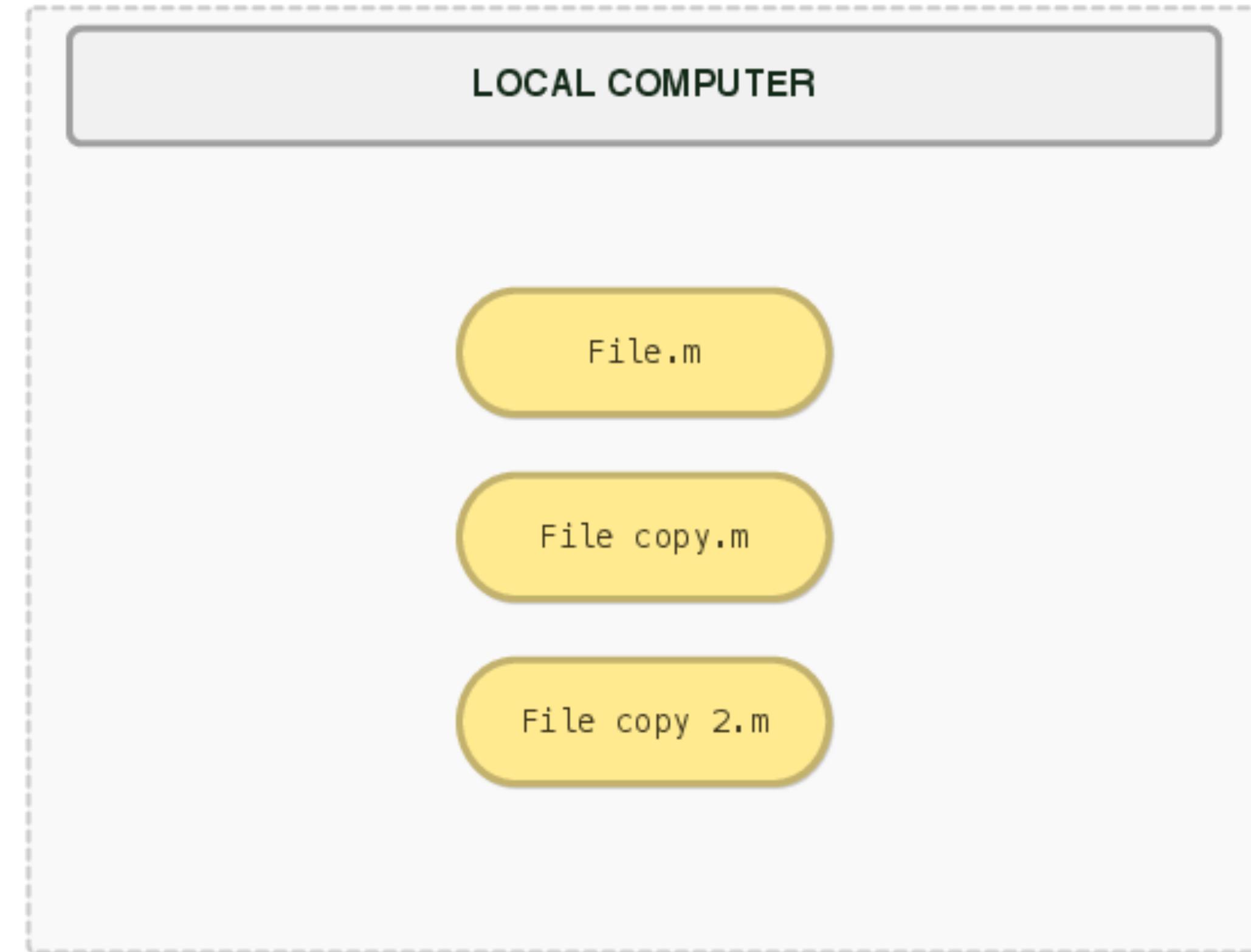
# Version Control



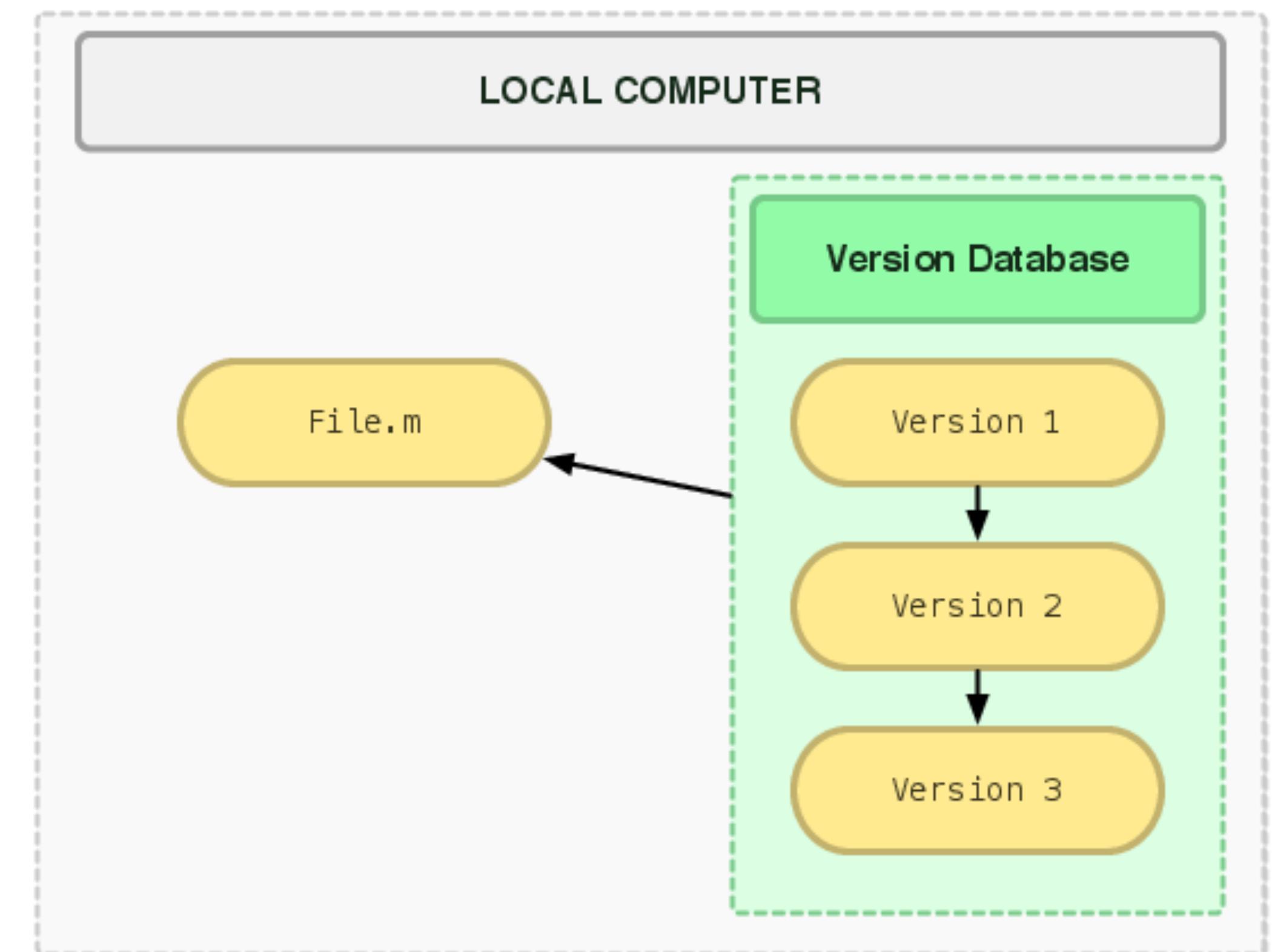
# Workflow

# History

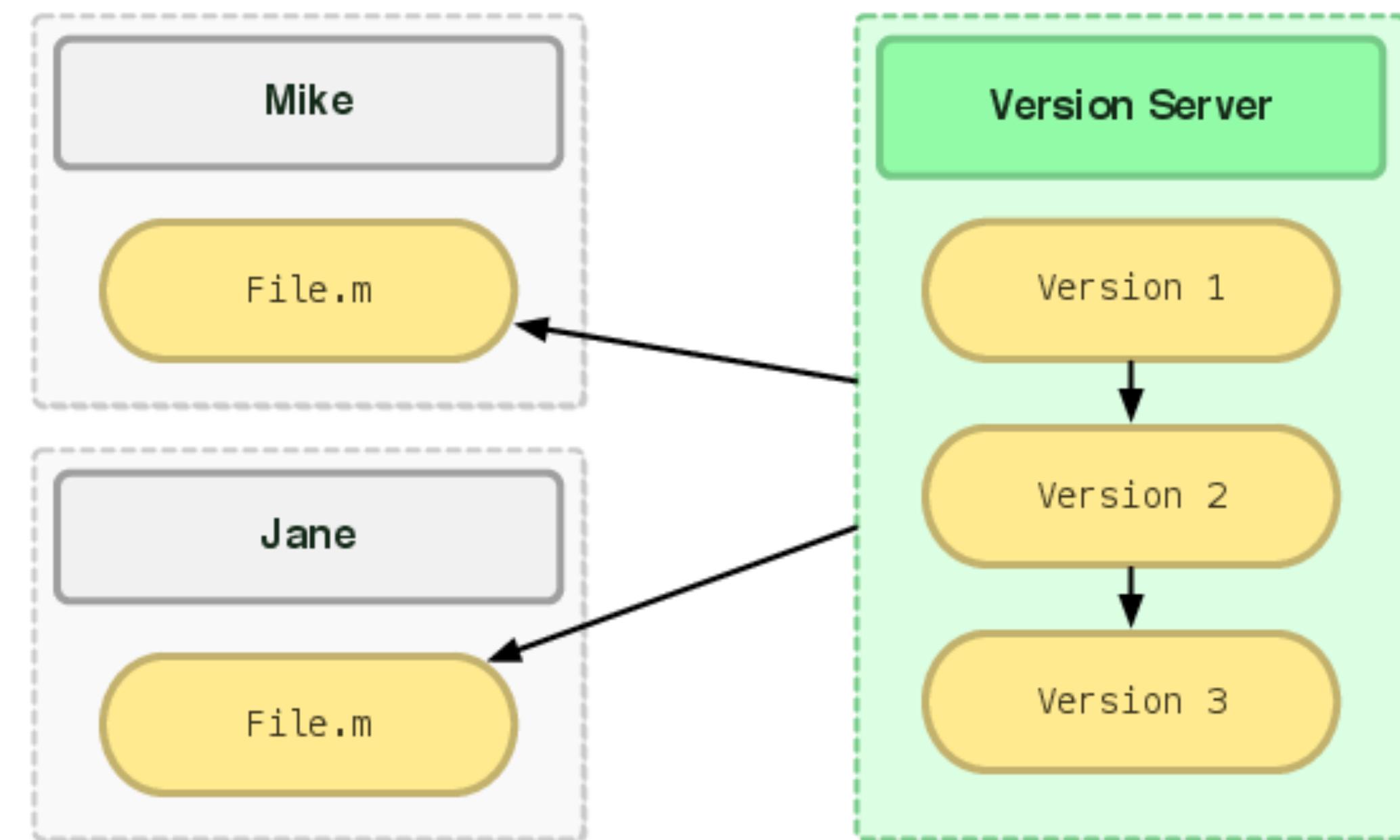
# Local Filesystem



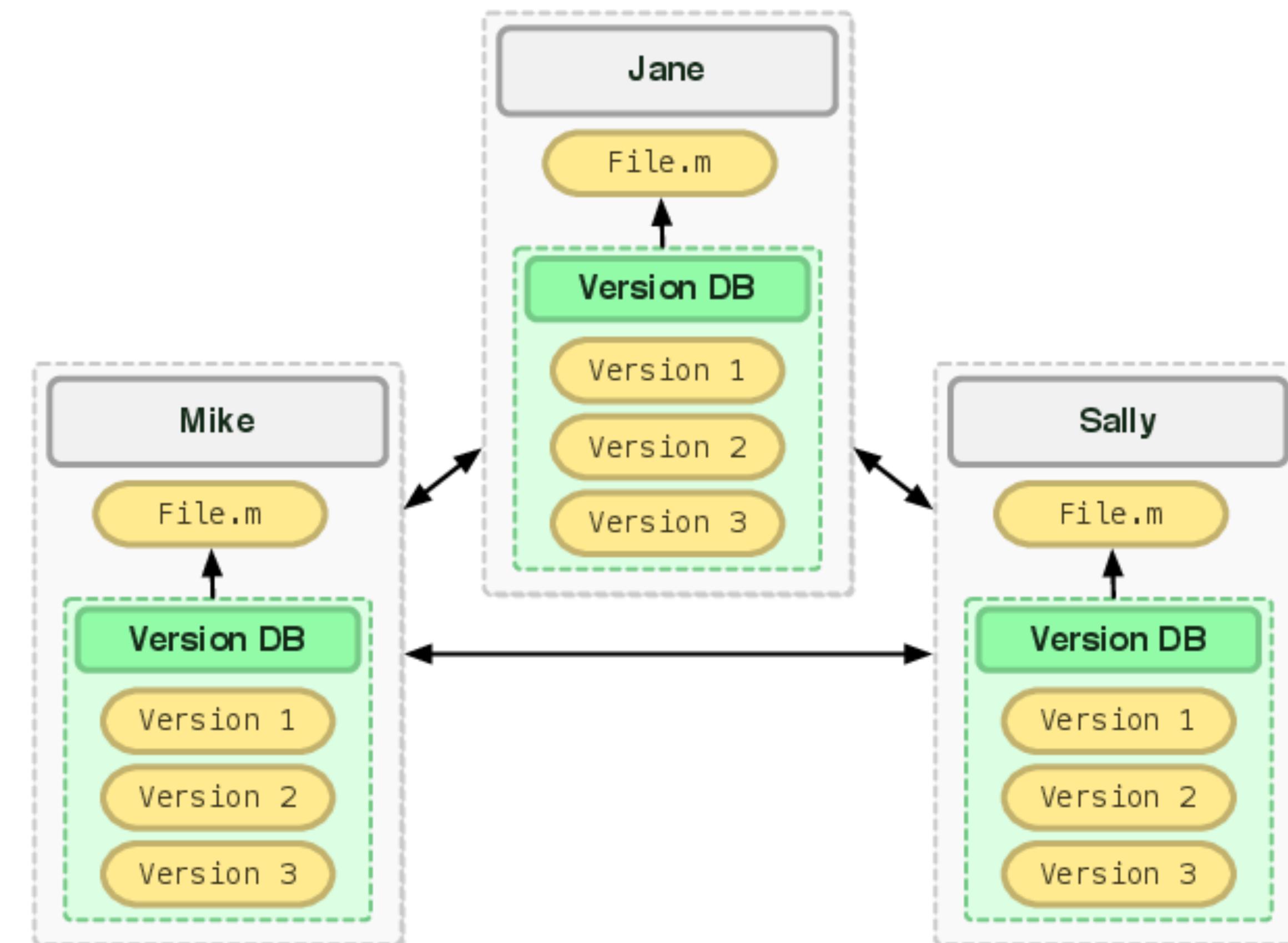
# LOCAL Version Control System



# CENTRALIZED Version Control System



# DISTRIBUTED Version Control System



# **Everything is Local (Almost)**

# No Network Required

**Create Repo**

**Status**

**Commit**

**Revisions**

**Merge**

**Diff**

**Branch**

**History**

**Rebase**

**Bisect**

**Tag**

**Local Sync**

# Advantages

**Everything is Fast**

**Everything is Transparent**

**Every Clone is a Backup**

**You Can Work Offline**

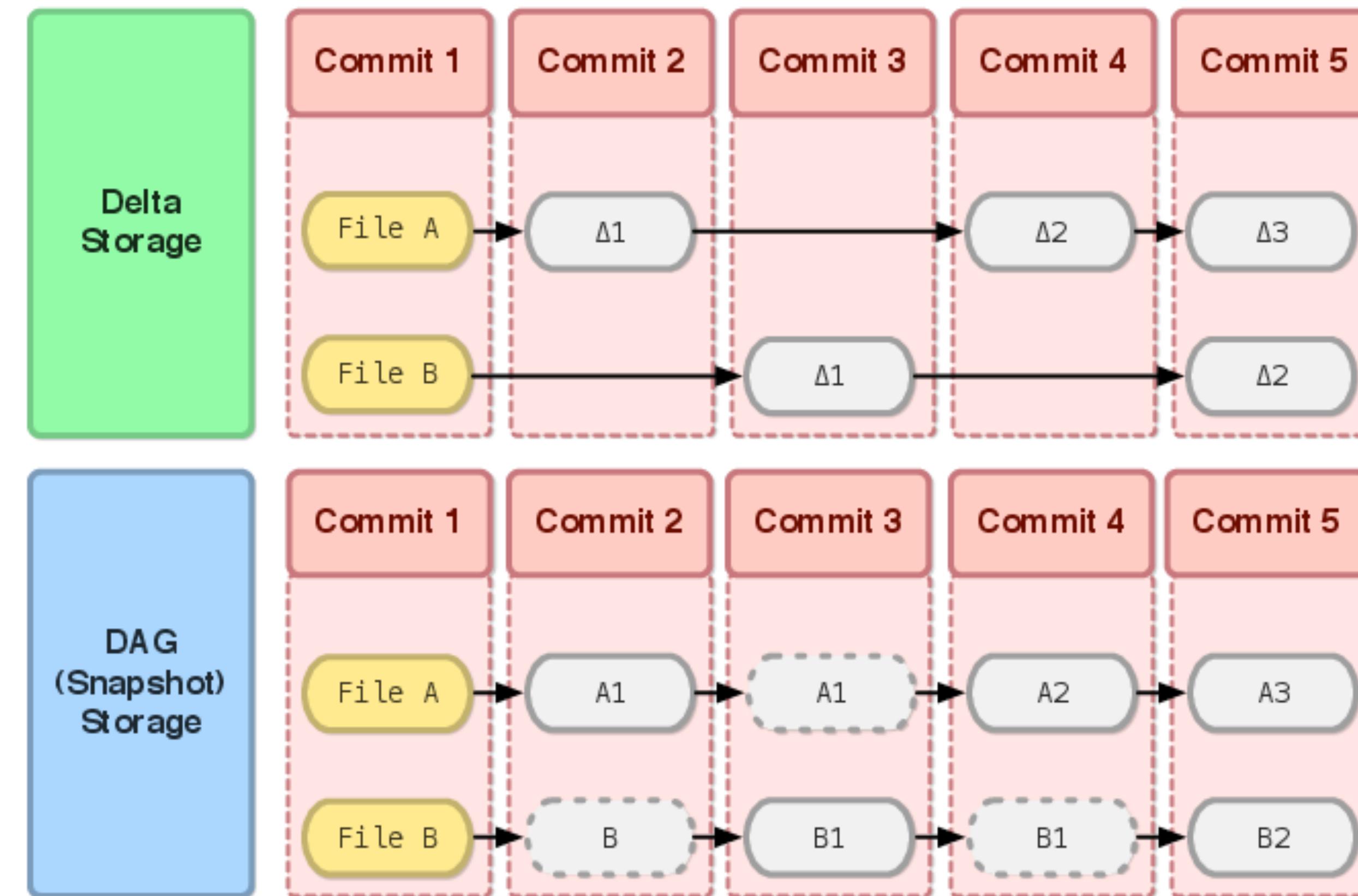
# Storage

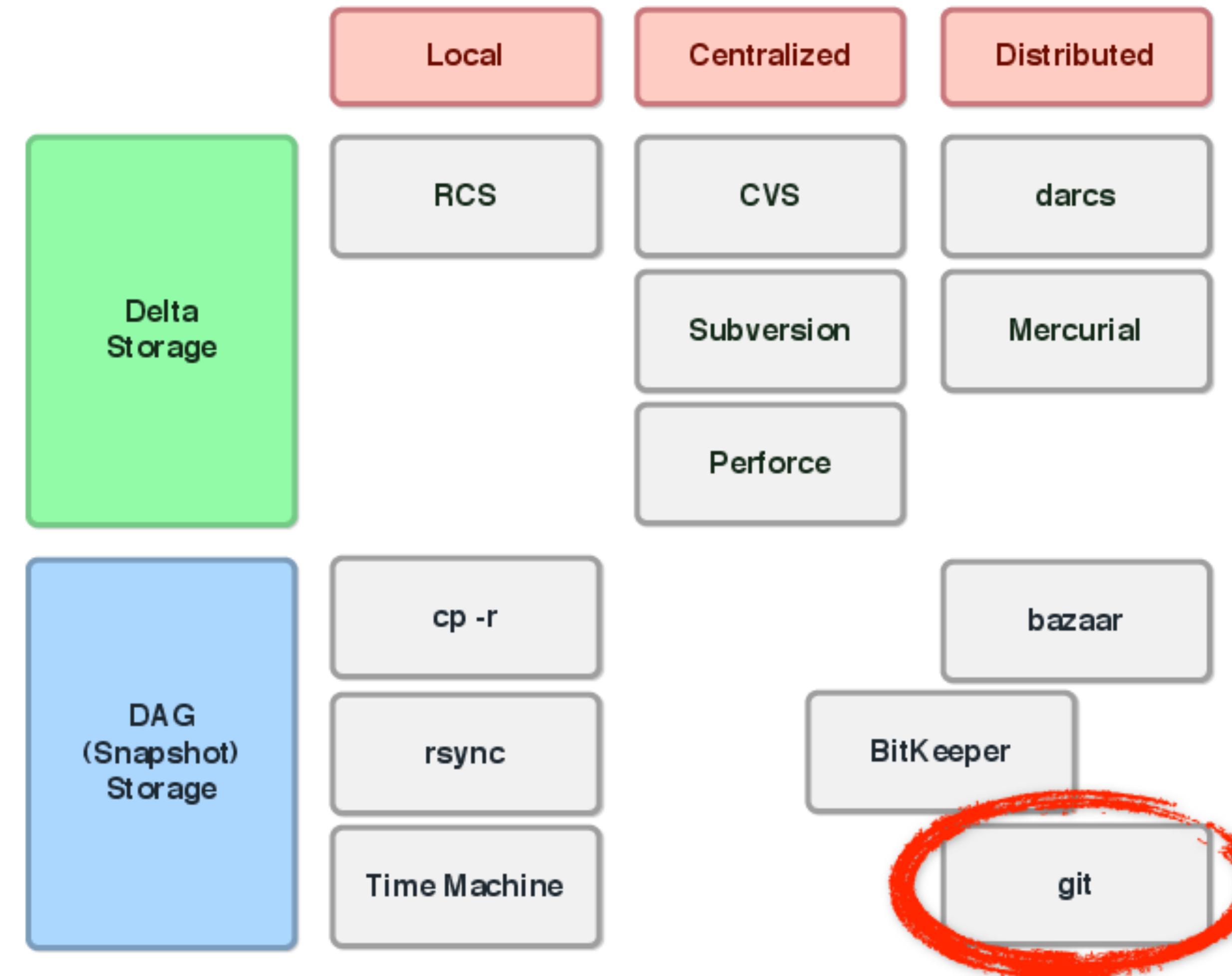
# Delta Storage

# **Snapshot Storage**

**(a.k.a. Direct Acyclic Graph)**

# Delta vs. Snapshot





# About Git

# **Git in a Nutshell**

**Free and Open Source**

**Distributed Version Control System**

**Designed to Handle Large Projects**

**Fast and Efficient**

**Excellent Branching and Merging**

# Projects Using Git

**Git**

**Linux**

**Perl**

**Eclipse**

**Qt**

**Rails**

**Android**

**PostgreSQL**

**KDE**

**Gnome**

# Under The Hood

# Git Directory

```
$ ls -lA
-rw-r--r--@ 1 pbhogan staff 21508 Jul  3 15:21 .DS_Store
drwxr-xr-x 14 pbhogan staff   476 Jul  3 14:46 .git
-rw-r--r--@ 1 pbhogan staff   115 Aug 11 2010 .gitignore
-rw-r--r--@ 1 pbhogan staff   439 Dec 27 2010 Info.plist
drwxr-xr-x 17 pbhogan staff   578 Feb  6 10:54 Resources
drwxr-xr-x  7 pbhogan staff   238 Jul 18 2010 Source
...
```

# Git Directory

```
$ tree .git
.git
├── HEAD
├── config
├── description
├── hooks
│   ├── post-commit.sample
│   └── ...
├── info
│   └── exclude
├── objects
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags
```

**.git only in root of  
Working Directory  
(unlike Subversion)**

# Git Directory

**Configuration File**

**Hooks**

**Object Database**

**References**

**Index**

# Git Directory

**Configuration File**

**Hooks**

**Object Database**

**References**

**Index**

# Object Database

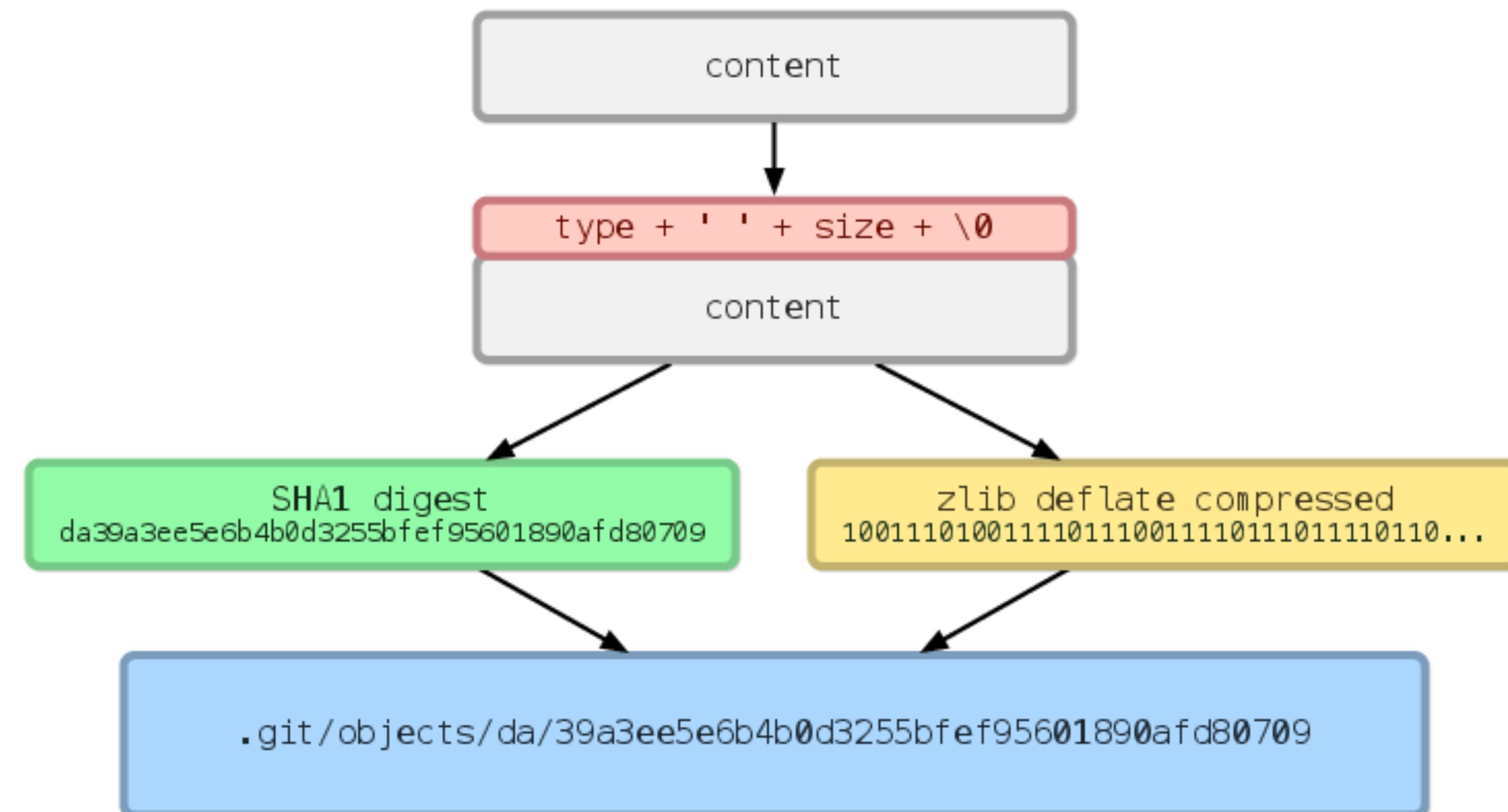
# Object Database

≈ **NSDictionary**  
**(Hash Table / Key–Value Store)**

# Object Database

content

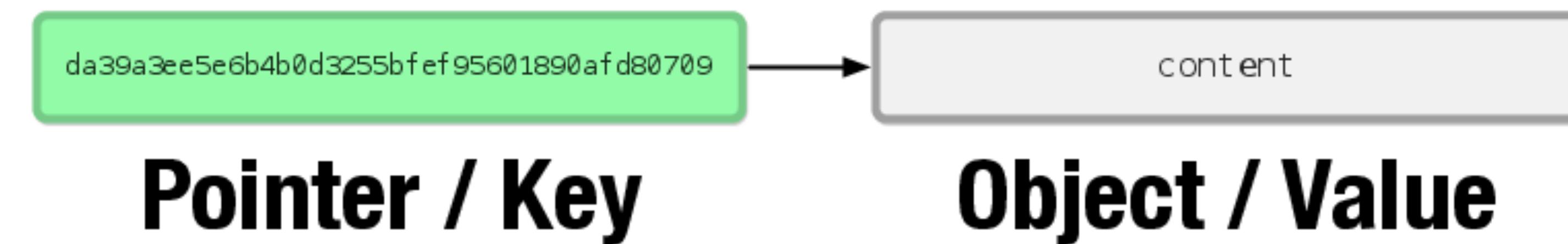
# Object Database



**"loose format"**

# Object Database

**≈ NSDictionary**



# Object Database

da39a3ee5e6b4b0d3255bfef95601890afd80709

da39a3ee5e6b4b0d3255...

da39a3ee5e6...

da39a...

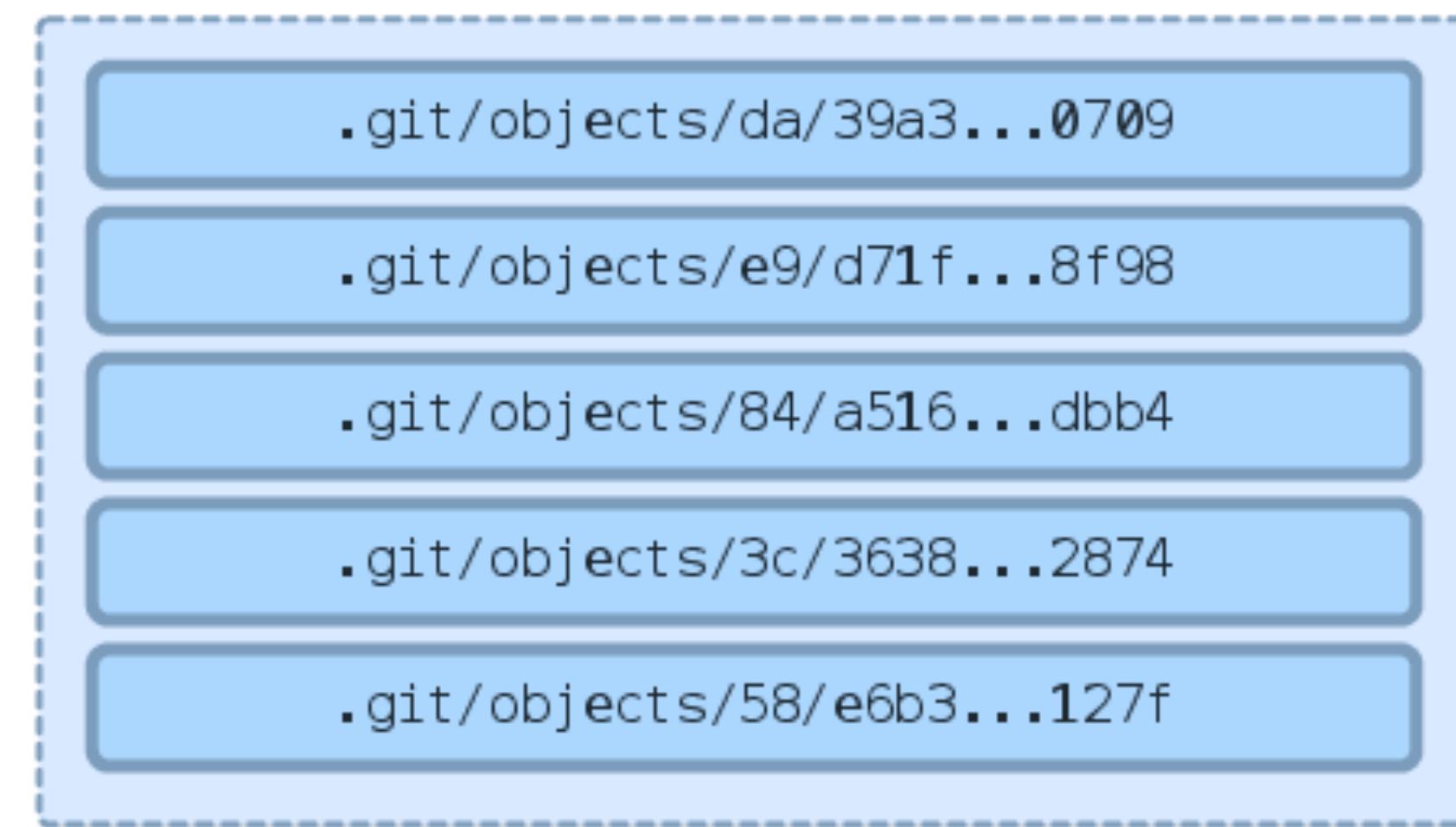
**Equivalent if common  
prefix is unique!**

# Object Database

## Garbage Collection

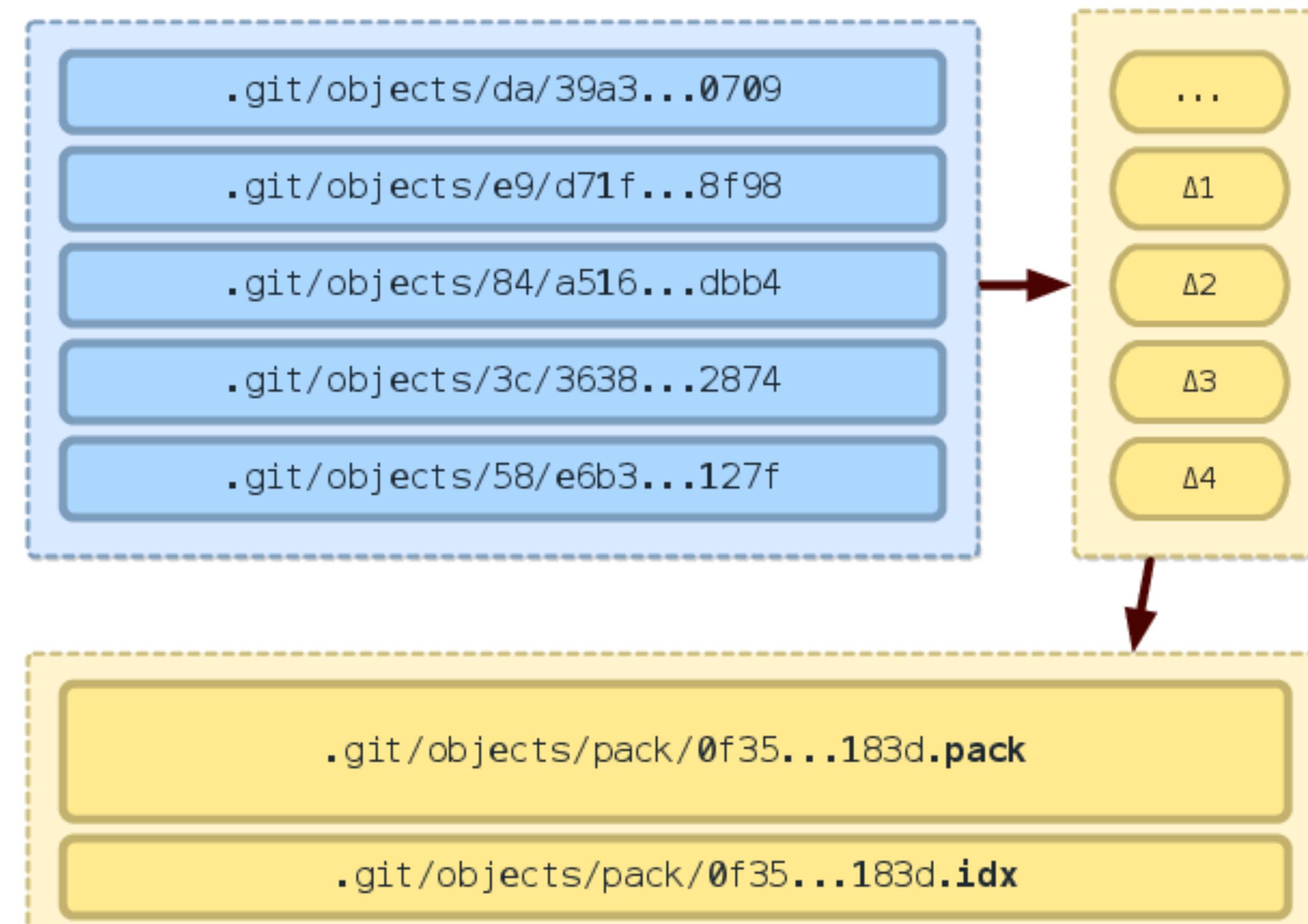
# Object Database

## Similar Objects



git gc

# Object Database



**"packed format"**

# Object Database

## Four Object Types

# Object Database

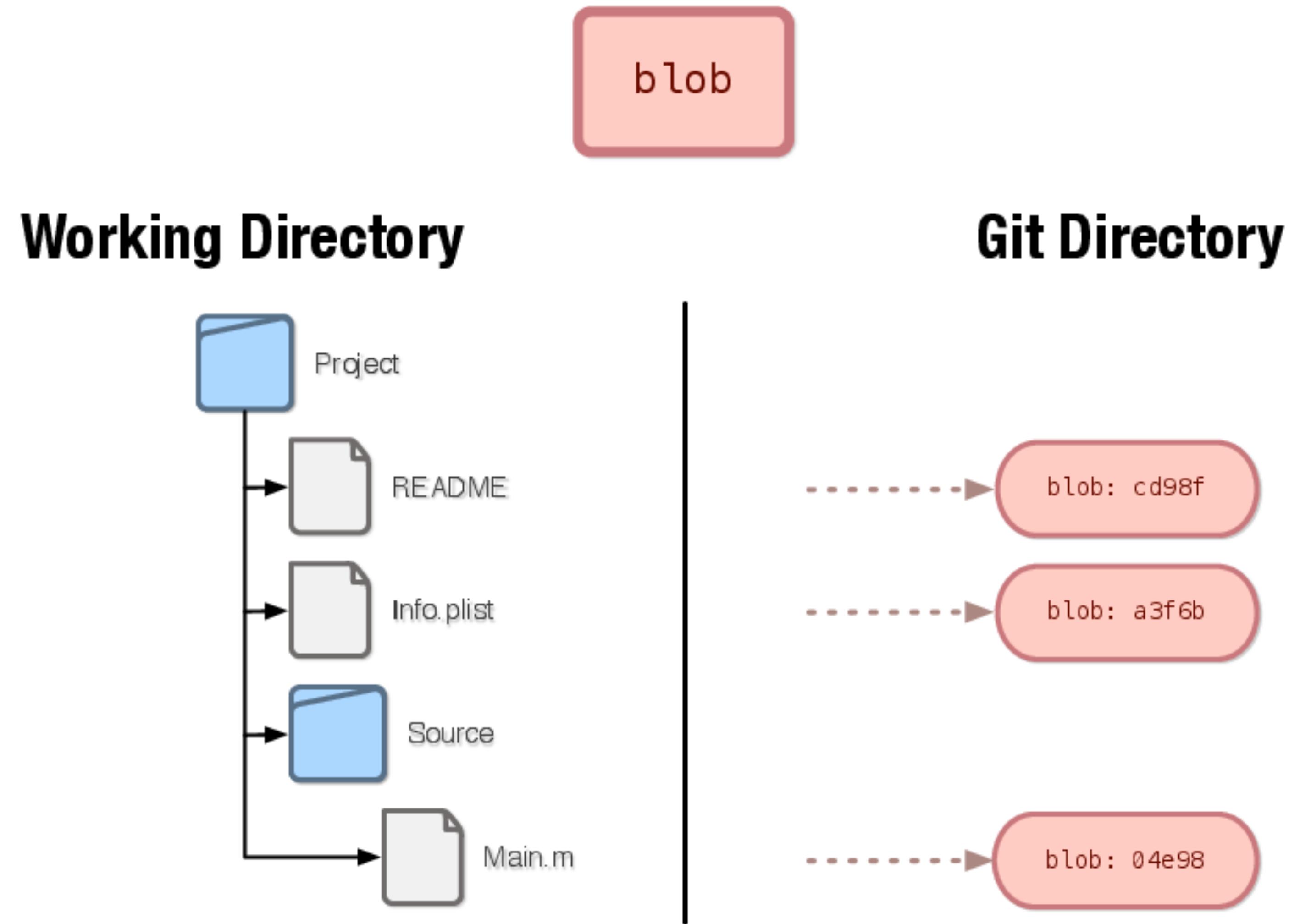
blob

tree

commit

tag

# Object Database



# Object Database

blob

blob 109\0

```
#import <Cocoa/Cocoa.h>

int main(int argc, const char *argv[])
{
    return NSApplicationMain(argc, argv);
}
```

# Object Database

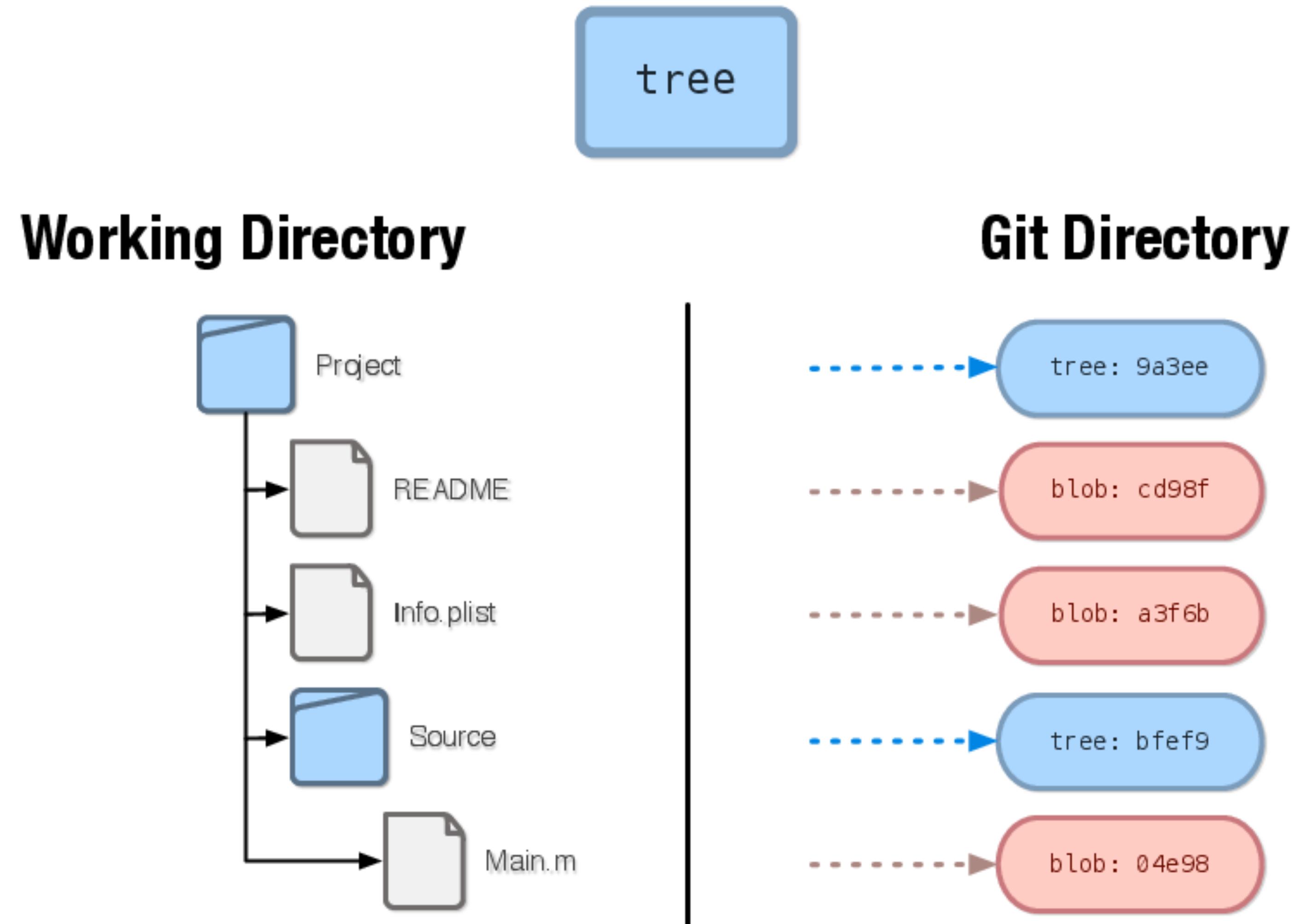
blob

tree

commit

tag

# Object Database



# Object Database

tree

tree 84\0

100644 blob cd98f	README
100644 blob a3f6b	Info.plist
040000 tree bfef9	Source

# Object Database

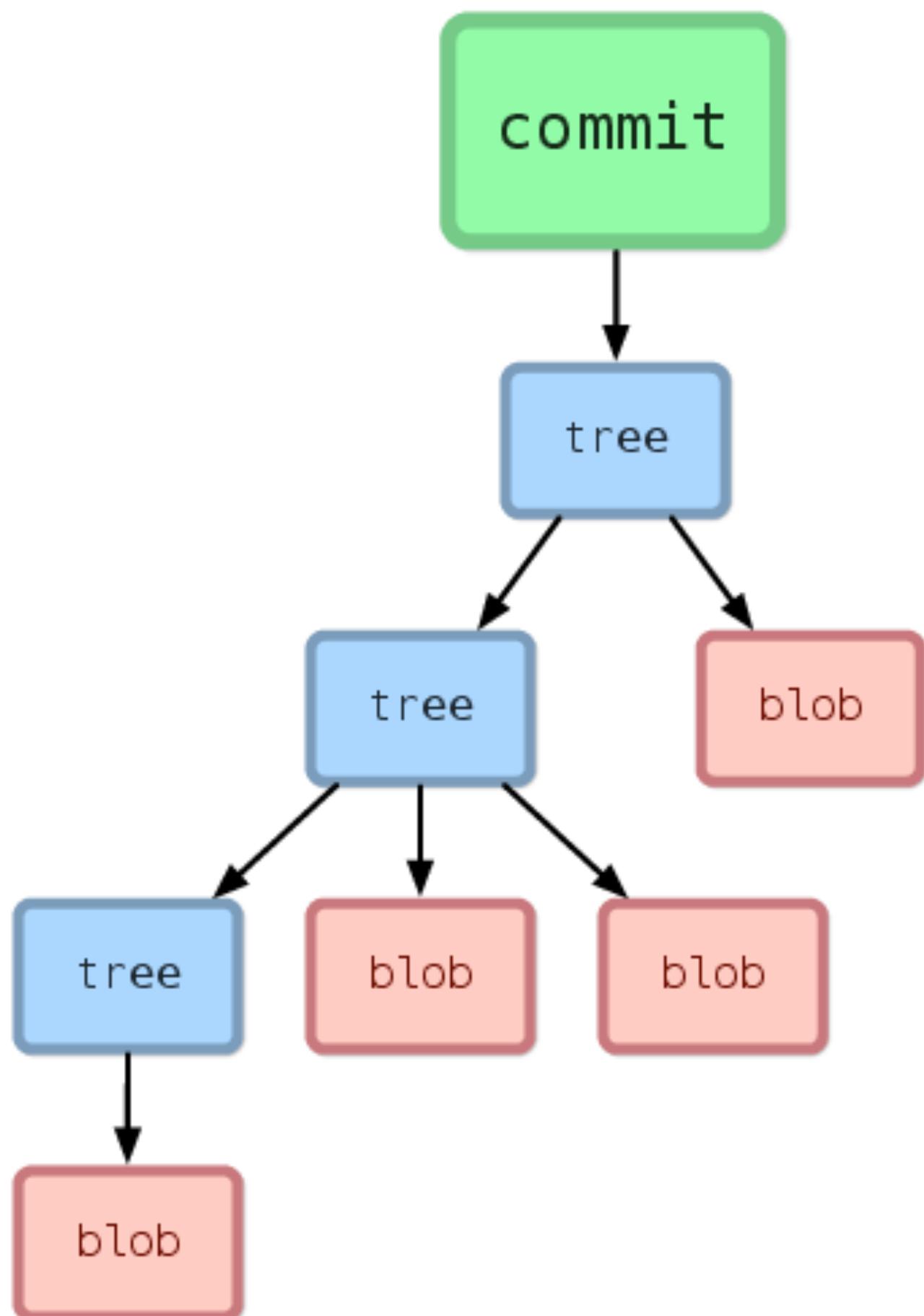
blob

tree

commit

tag

# Object Database



# Object Database

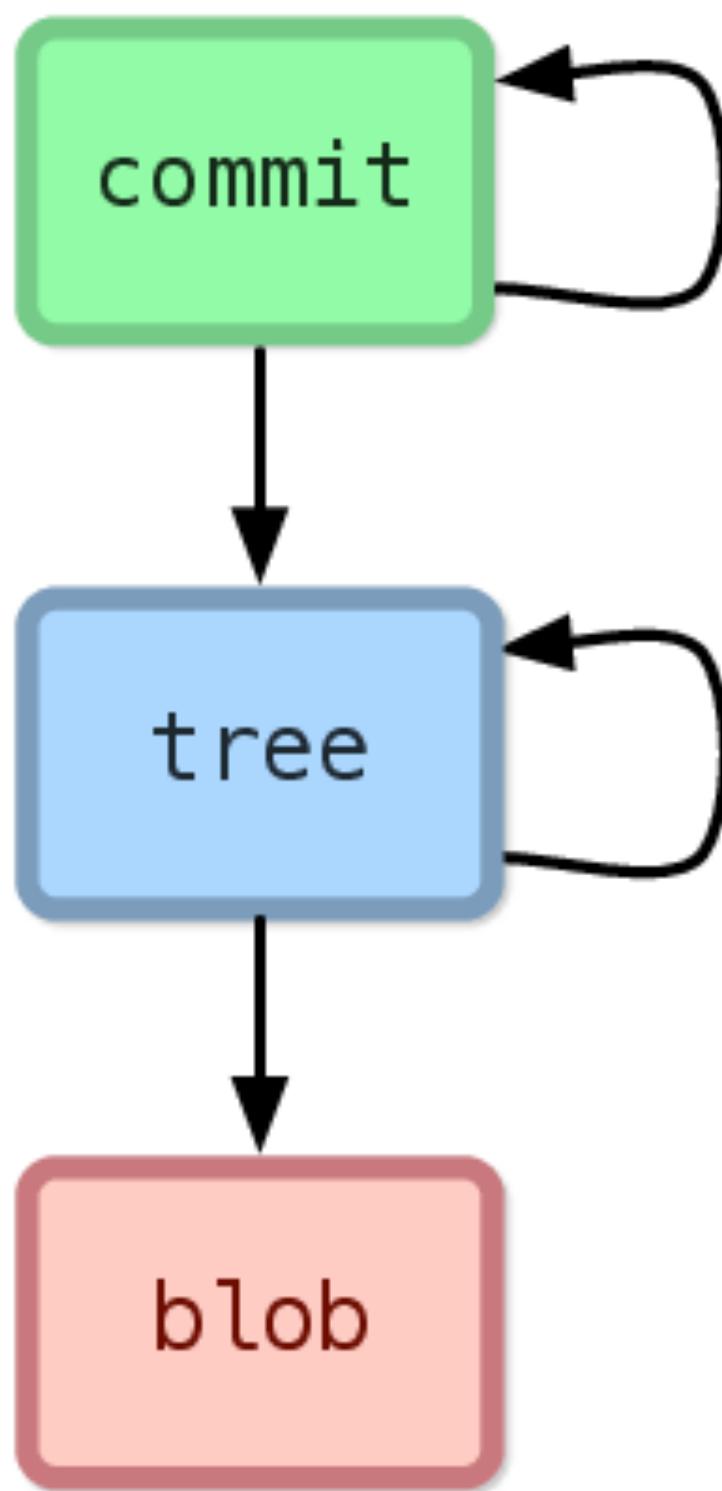
commit

commit 155\0

```
tree 9a3ee
parent fb39e
author Patrick Hogan <pbhogan@gmail.com> 1311810904
committer Patrick Hogan <pbhogan@gmail.com> 1311810904
```

Fixed a typo in README.

# Object Database



# Object Database

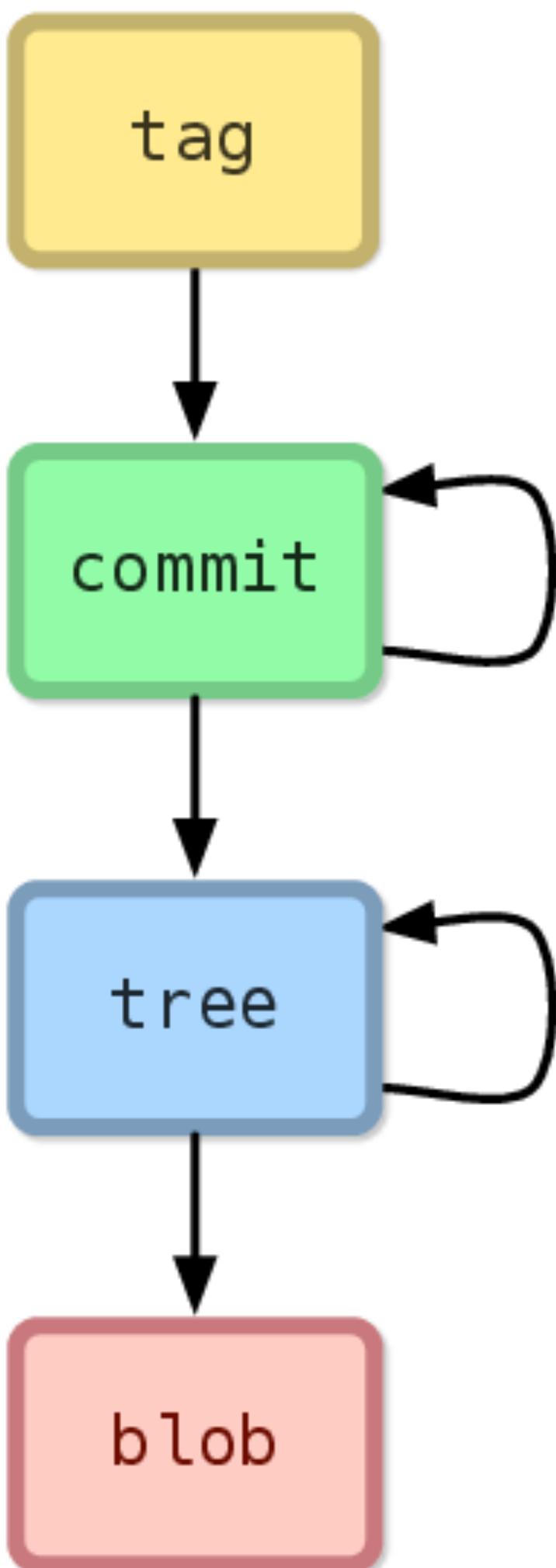
blob

tree

commit

tag

# Object Database



# Object Database

tag

tag 121\0

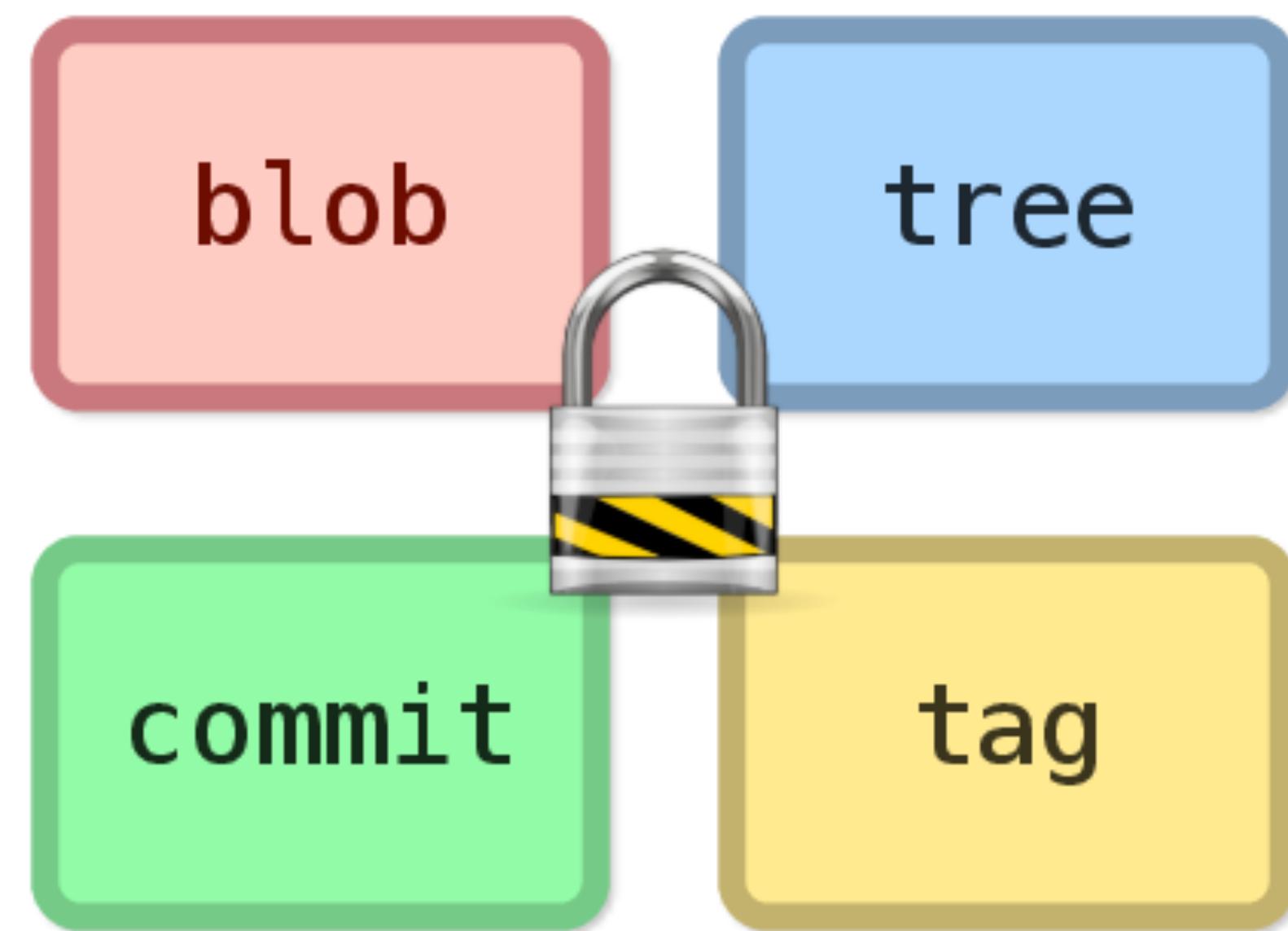
```
object e4d23e
type commit
tag v1.2.0
tagger Patrick Hogan <pbhogan@gmail.com> 1311810904
```

Version 1.2 release -- FINALLY!

.git/objects/20/c71174453dc760692cd1461275bf0cffeb772f

.git/refs/tags/v1.2.0

# Object Database



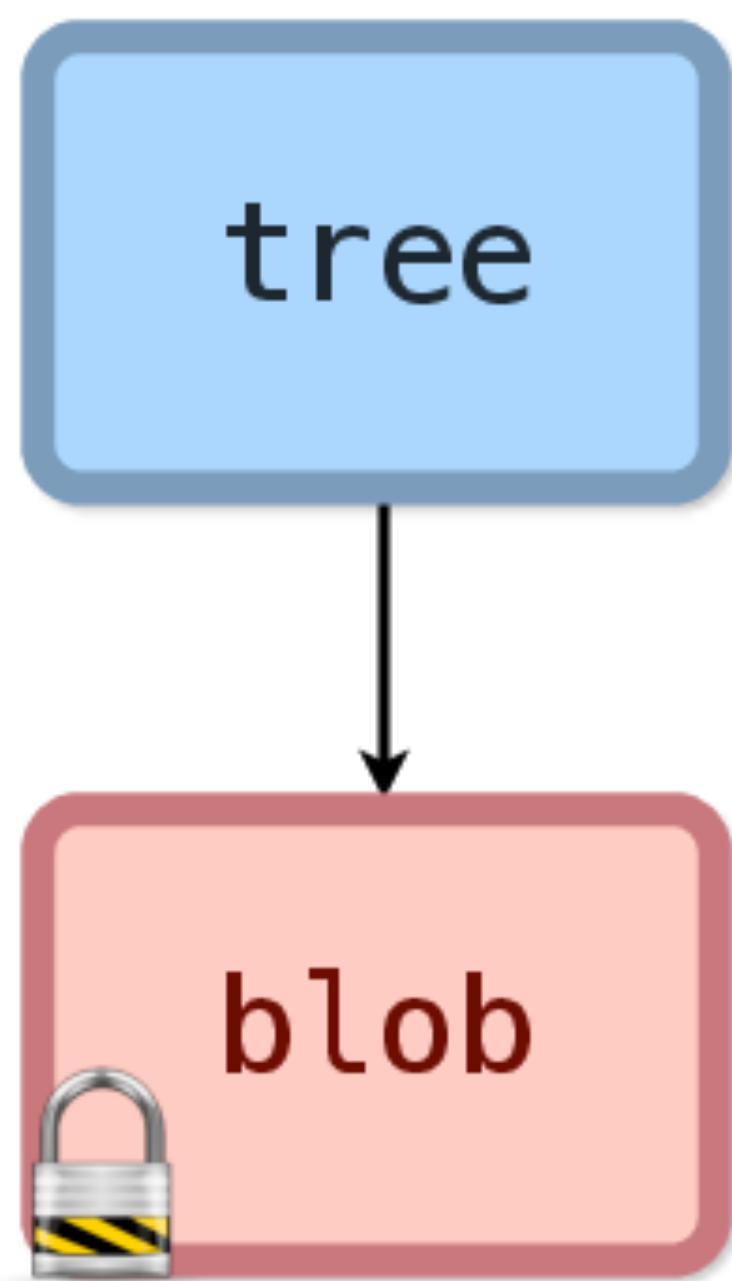
**Immutable!**

# **Never Removes Data (Almost)**

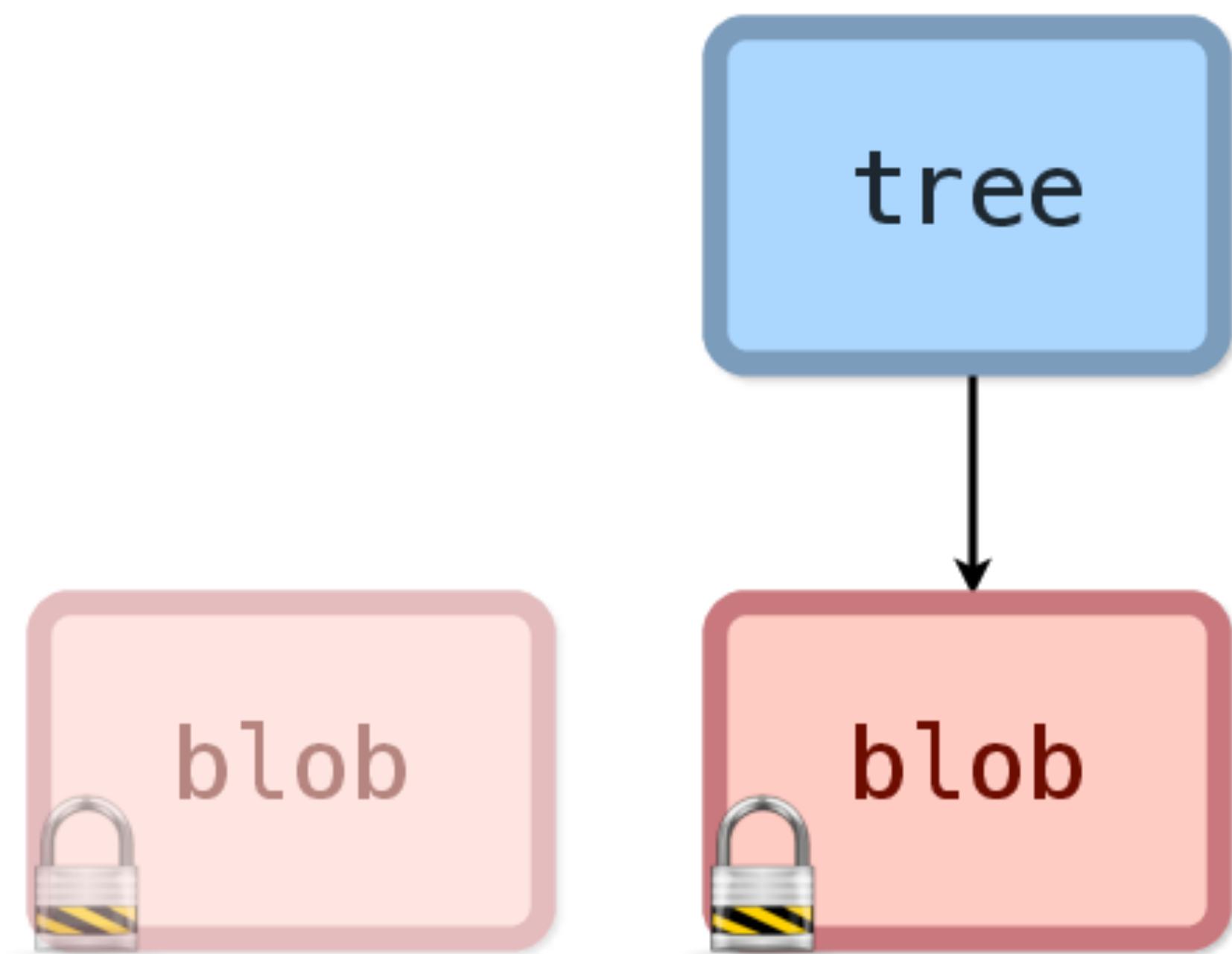
# **"Rewriting History"**

# **Writes Alternate History**

# Object Database



# Object Database



# Git Directory

**Configuration File**

**Hooks**

**Object Database**

**References**

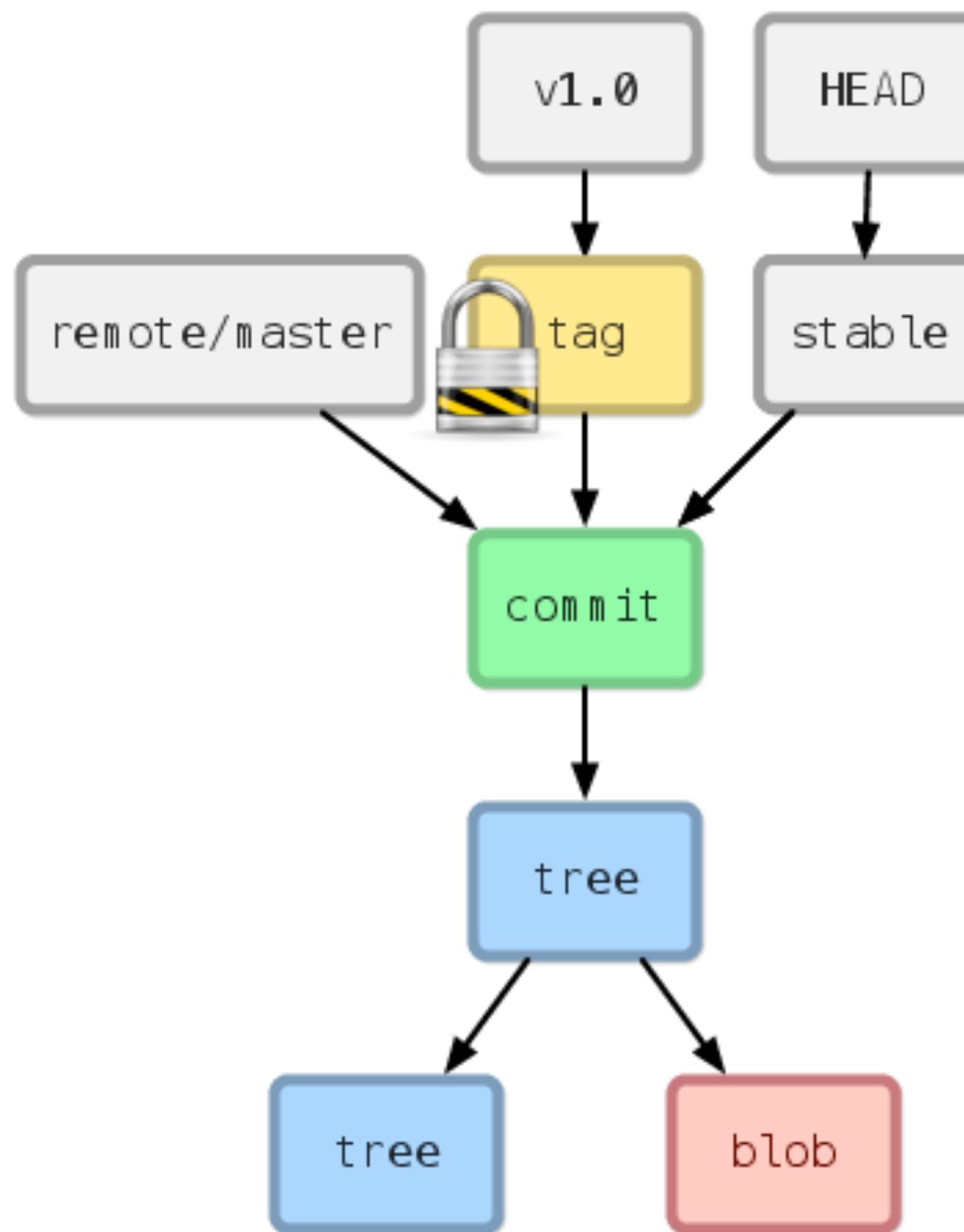
**Index**

# References

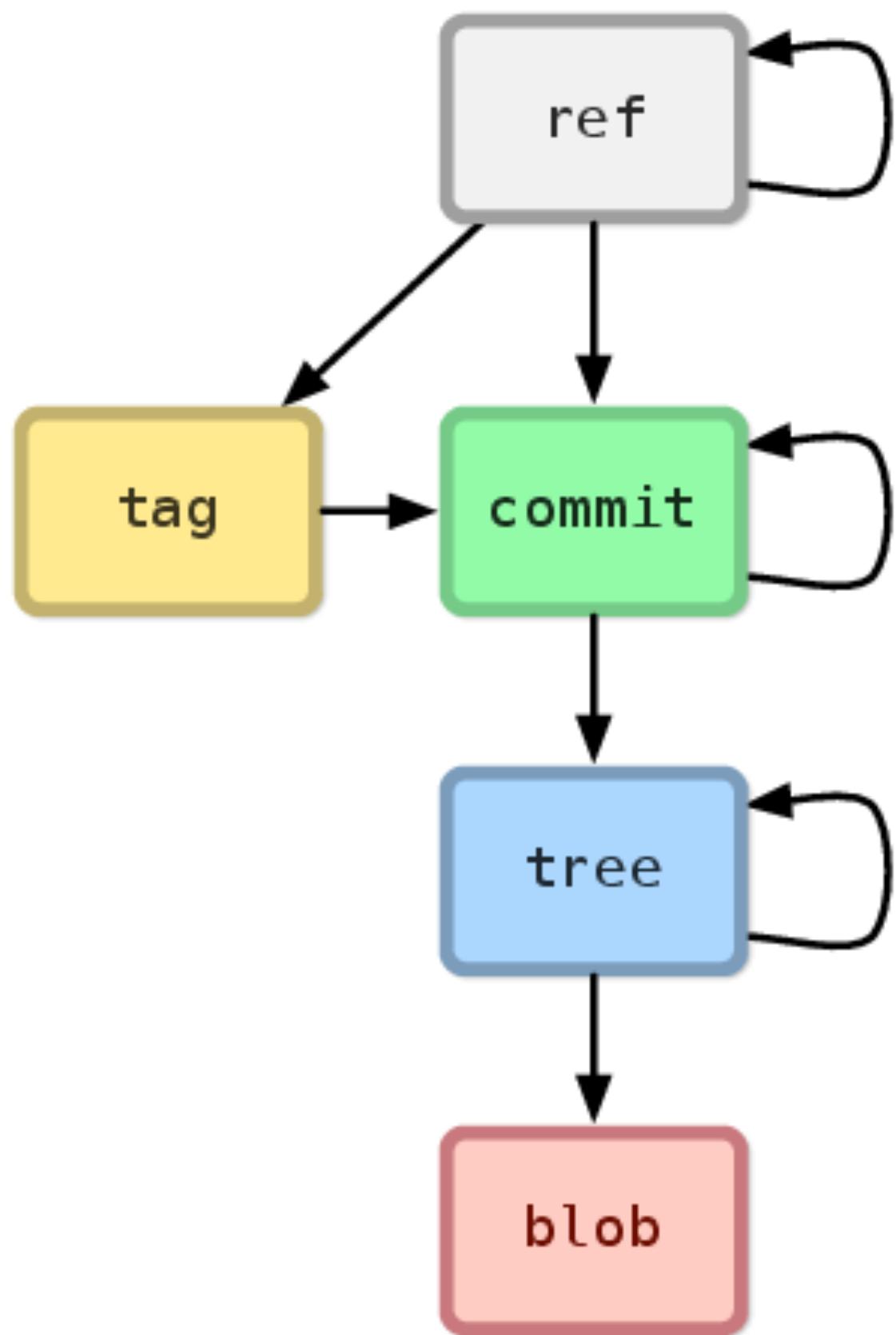
# References

**Lightweight, Movable  
Pointers to Commits  
(and other things)**

# References

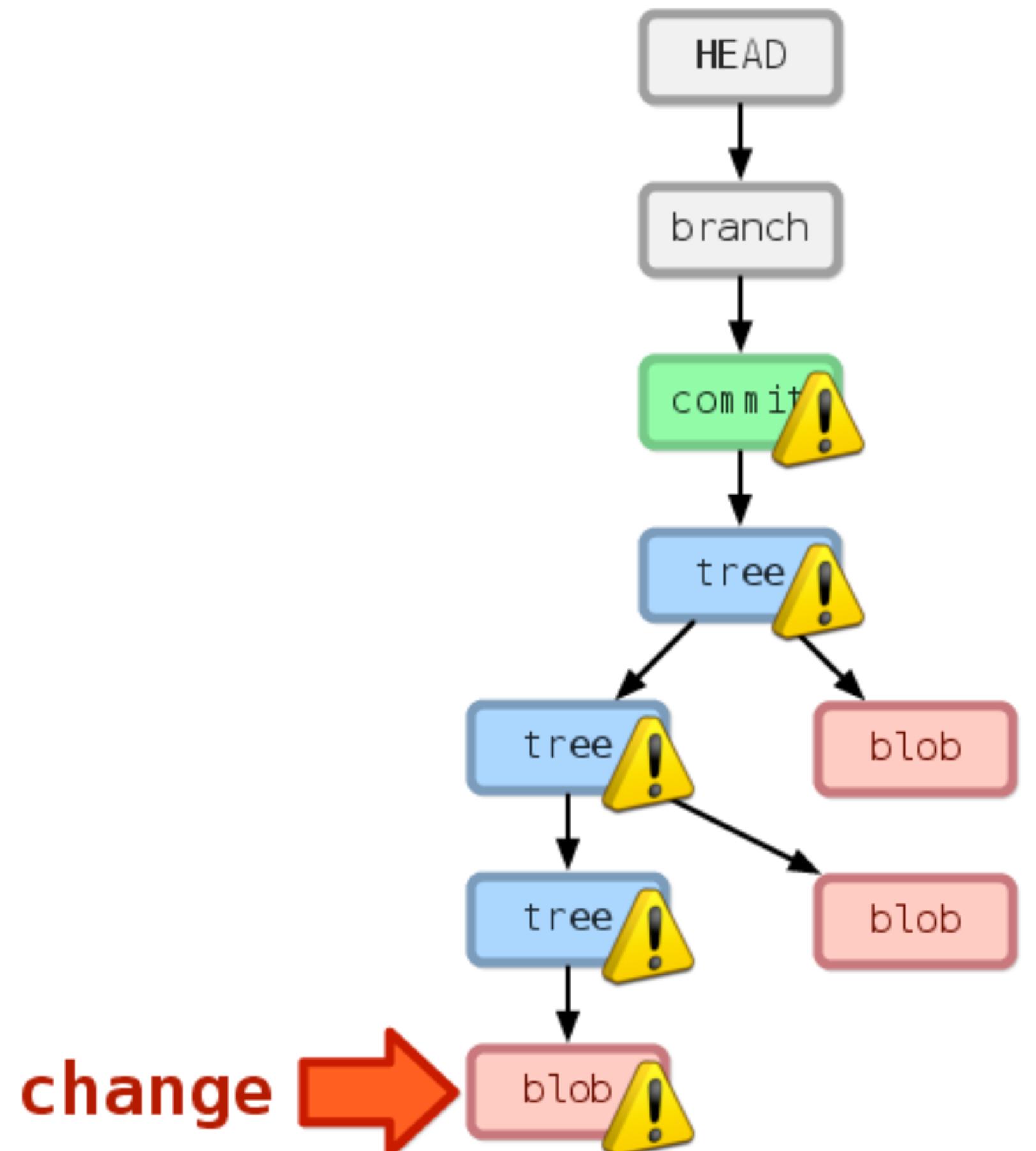


# References



# Scenario

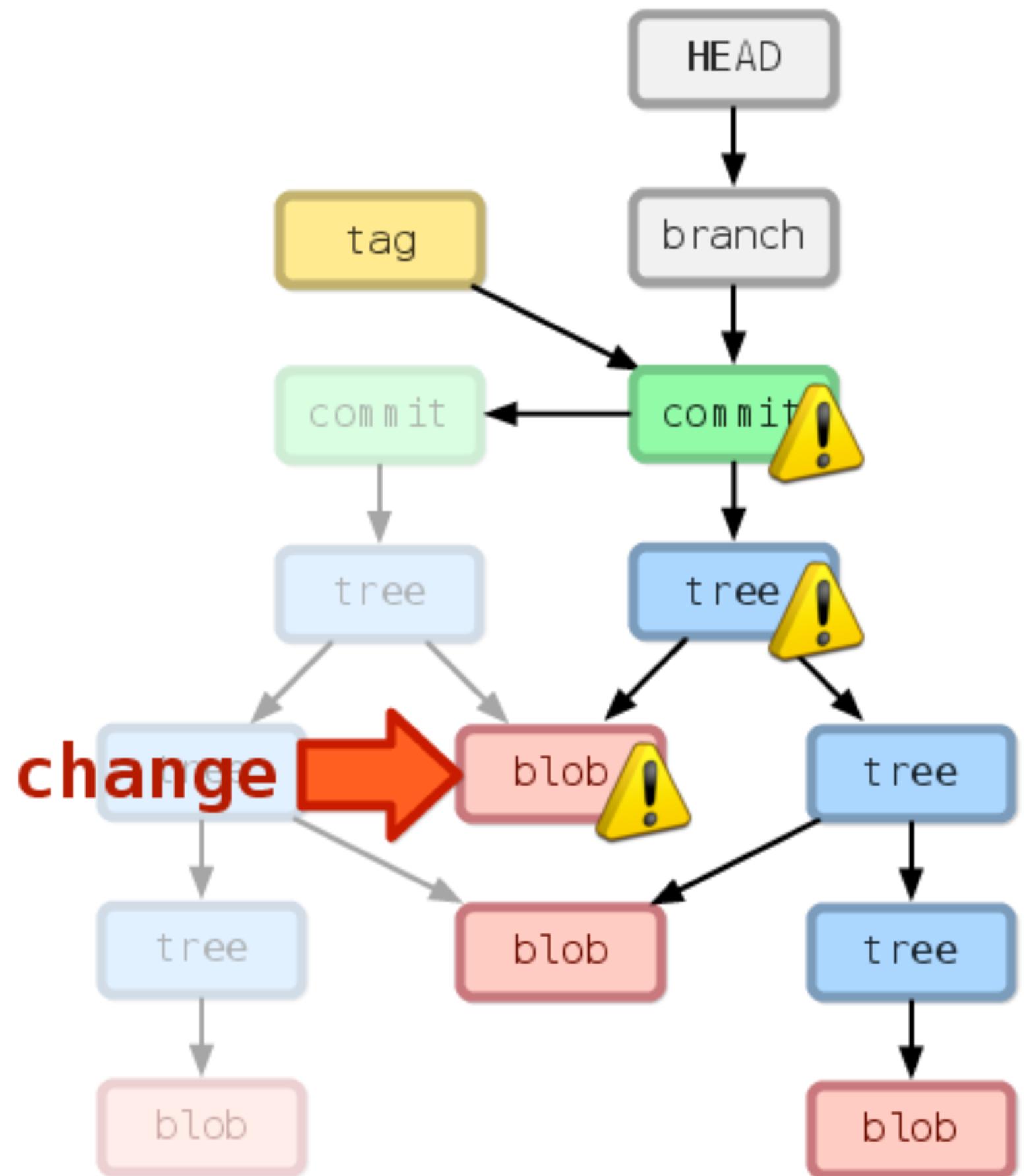
# Scenario



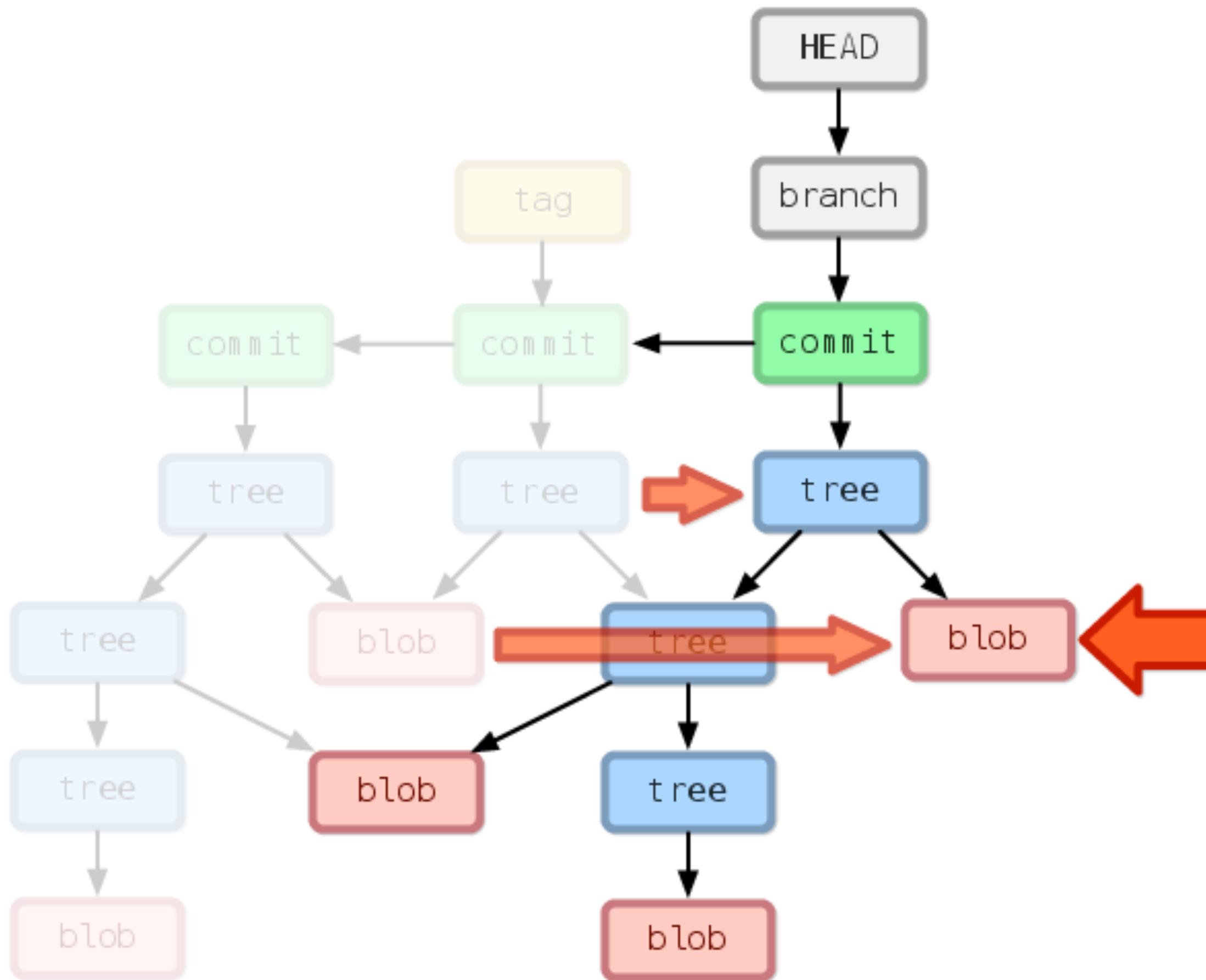
# Scenario



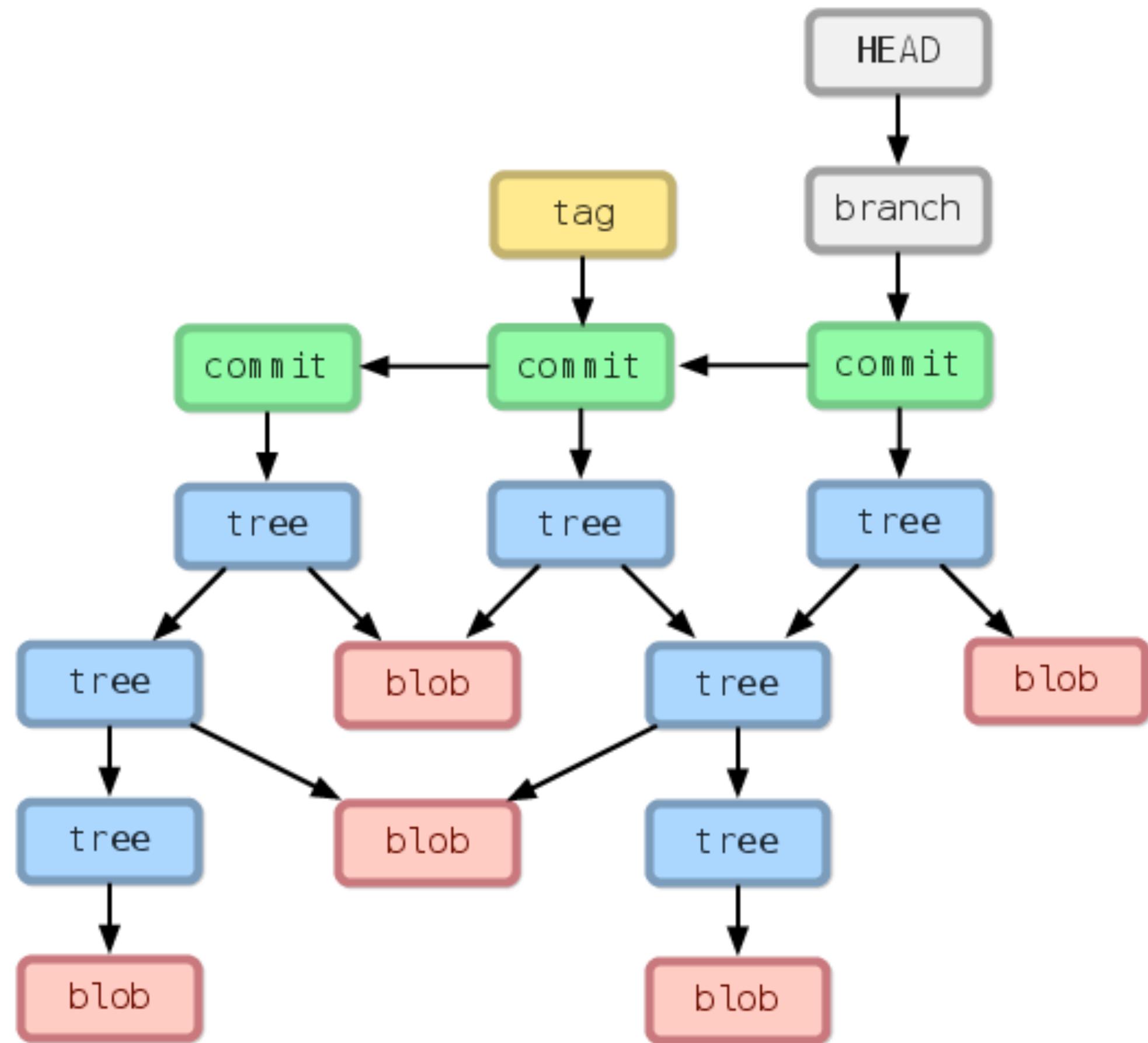
# Scenario



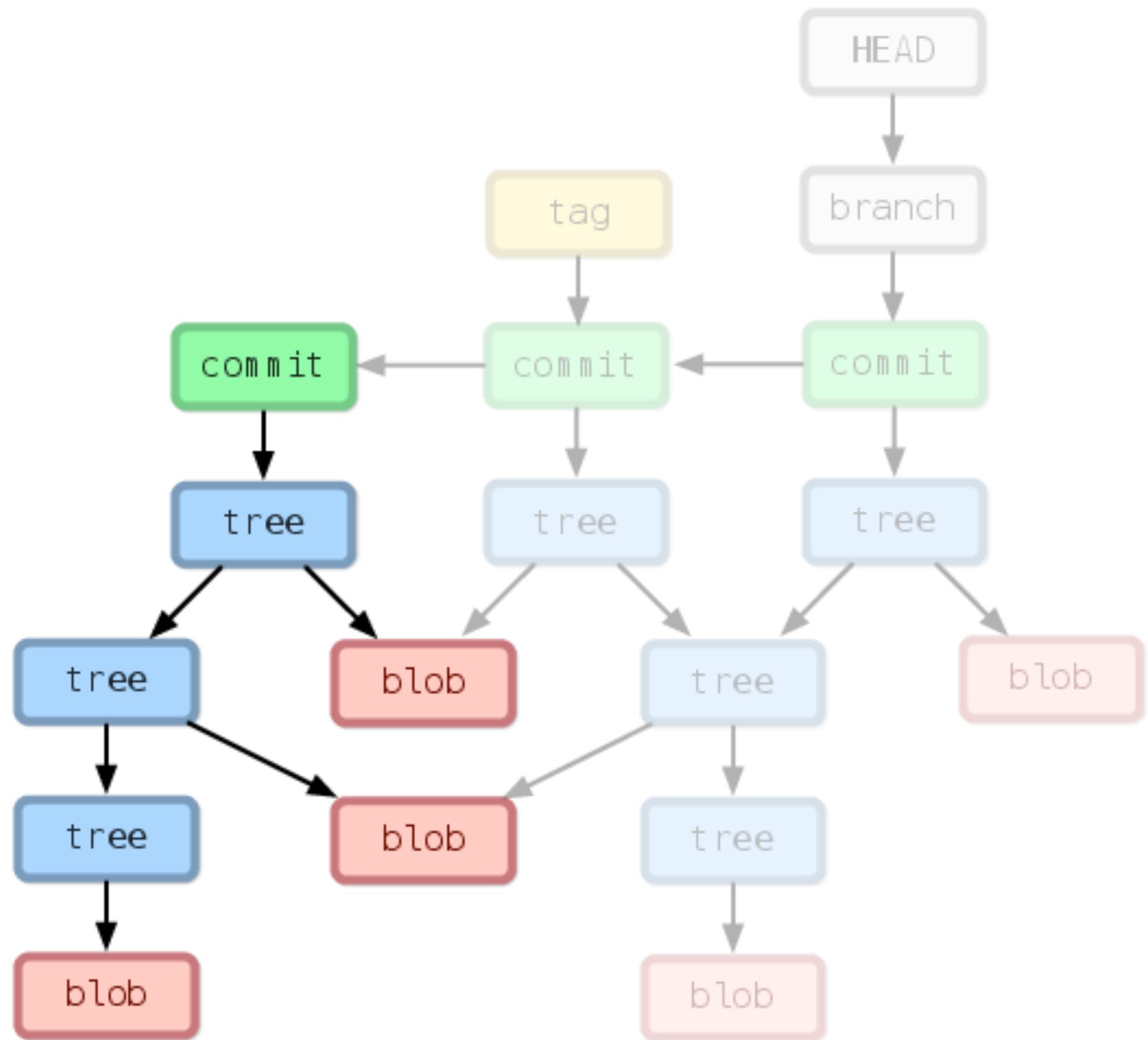
# Scenario



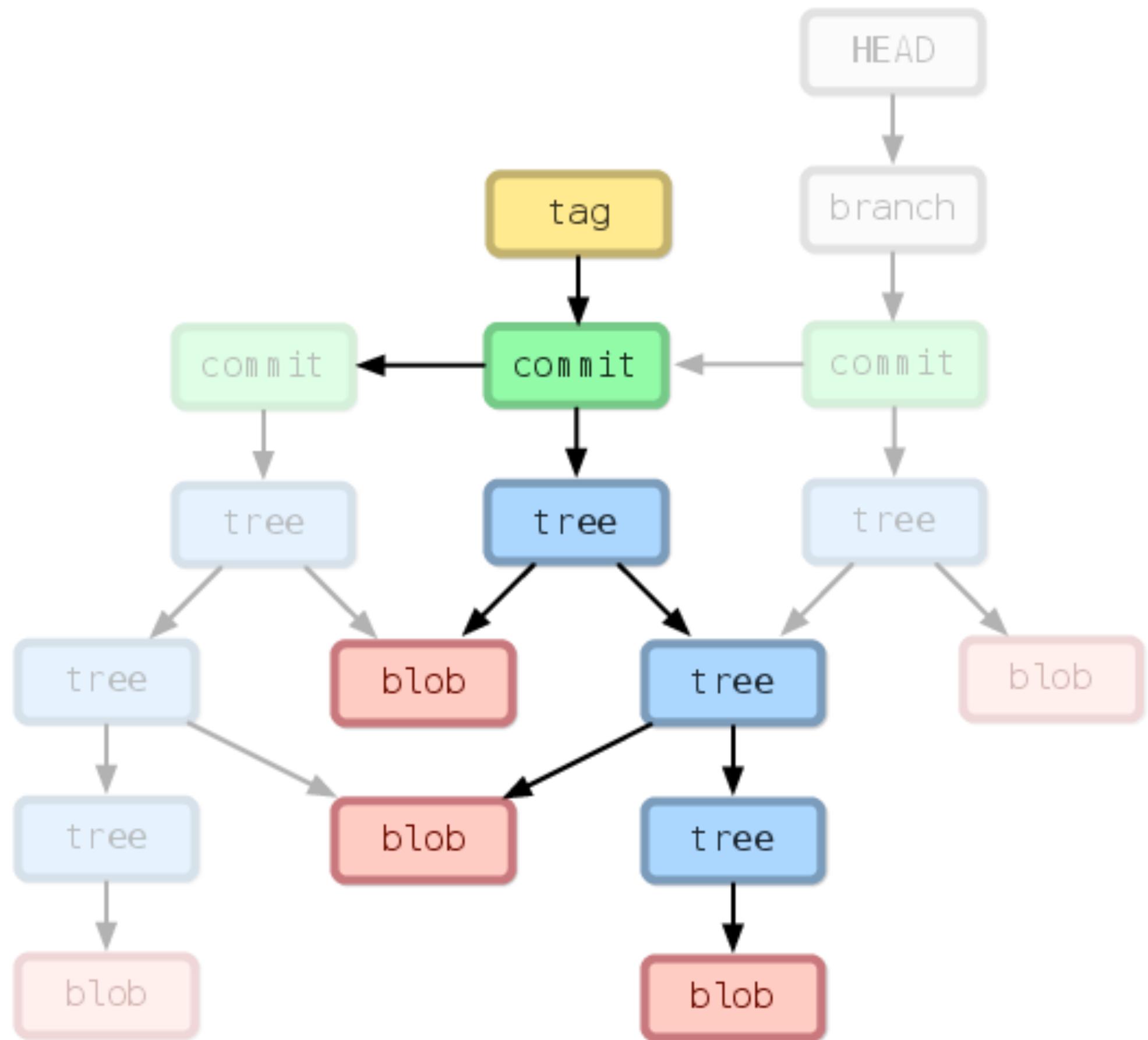
# Scenario



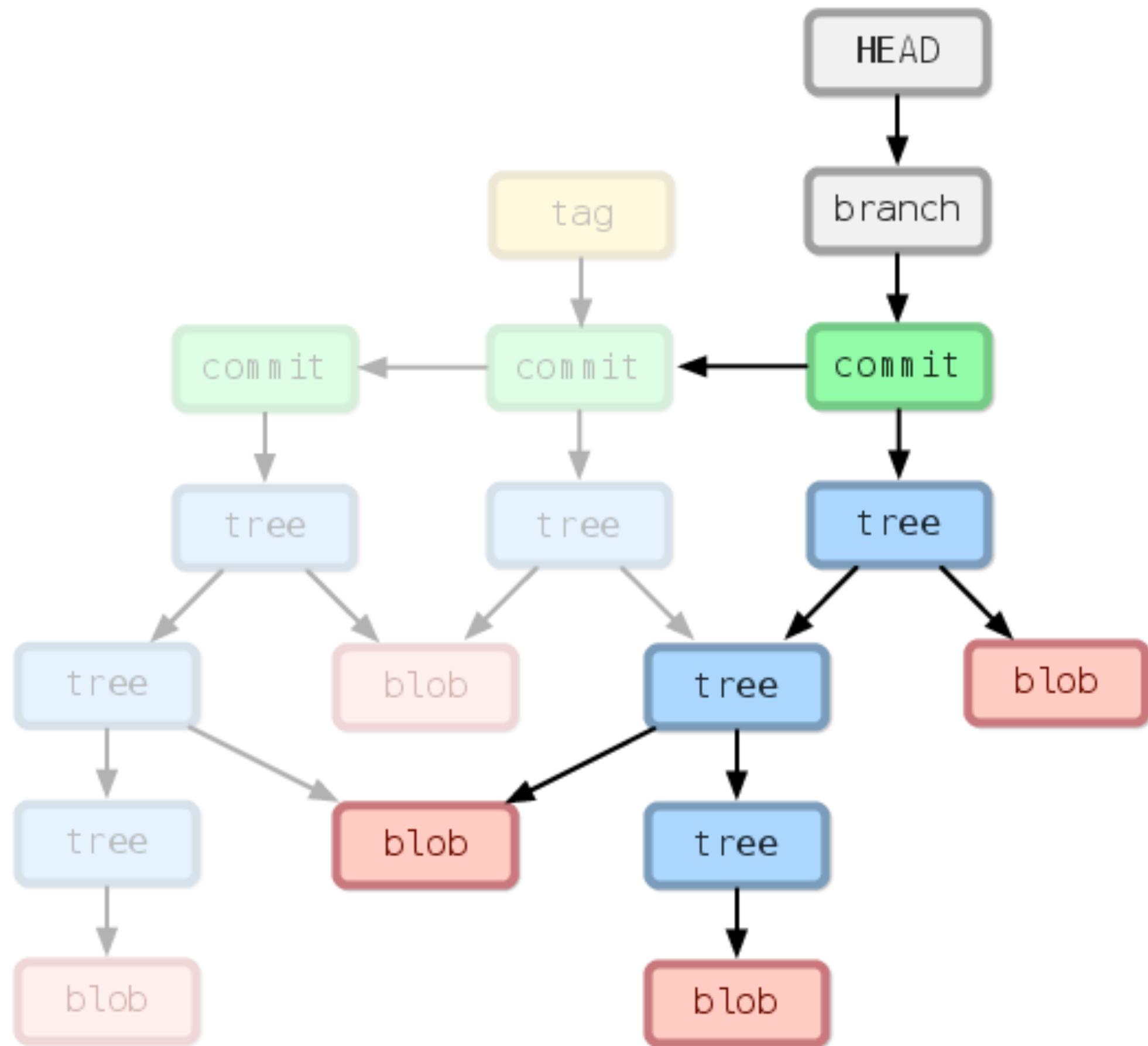
# Scenario



# Scenario



# Scenario



# Git Directory

**Configuration File**

**Hooks**

**Object Database**

**References**

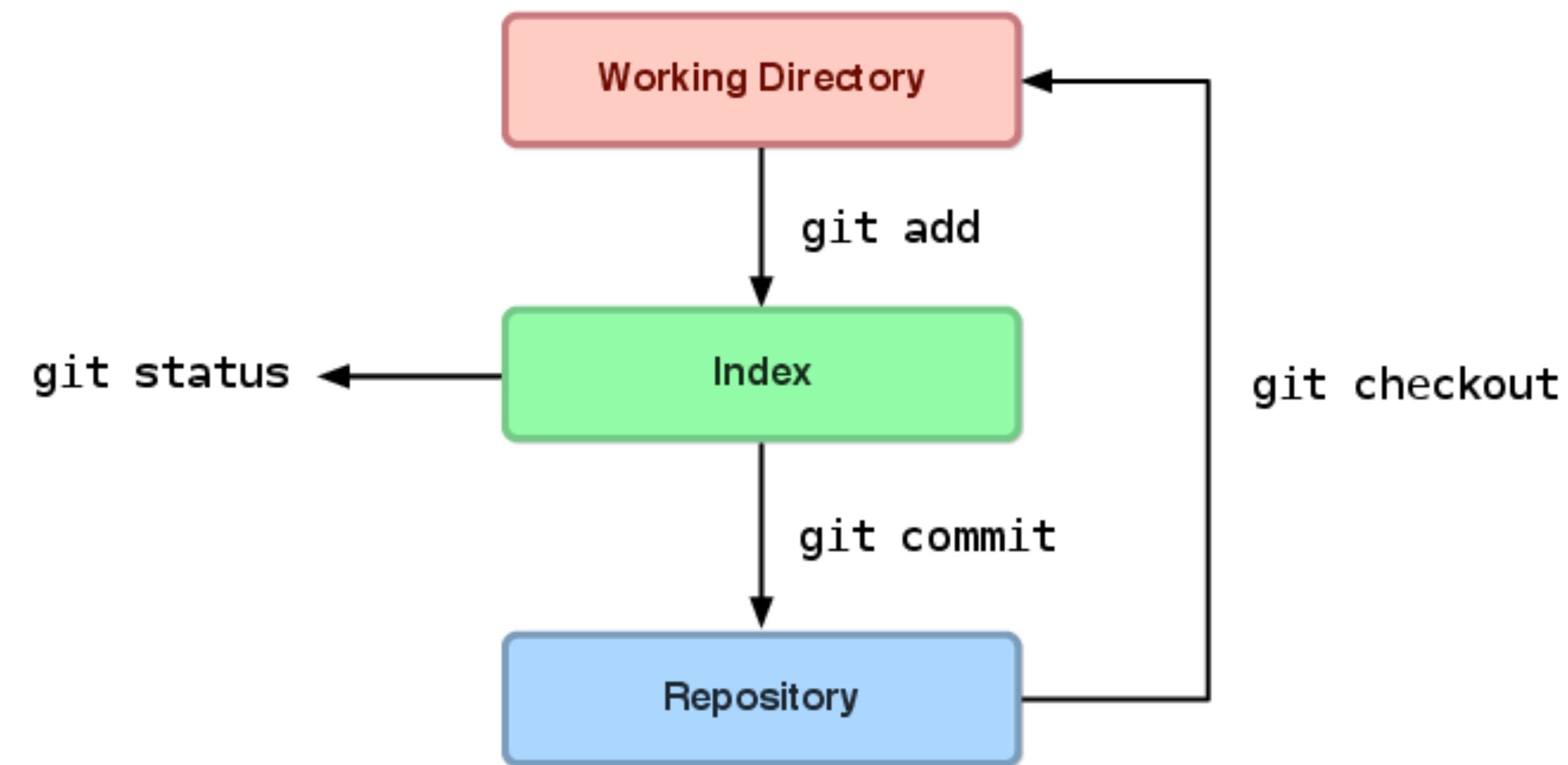
**Index**

# Index

# Index

**== Staging Area**

# Index



# Index FTW

**No Need To Commit All At Once**

**Pick (Stage) Logical Units to Commit**

**Helps You Review Your Changes**

**Lets You Write Your History Cleanly**

# Git Started

# An introduction to modern tools for collaborative science

## Exercises - TBD before lecture 2

Luca Heltai <[luca.heltai@sissa.it](mailto:luca.heltai@sissa.it)>

Mathematical Analysis, Modeling, and Applications ([math.sissa.it](http://math.sissa.it))

Theoretical and Scientific Data Science ([datascience.sissa.it](http://datascience.sissa.it))

# New Project

**Create and initialize the Git Directory (.git)**

```
$ git init
```

# Existing Project

**Create and pull down remote repository.**

```
$ git clone git://github.com/pbhogan/Archivist.git
```

# .gitignore

## Specify files which will be ignored by Git

```
$ cat .gitignore
.svn
.DS_Store
build
*.pbxuser
*.perspective
*.perspectivev3
*.modelv3
*.mode2v3
*.xcuserstate
*.xcworkspace
xcuserdata
```

# Staging

**Stage files to the index.**

```
$ git add .
```

# Committing

**Create a commit tagged with a message.**

```
$ git commit -m "My first commit!"
```

# Branching & Merging

# Branching & Merging

**Create new branch (from current branch)**

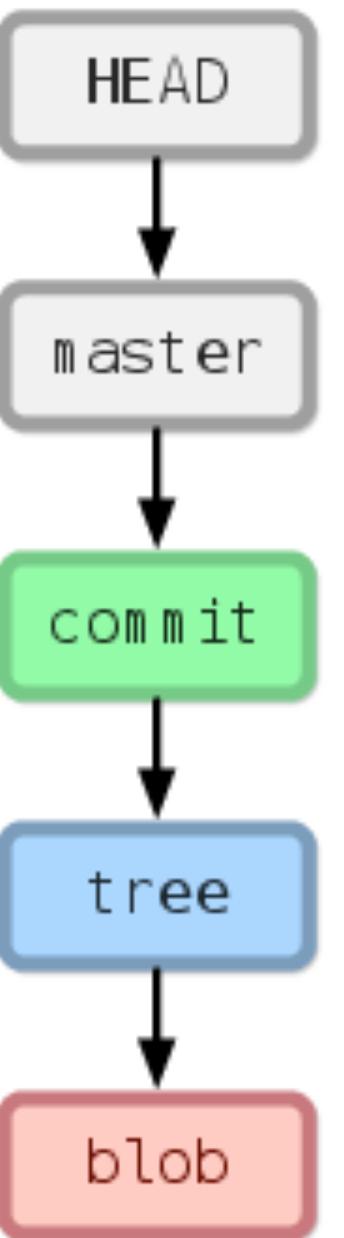
```
$ git branch <name>
```

# Branching & Merging

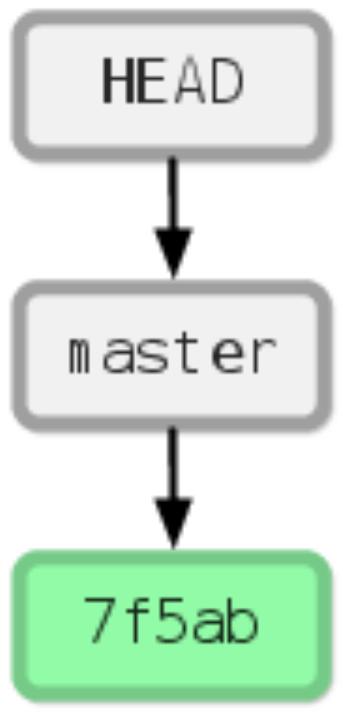
**Switch to branch (overwrites Working Dir!)**

```
$ git checkout <name>
```

```
$ git commit -m "First commit!"
```

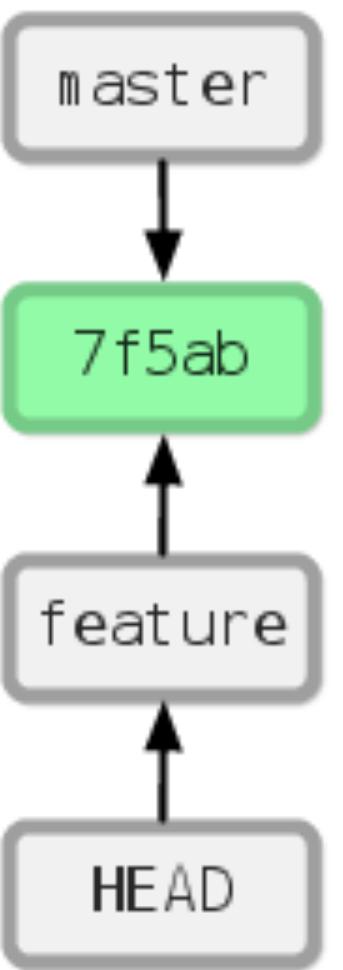


```
$ git commit -m "First commit!"
```

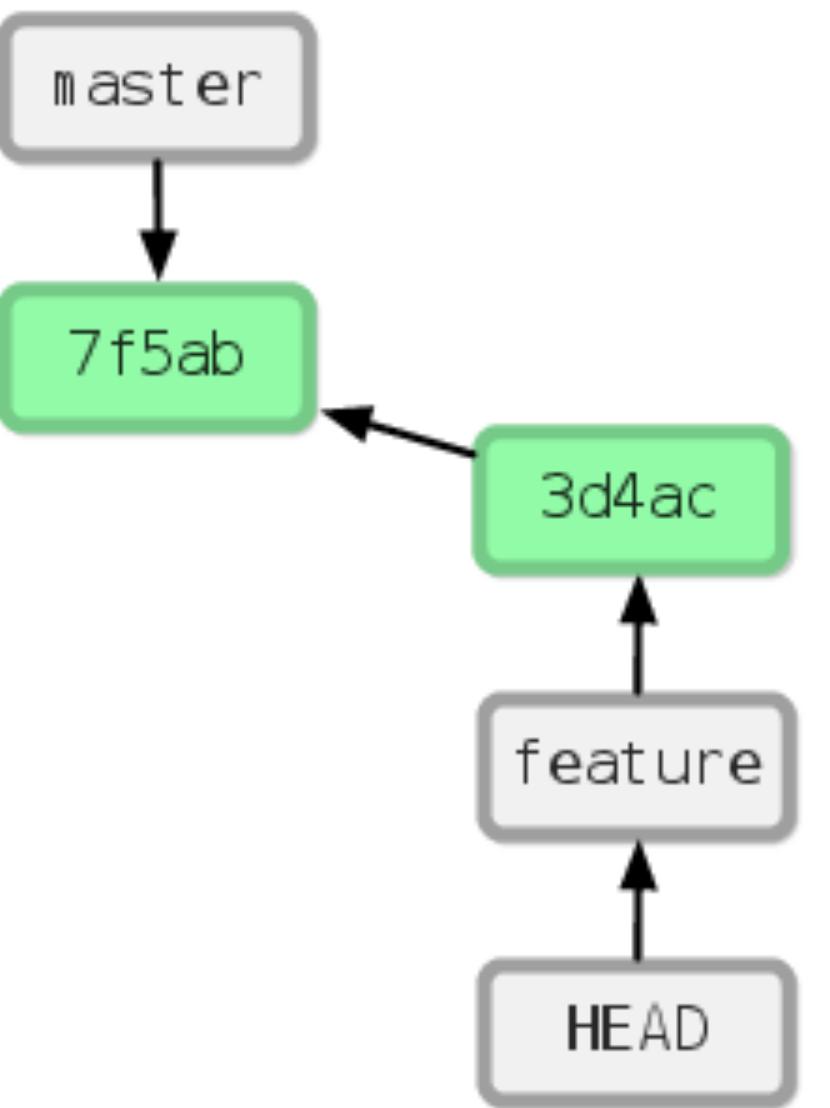


← **represents commit + subtree**

```
$ git branch feature  
$ git checkout feature
```

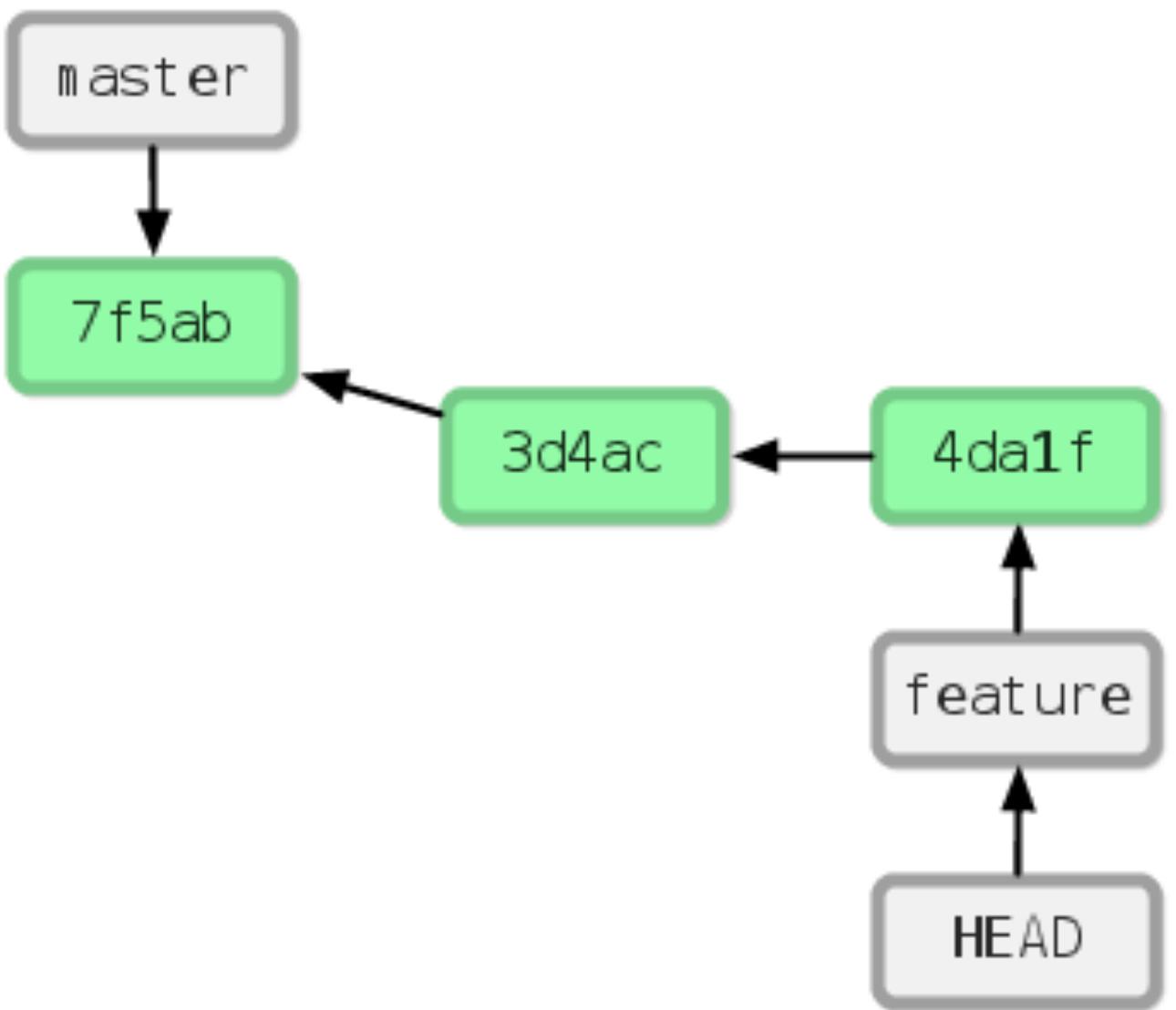


```
$ git add feat.c  
$ git commit -m "Added feat.c"
```



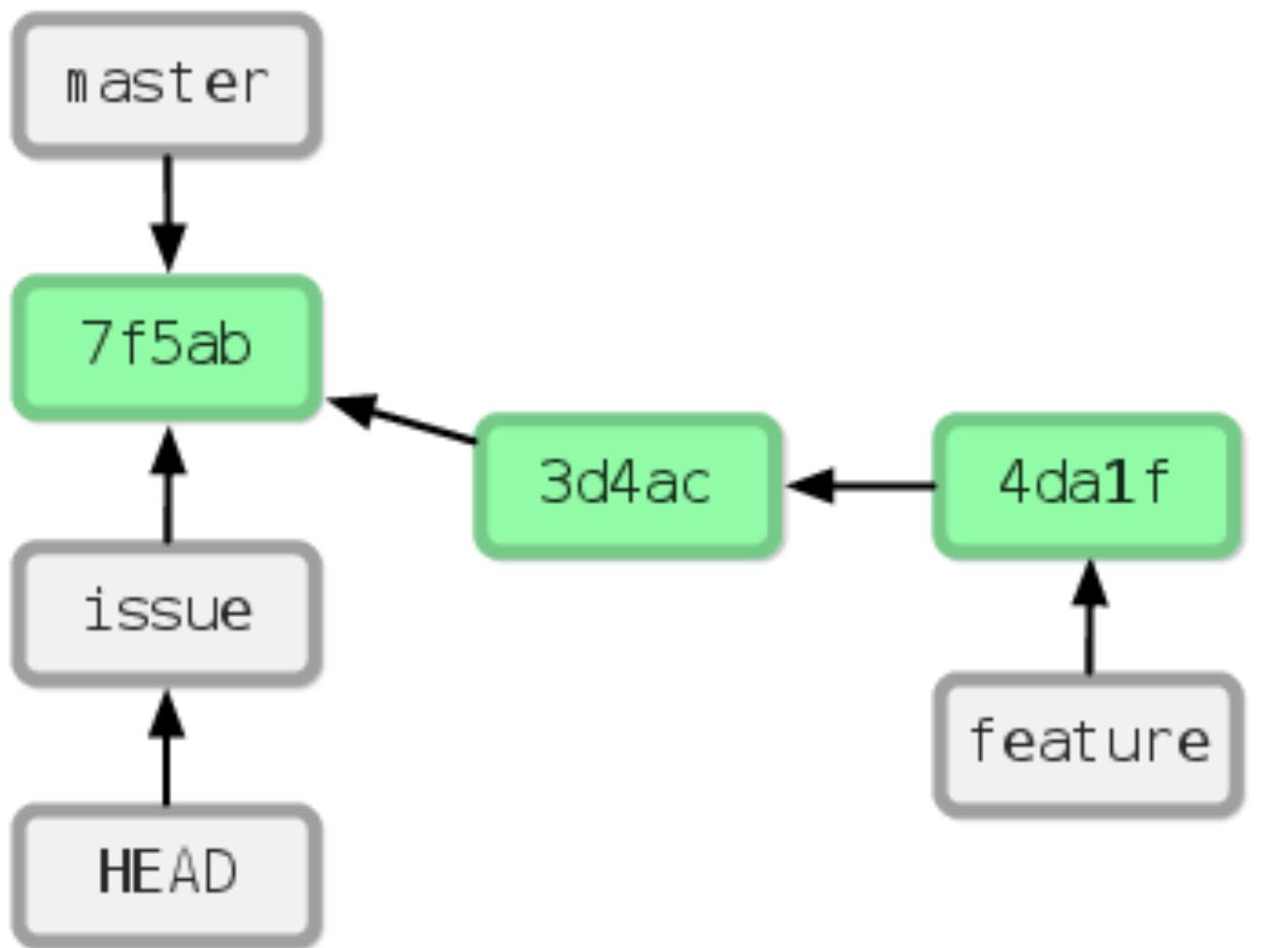
```
$ git commit -a -m "Updated feat.c"
```

feat.c already tracked so -a automatically stages.

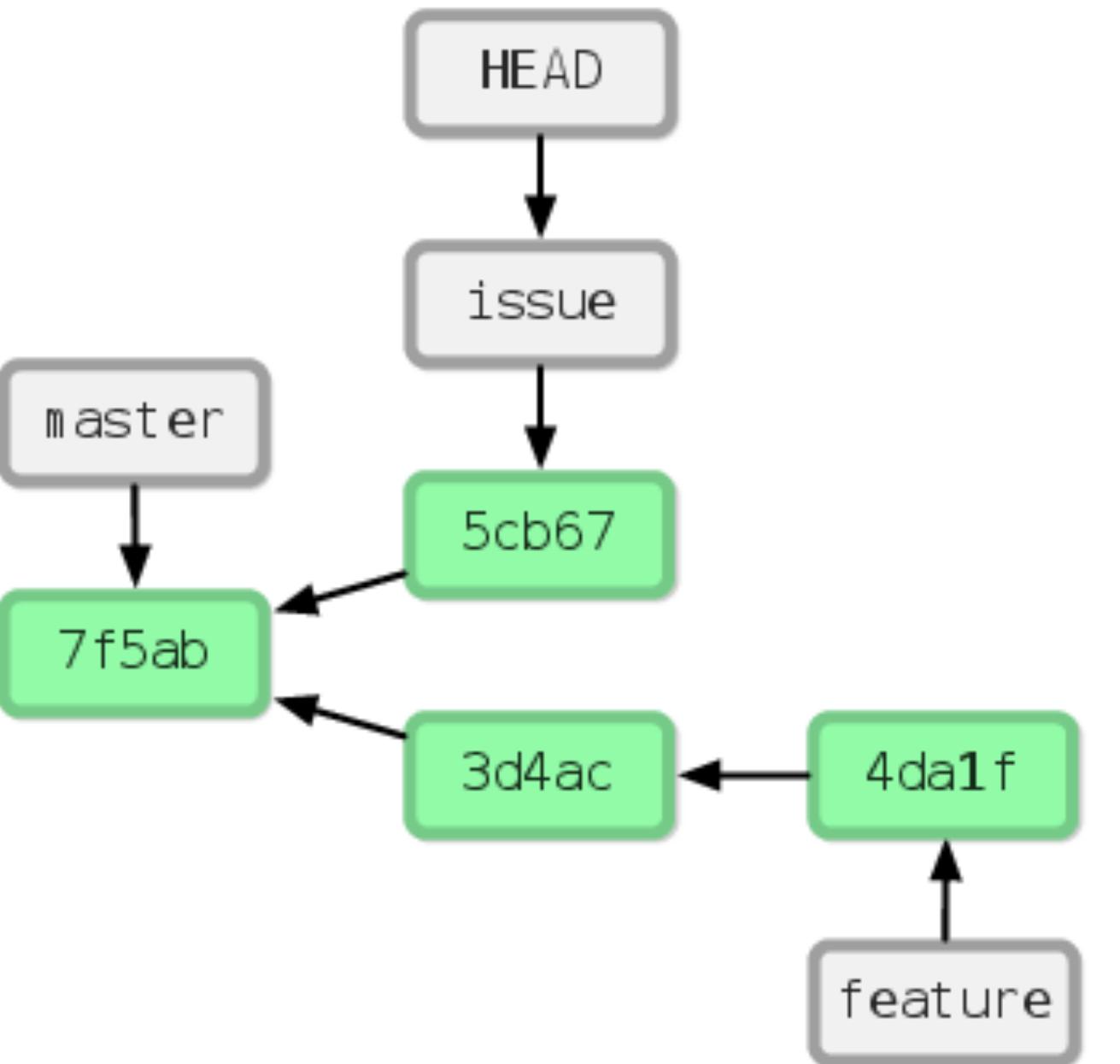


```
$ git checkout -b issue master
```

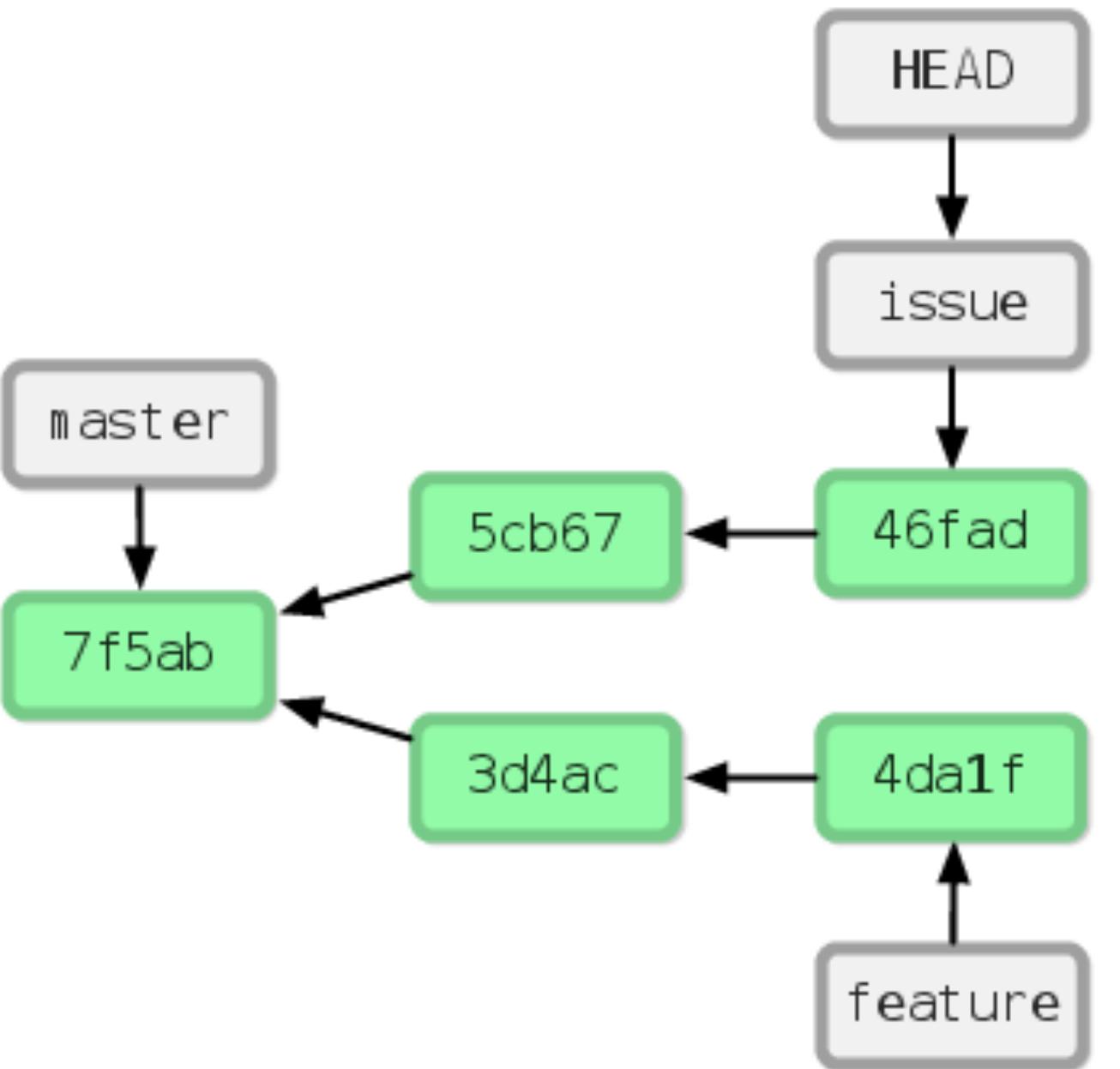
```
git checkout master  
git branch issue  
git checkout issue
```



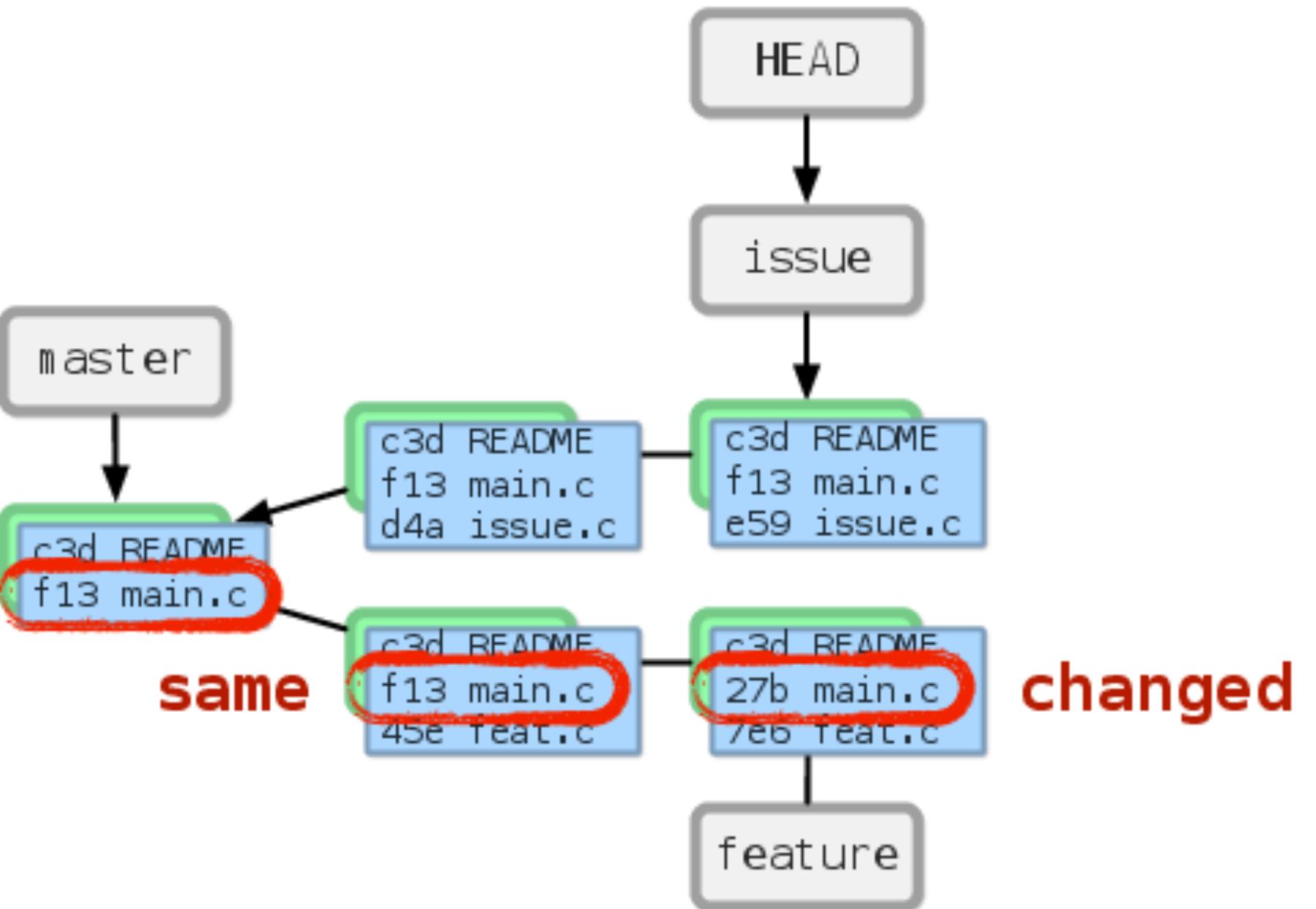
```
$ git add issue.c  
$ git commit -m "Added issue.c"
```



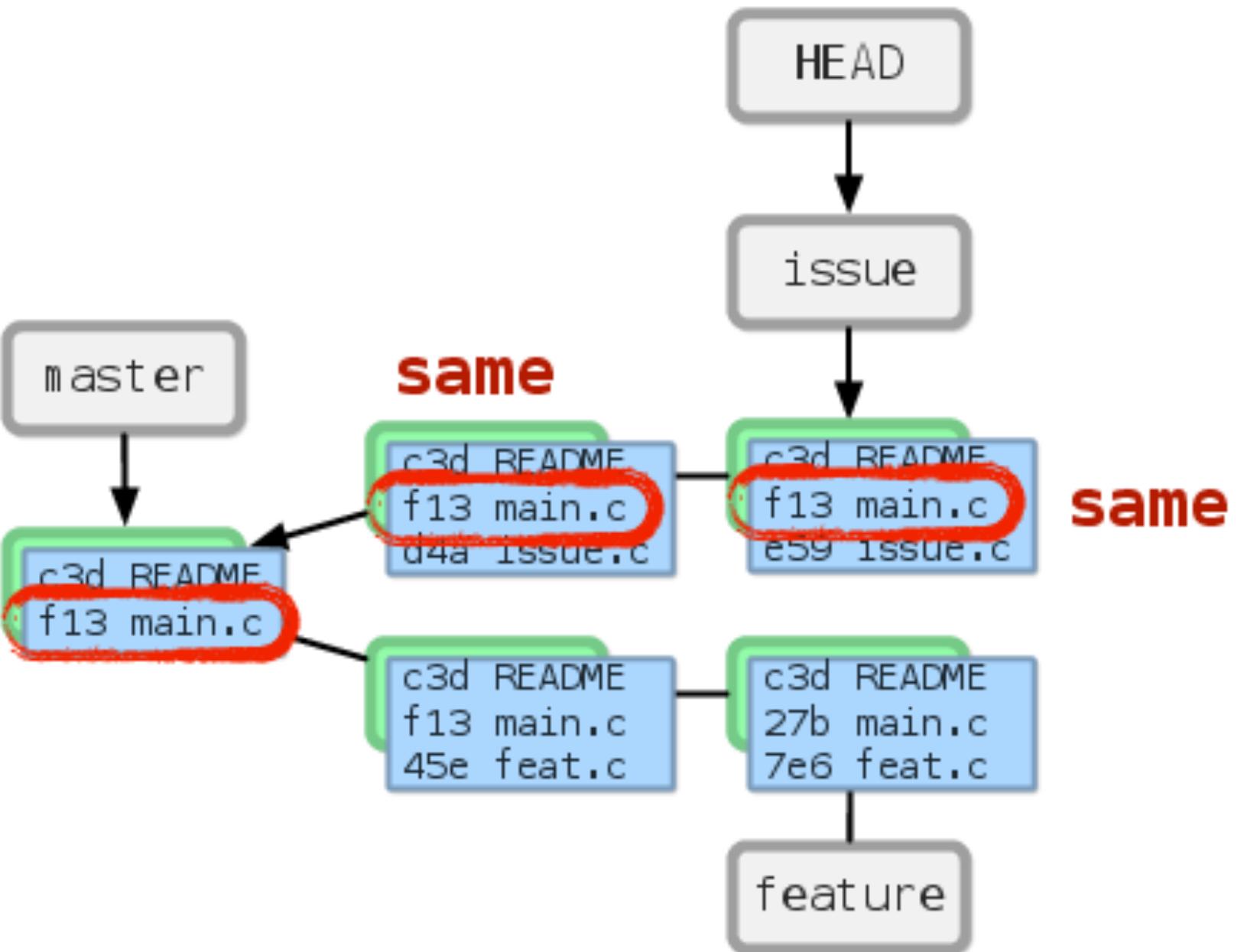
```
$ git commit -a -m "Updated issue.c"
```



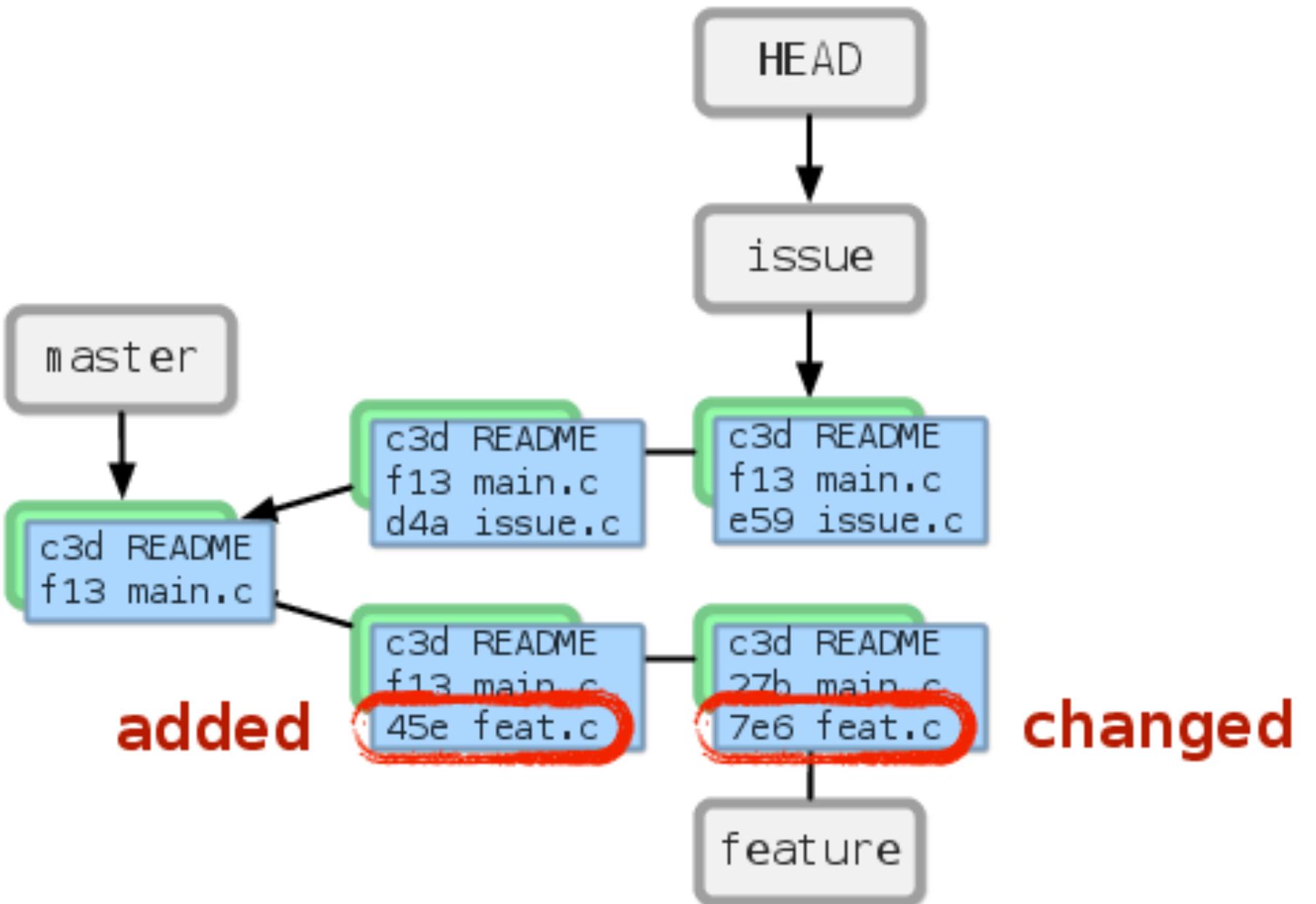
```
$ git log --stat
```



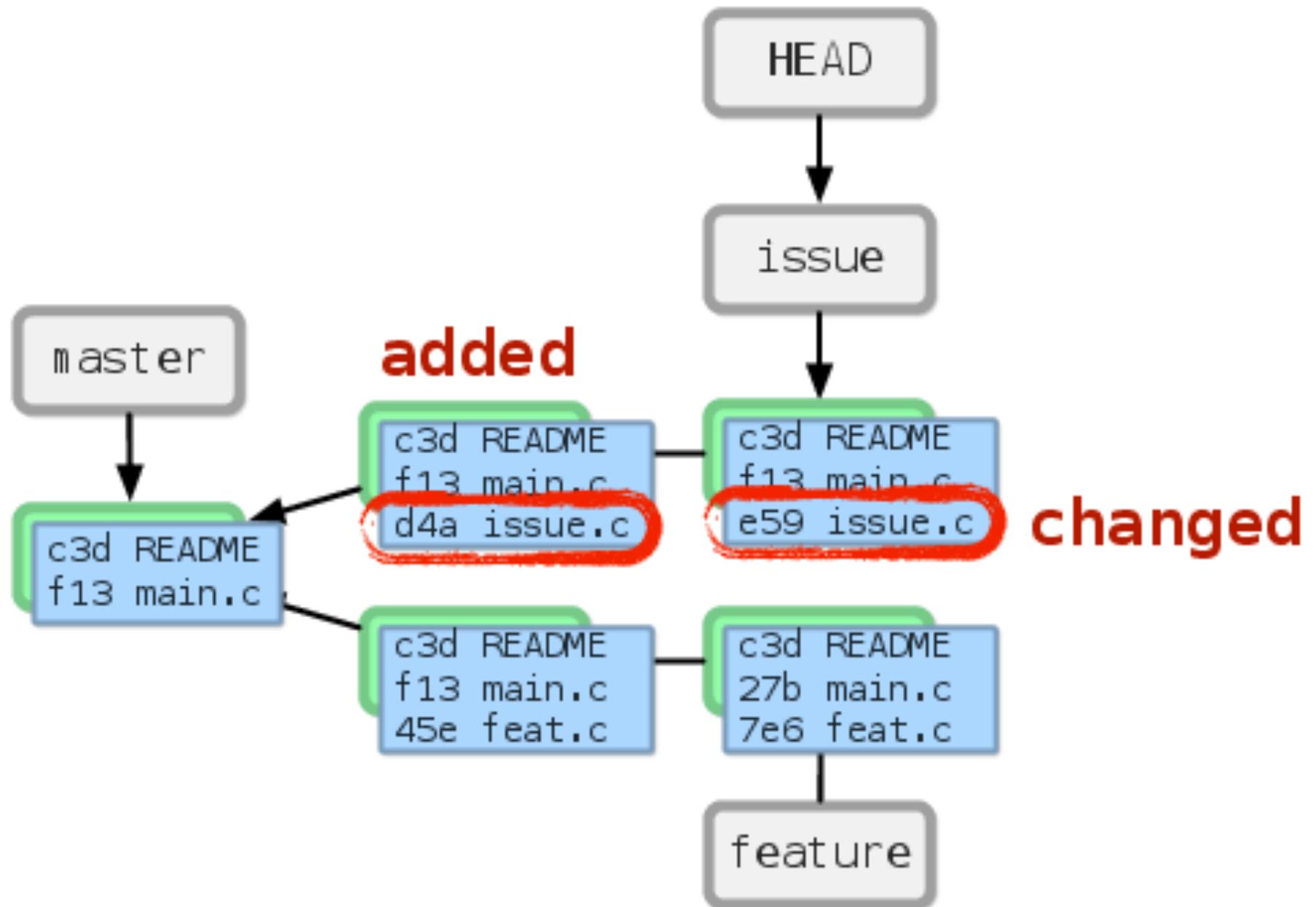
```
$ git log --stat
```

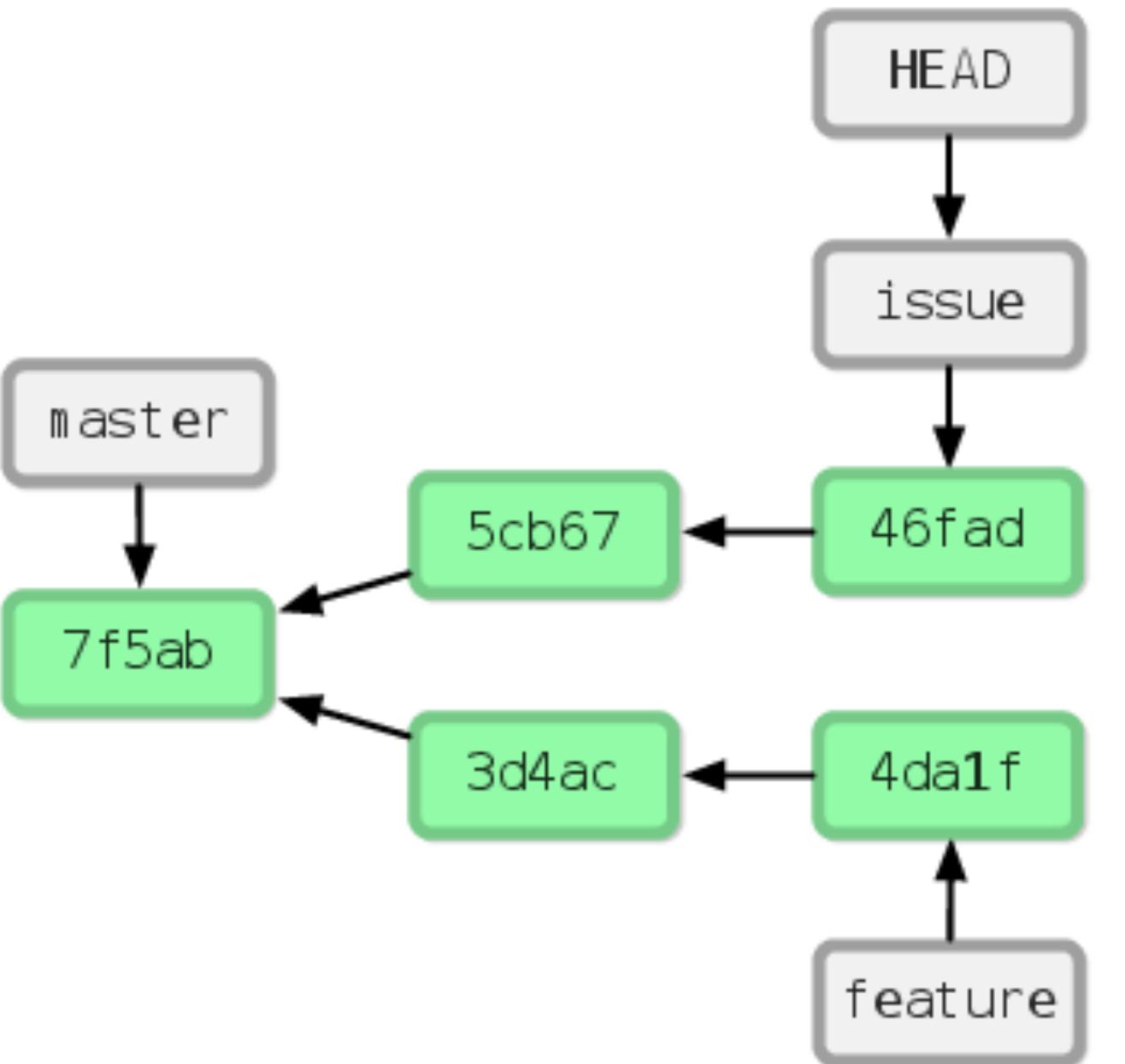


```
$ git log --stat
```

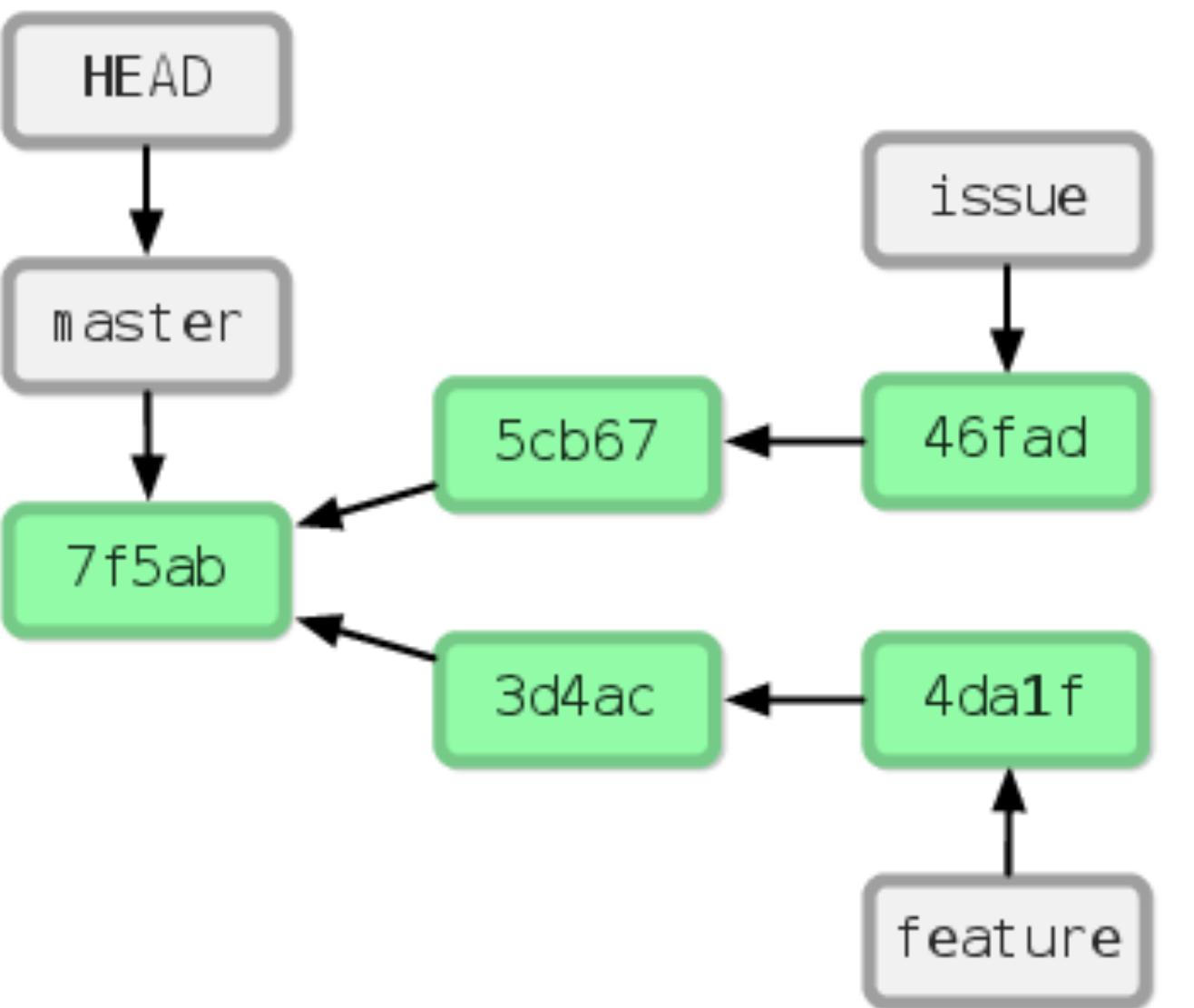


```
$ git log --stat
```

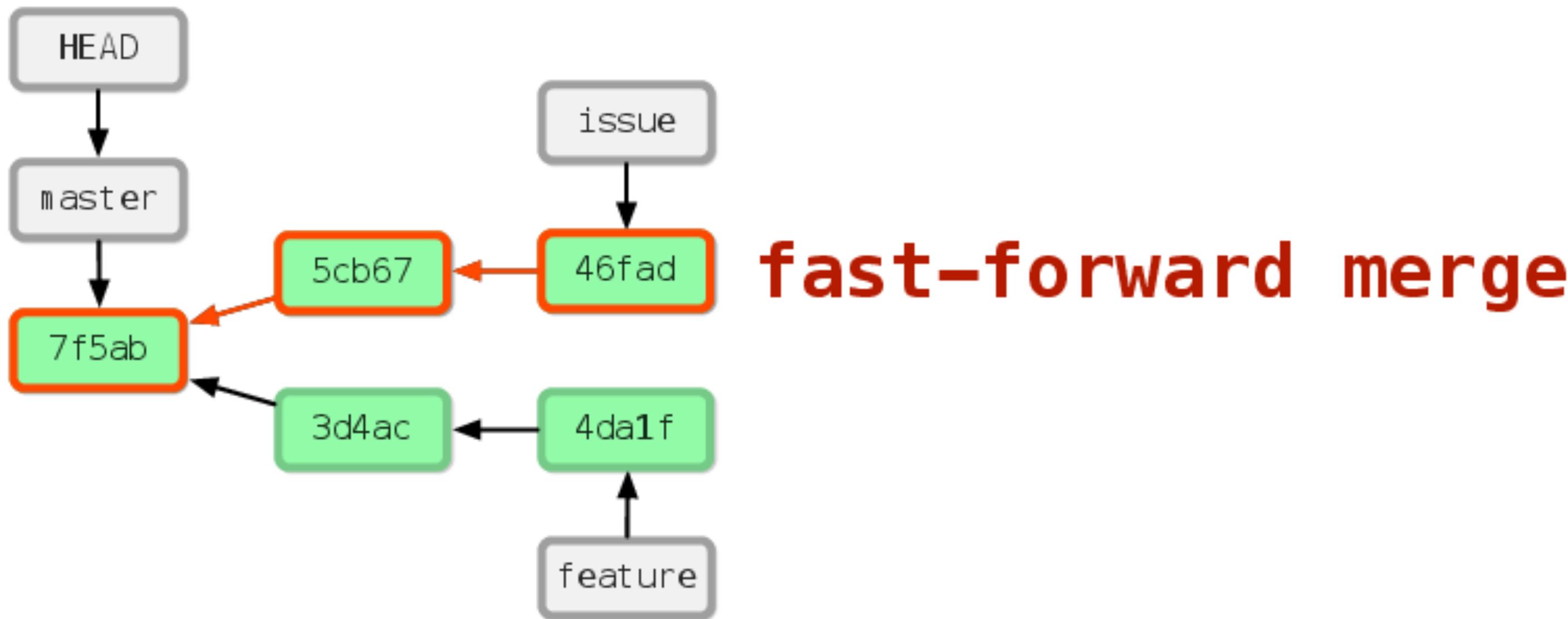




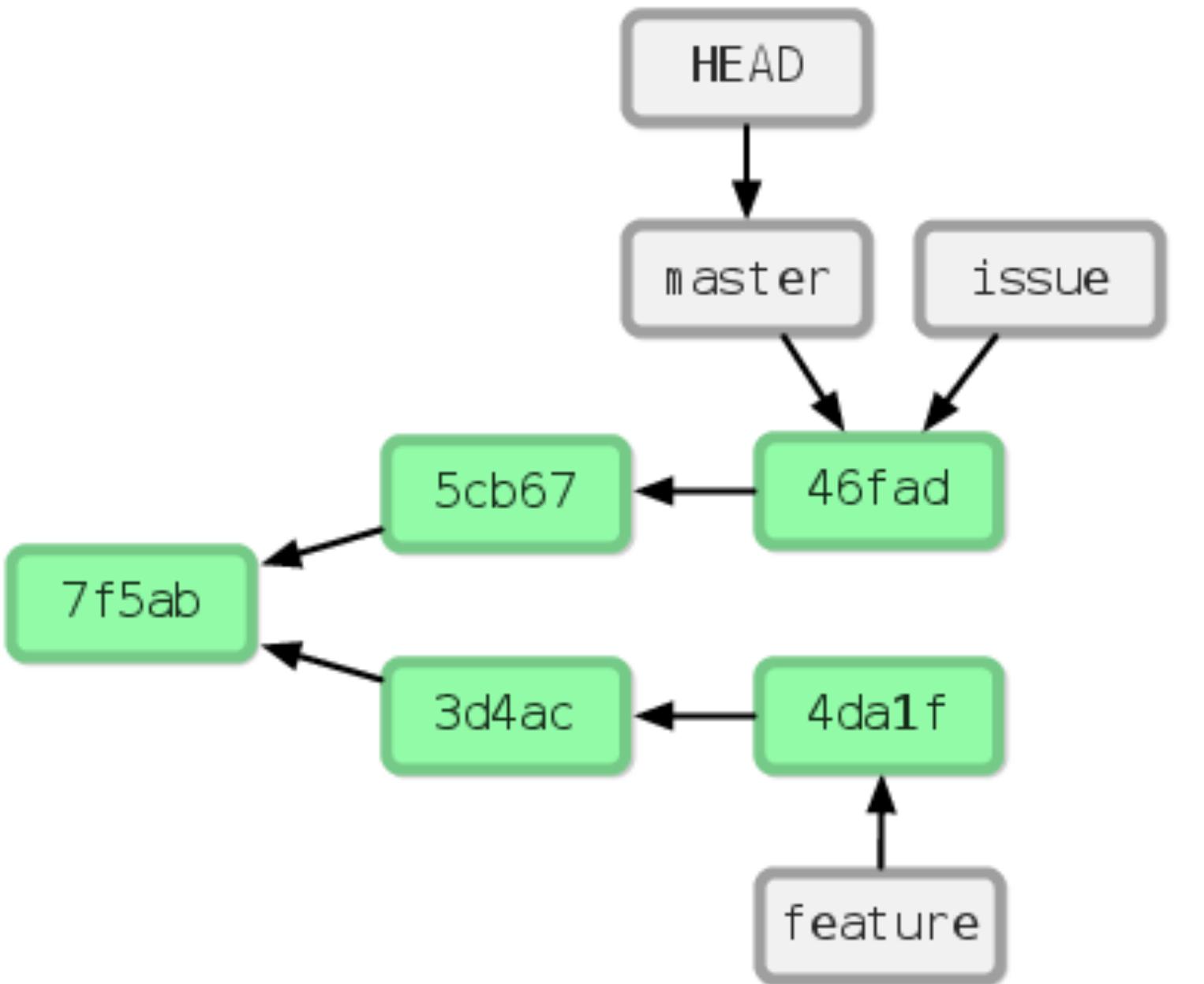
```
$ git checkout master
```



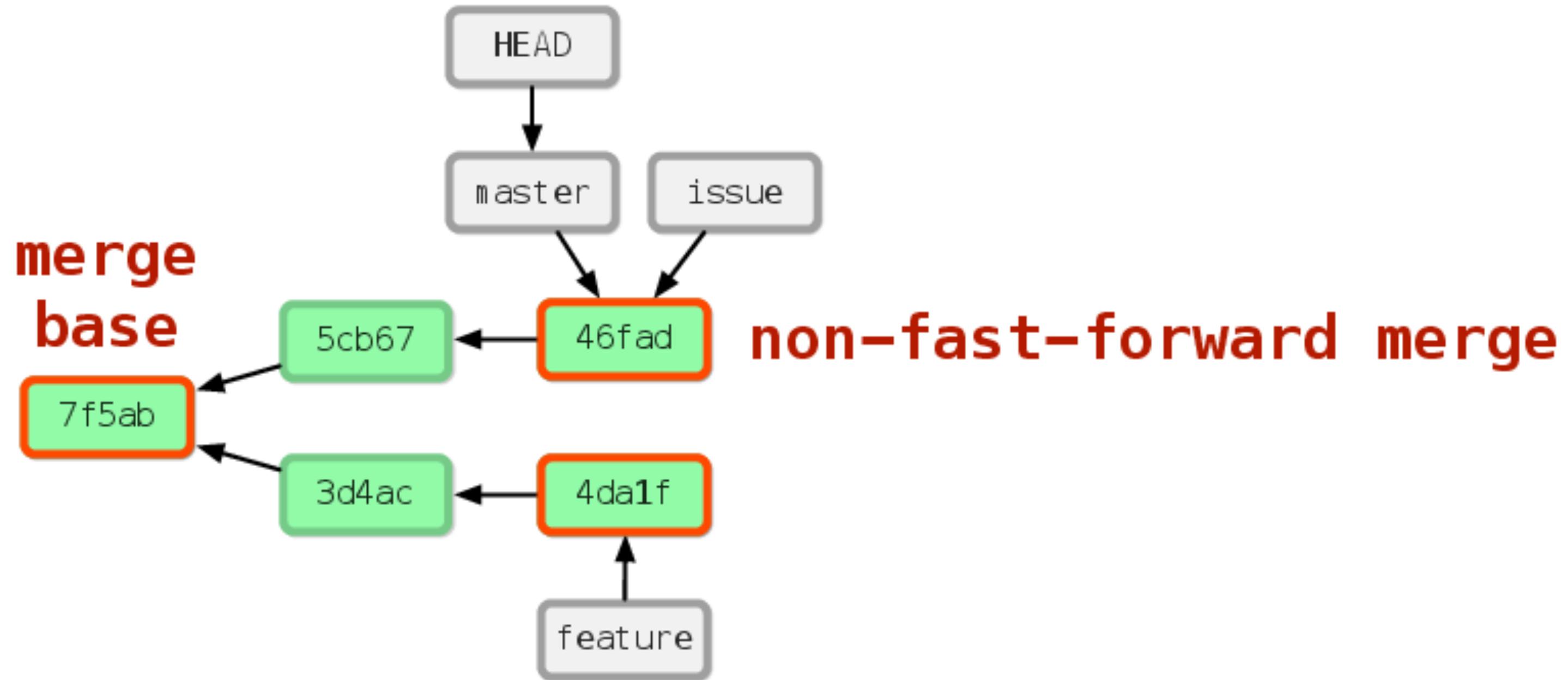
```
$ git merge issue
```



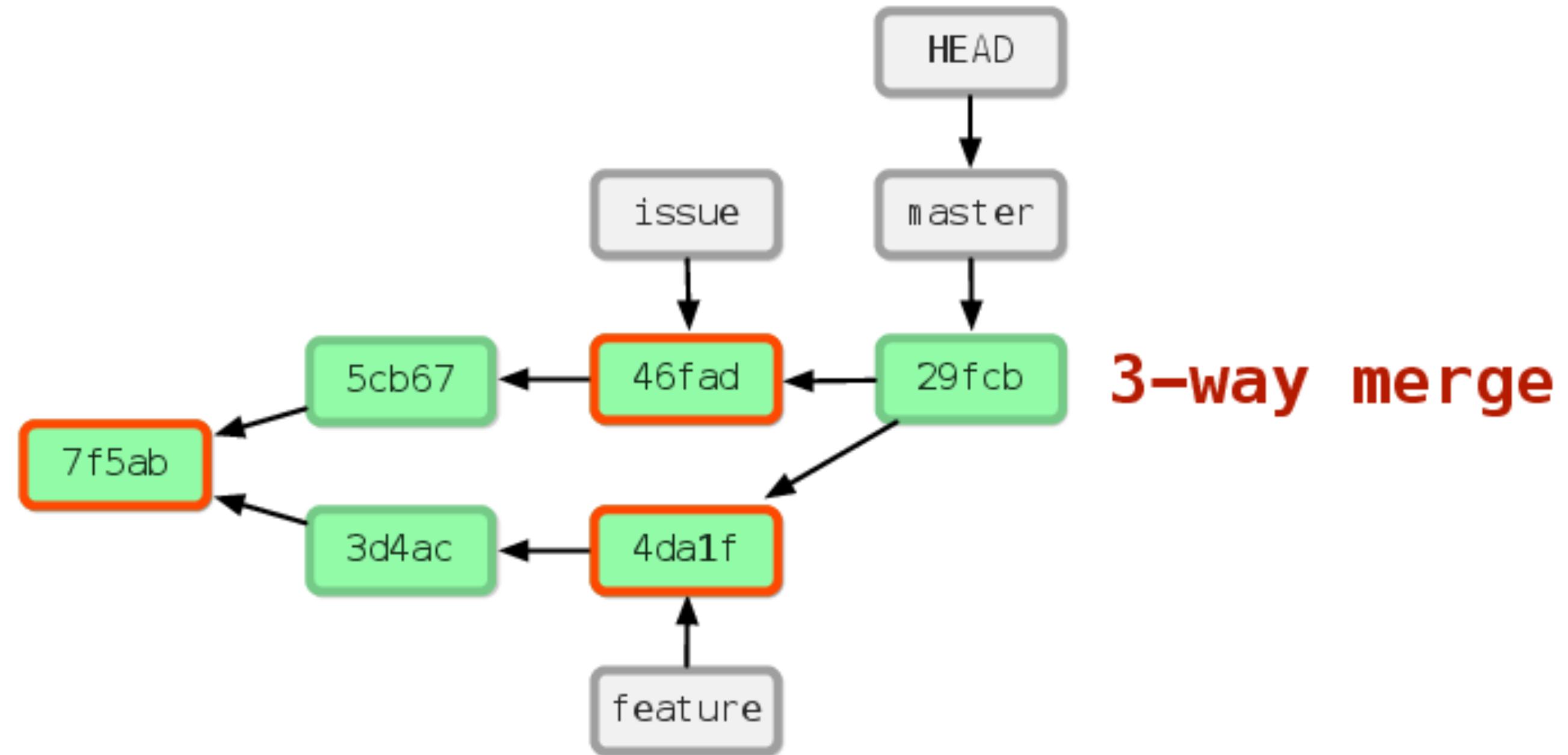
```
$ git merge issue
```



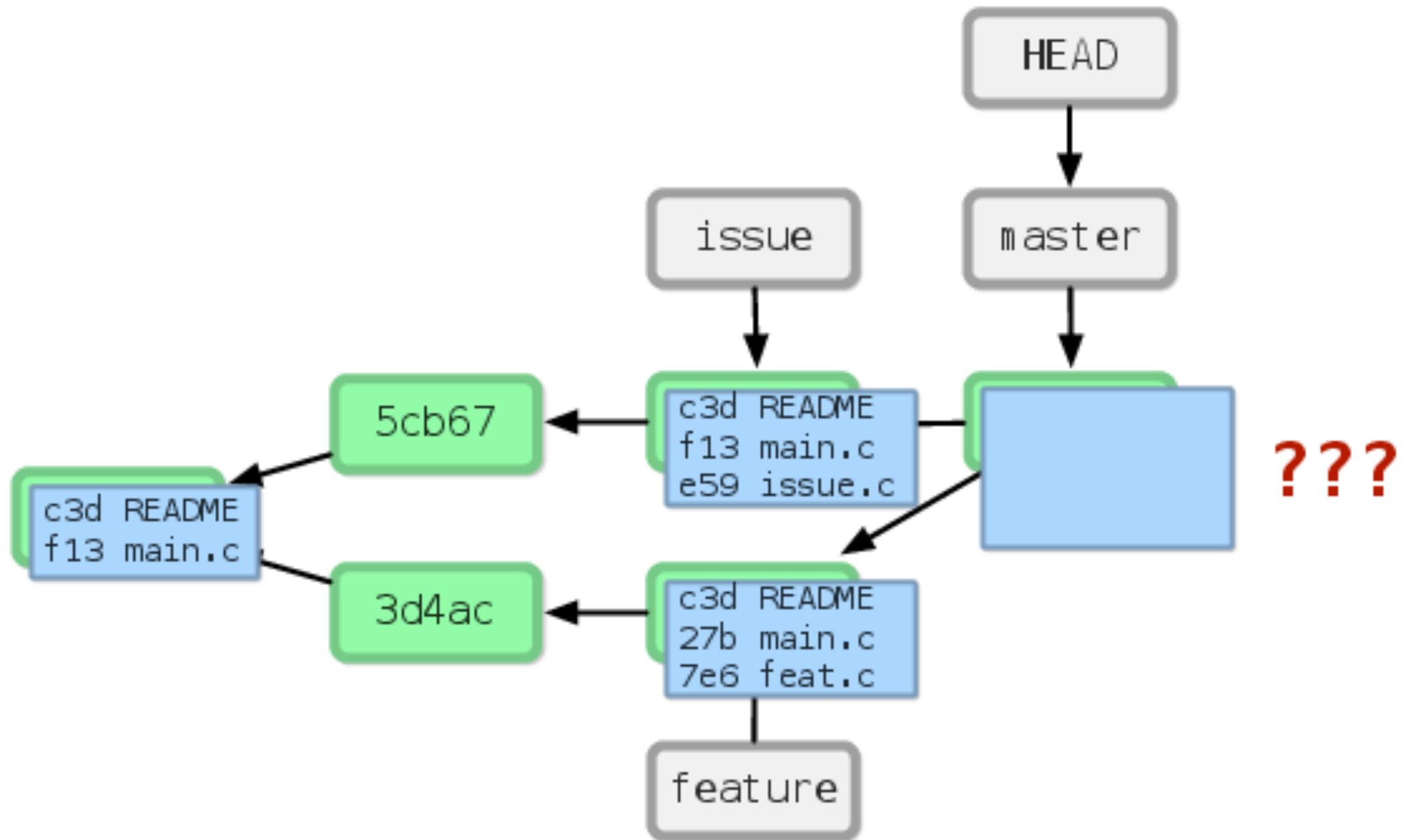
```
$ git merge feature
```



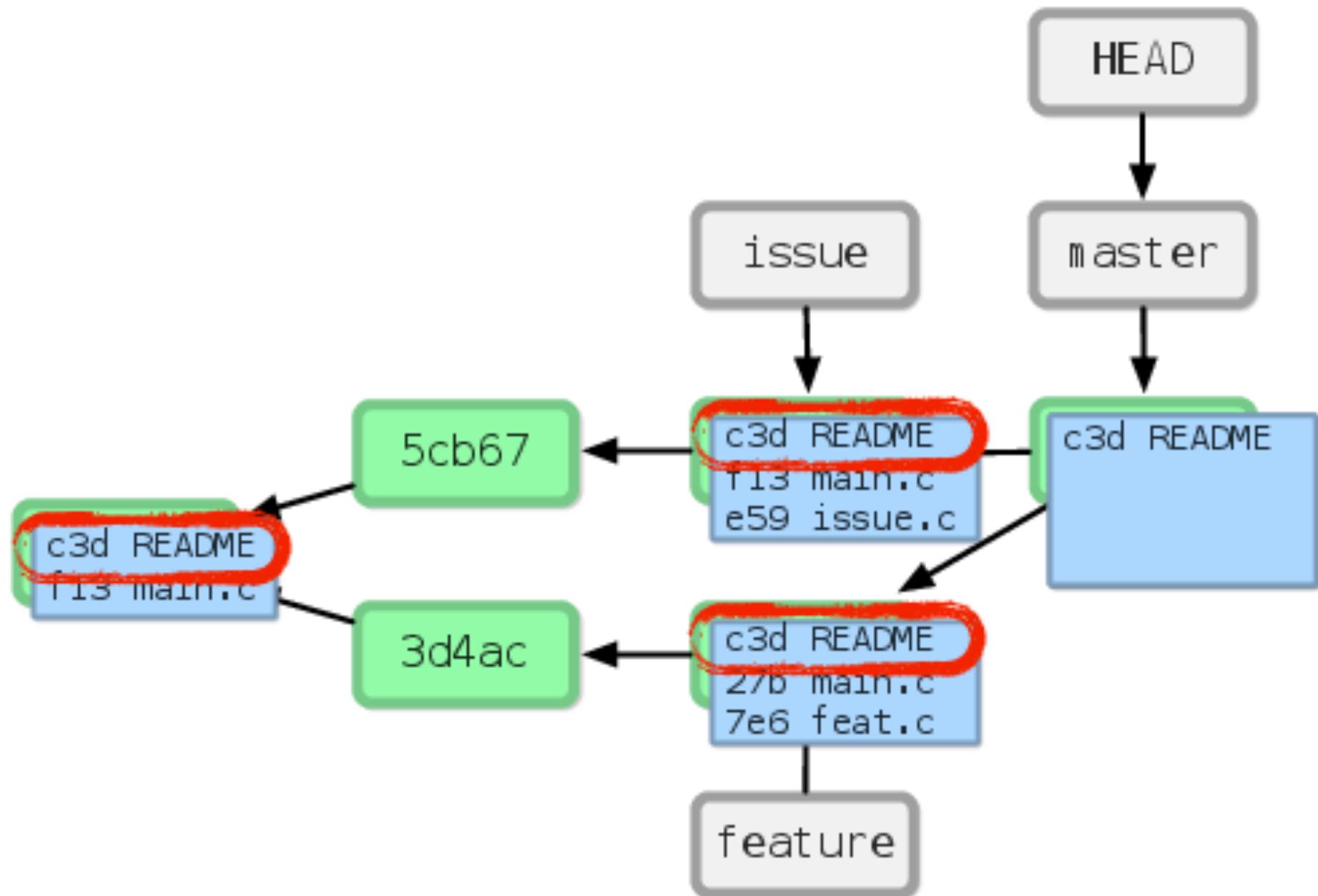
```
$ git merge feature
```



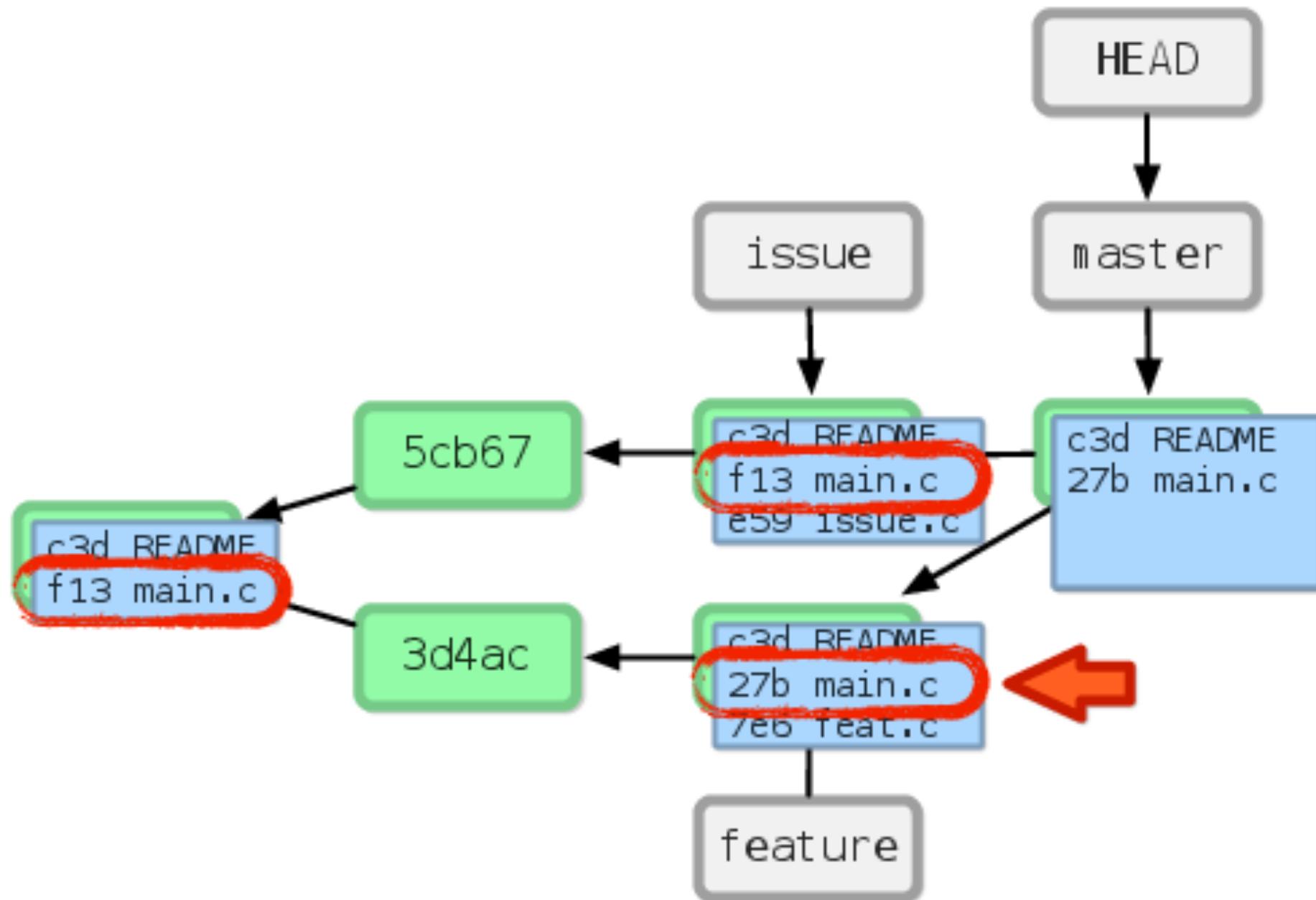
```
$ git merge feature
```



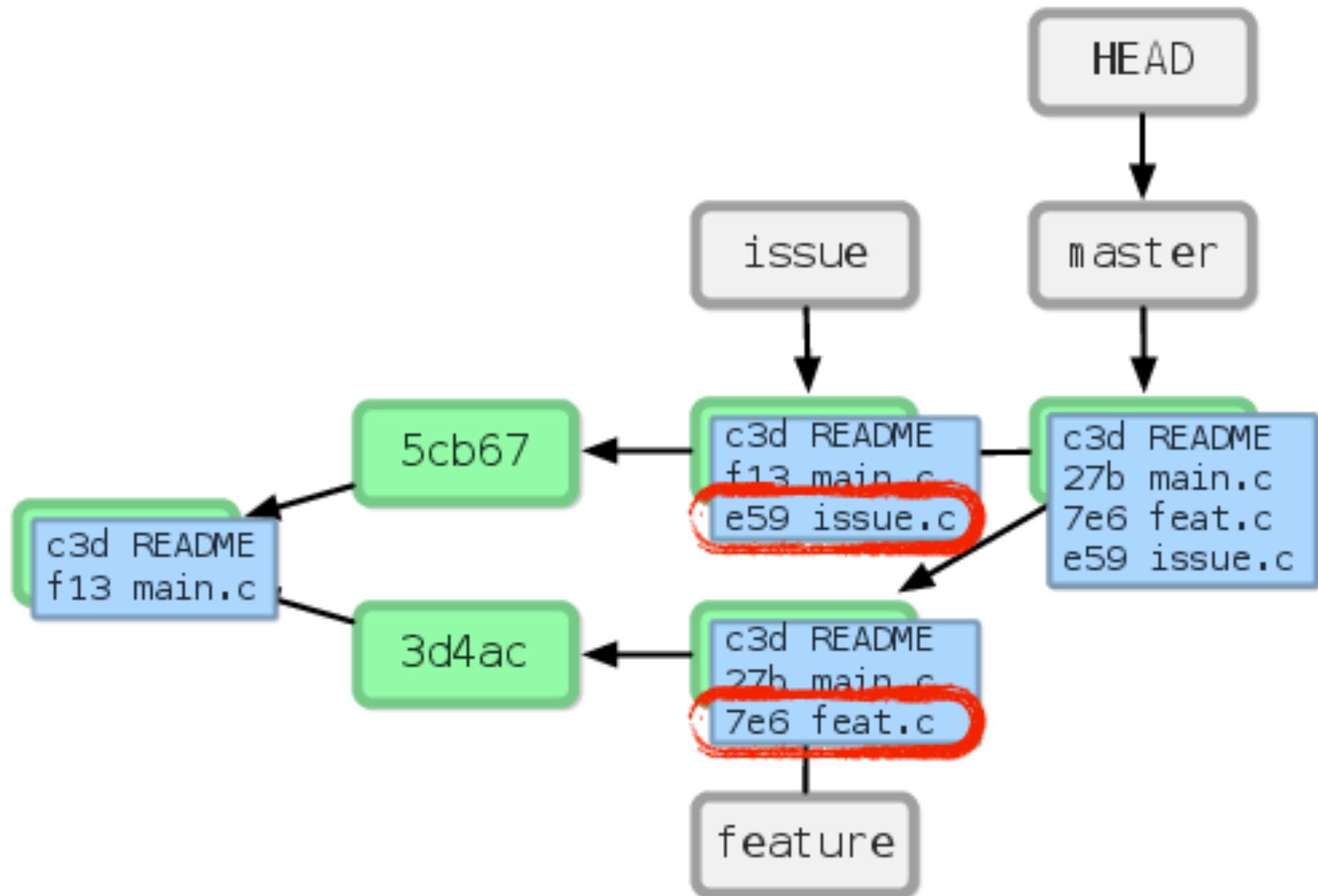
```
$ git merge feature
```



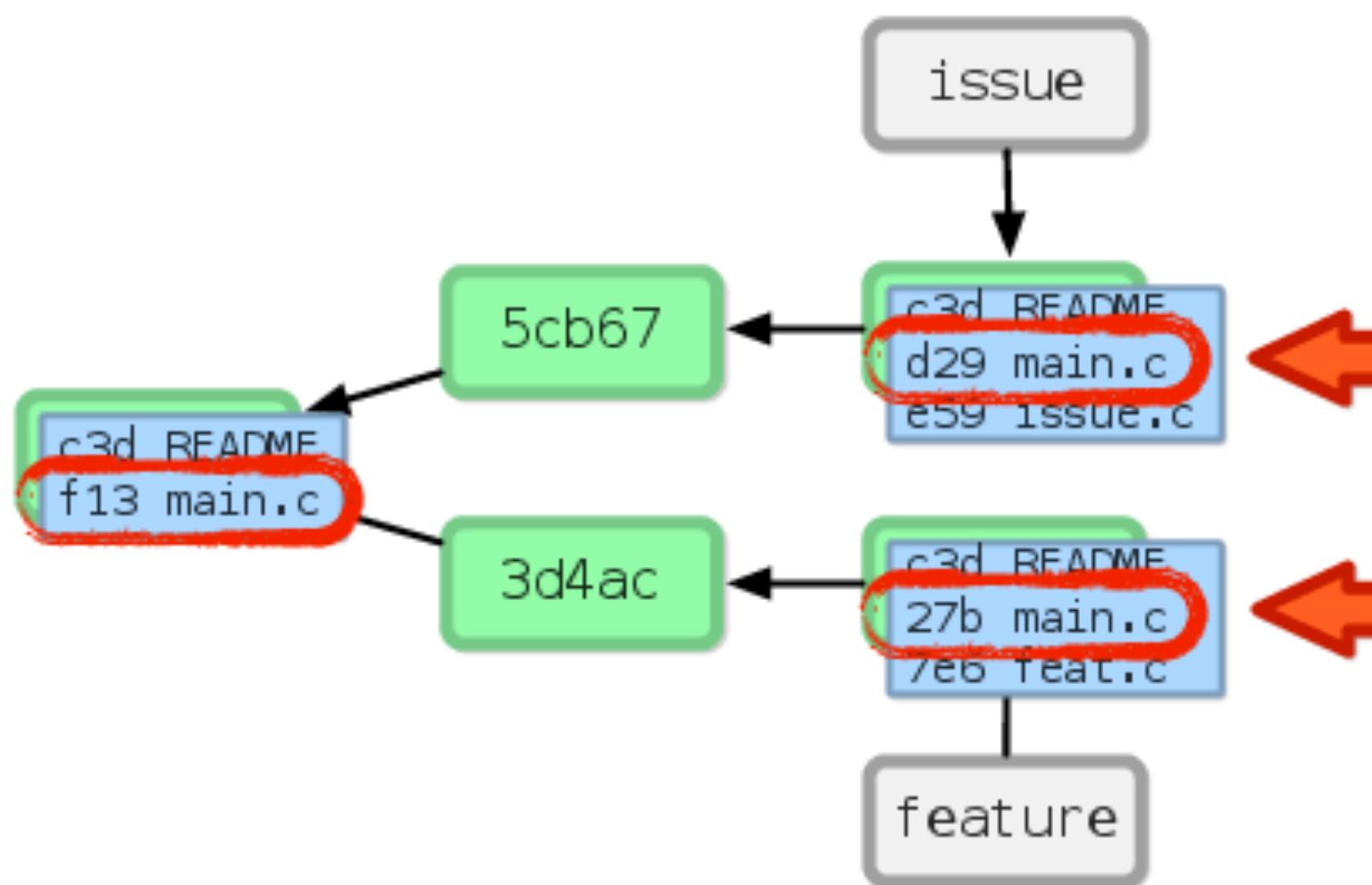
```
$ git merge feature
```



```
$ git merge feature
```



```
$ git merge feature
```



What if `main.c`  
MERGEs CONFLICTED!  
in both branches?

# Merge Conflict

```
$ git merge feature
```

Auto-merging main.c

**CONFLICT (content): Merge conflict in main.c**

Automatic merge failed; fix conflicts and then  
commit the result.

# Merge Conflict

```
$ git merge feature

# On branch master
# Unmerged paths:
#   (use "git add/rm <file>..." as appropriate
#    to mark resolution)
#
# both modified:      main.c
#
no changes added to commit (use "git add" and/
or "git commit -a")
```

# Merge Conflict

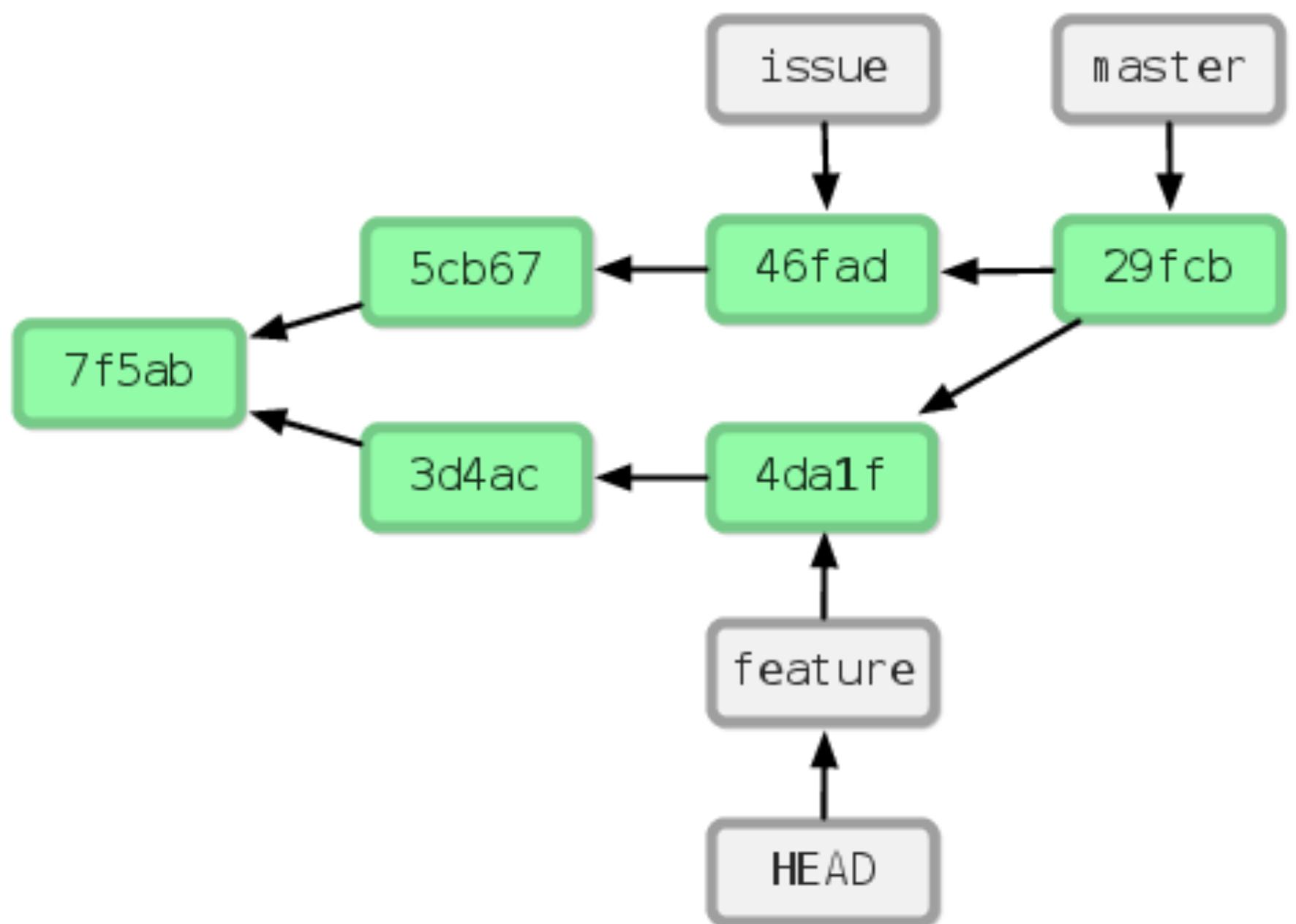
```
$ cat main.c

int main (int argc, char const *argv[])
{
<<<<< HEAD
    printf( "Hola World!" );
=====
    printf("Hello World!");
>>>>> feature
    return 0;
}
```

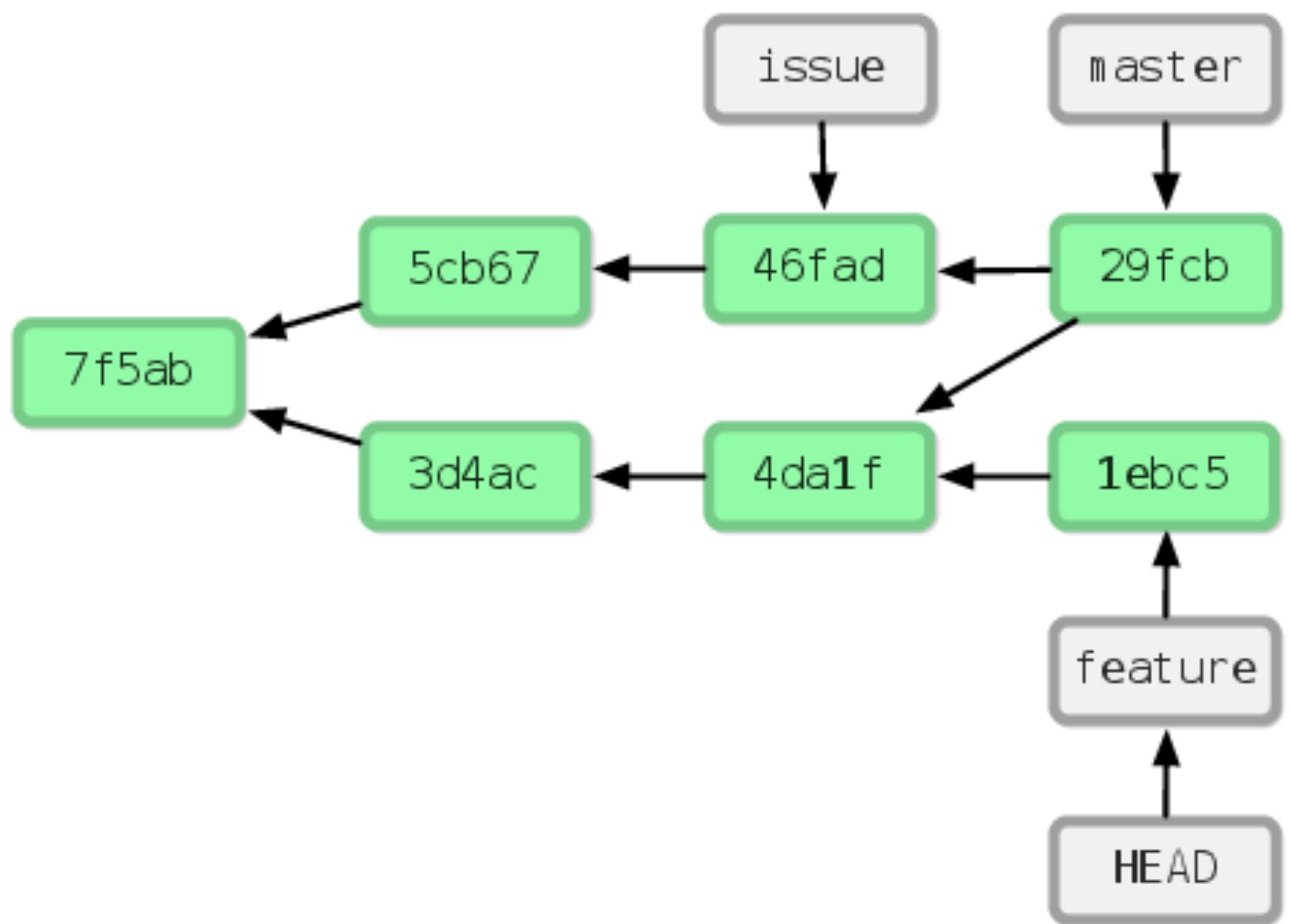
# Merge Conflict

```
$ git mergetool
$ git add main.c
$ git commit -m "Merged feature"
```

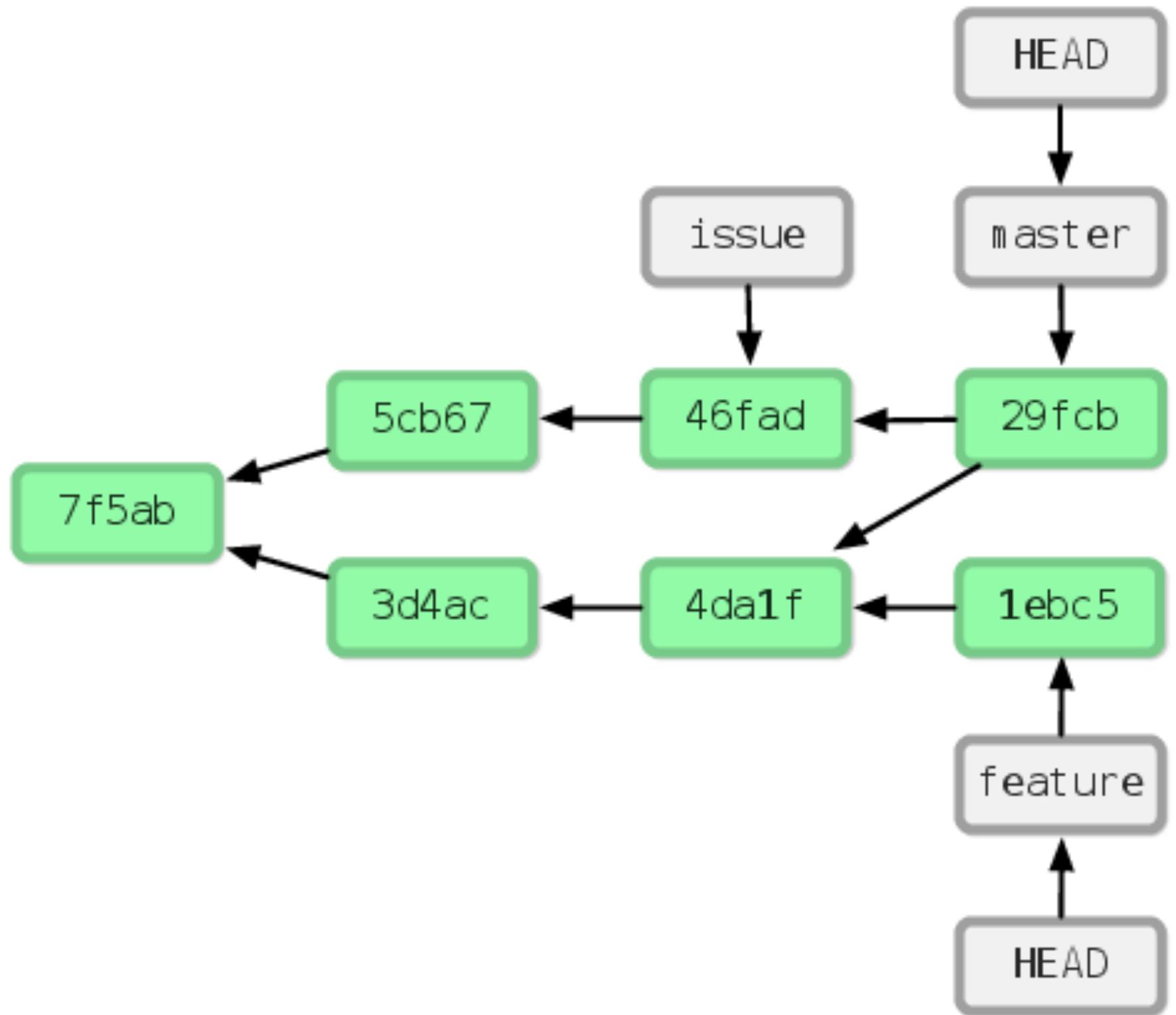
```
$ git checkout feature
```



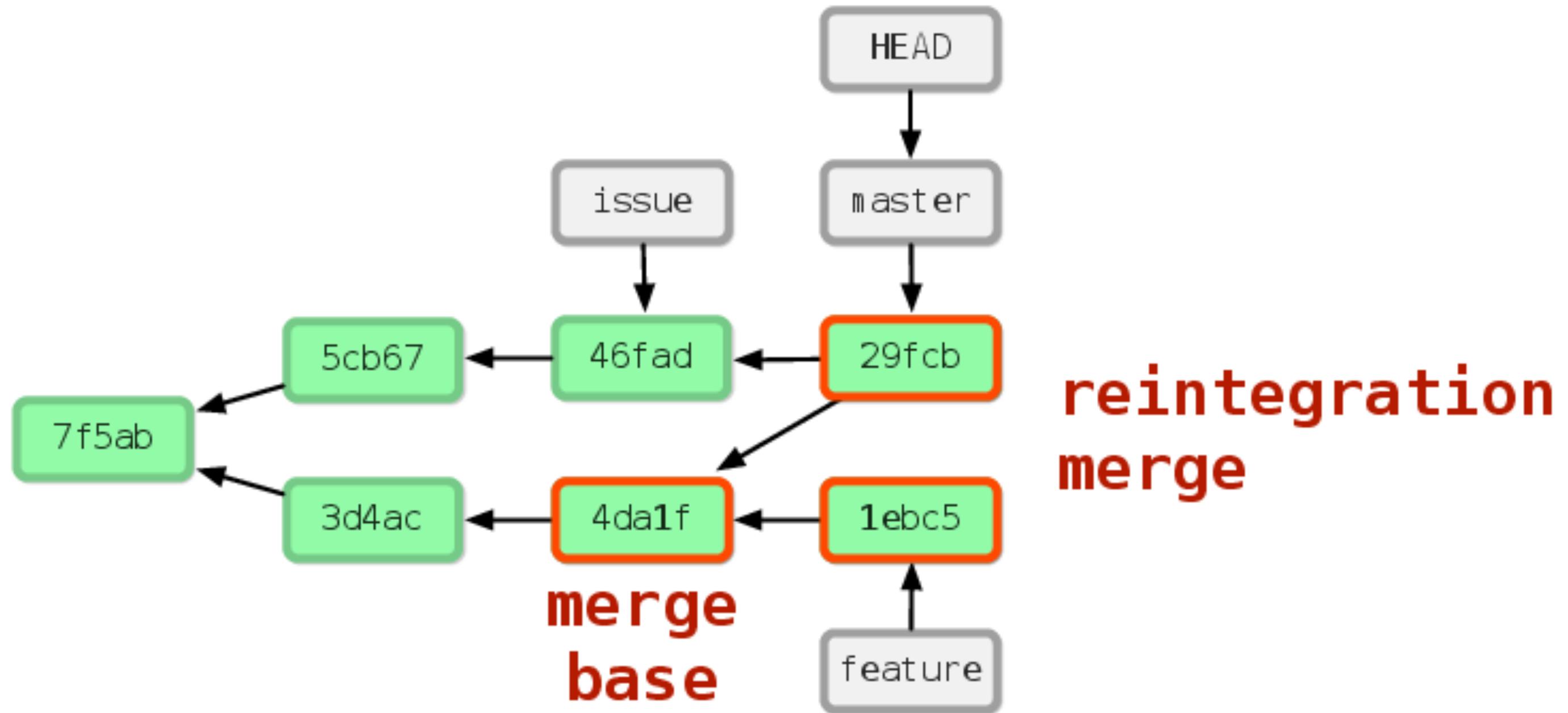
```
$ git commit -am "More on feature."
```



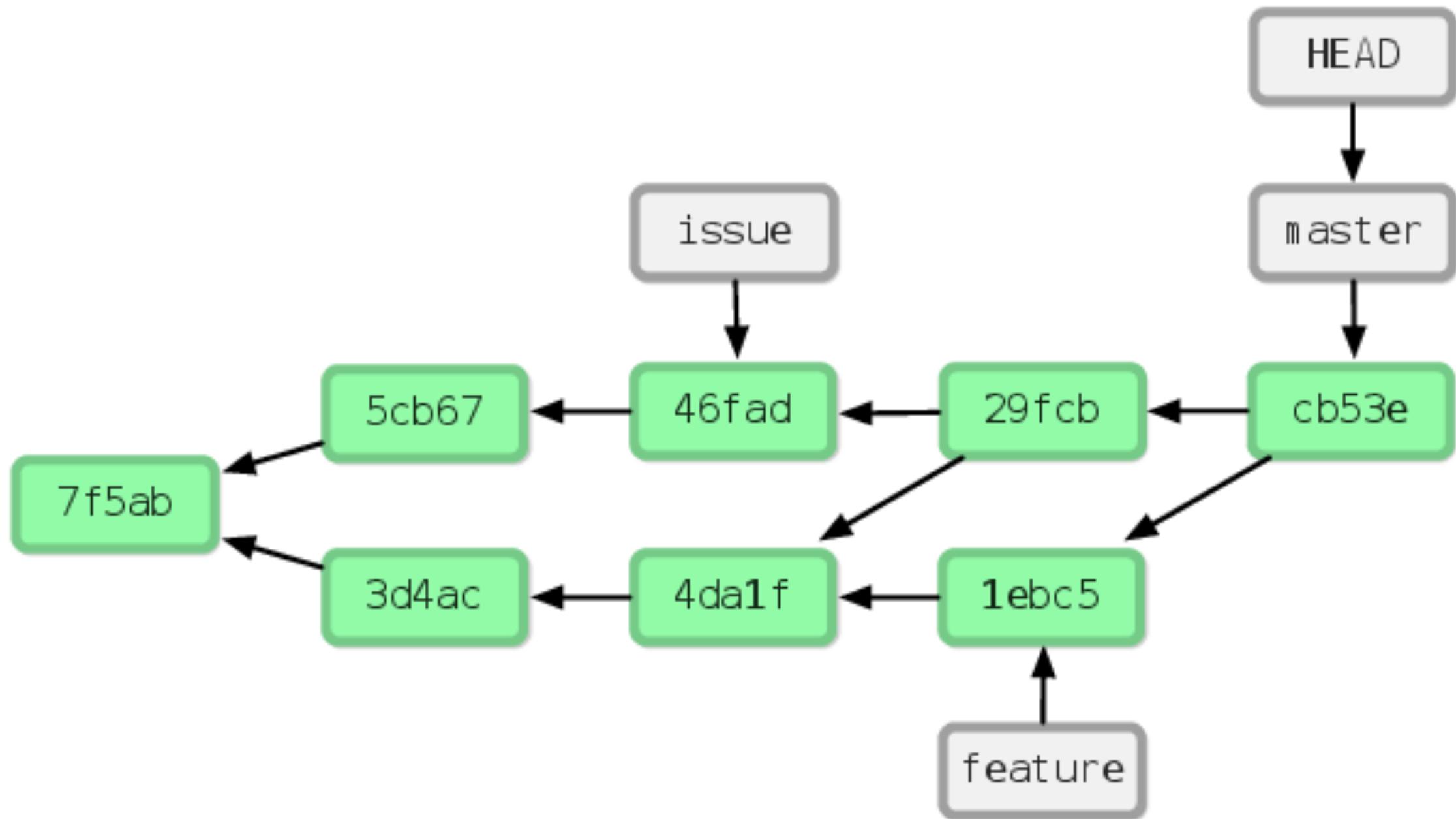
```
$ git checkout master
```



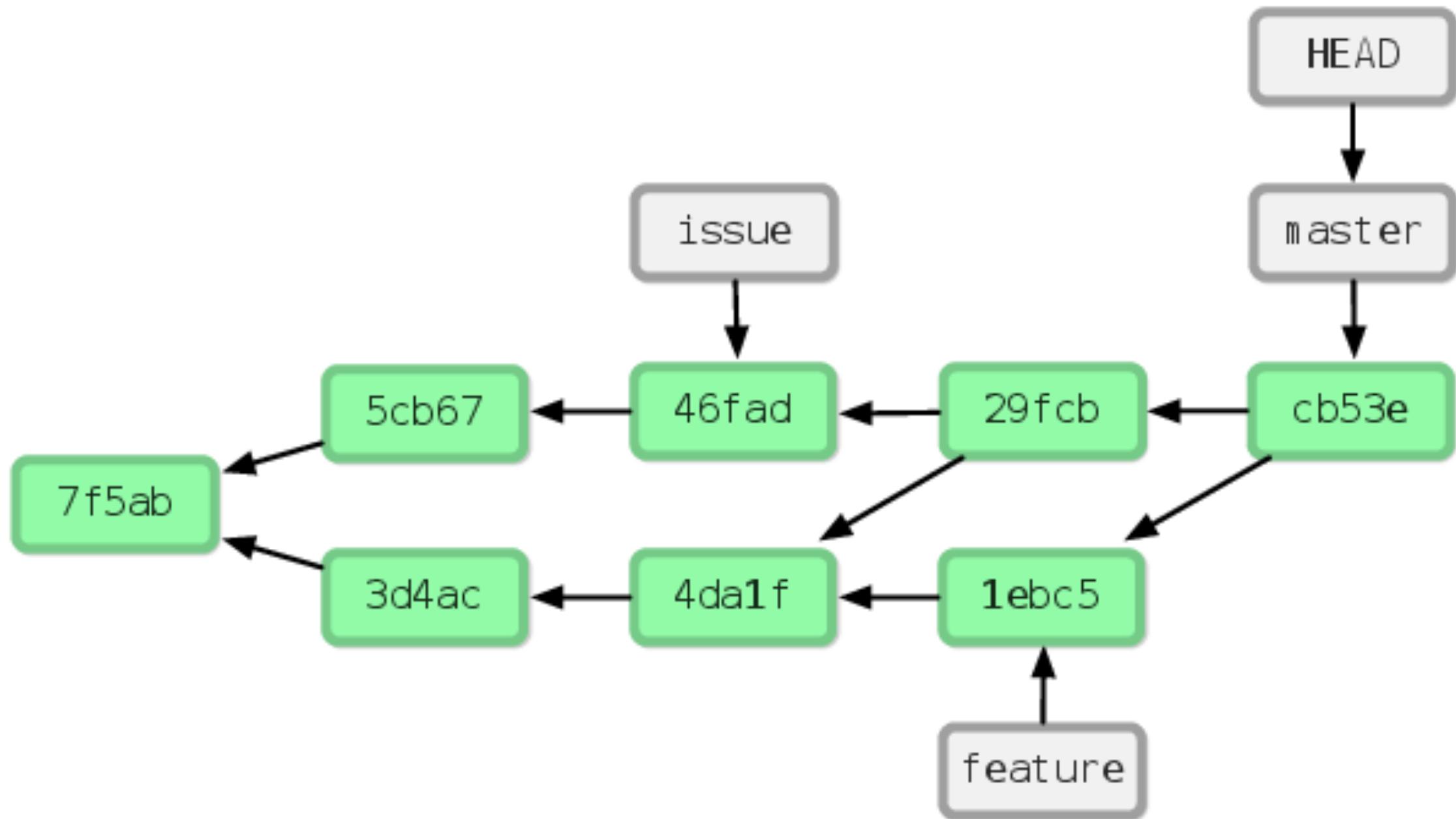
```
$ git merge feature
```



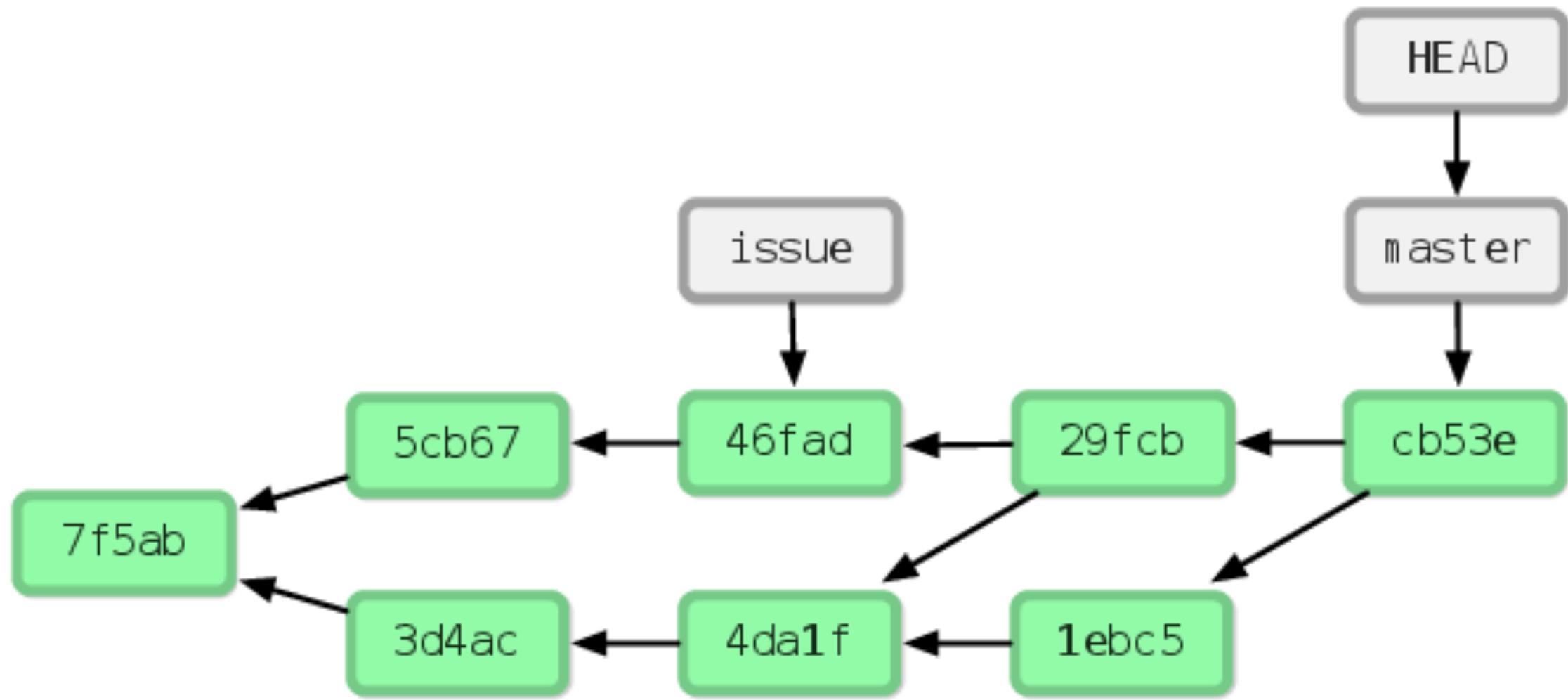
```
$ git merge feature
```



```
$ git branch -d feature
```



```
$ git branch -d issue
```



# Branching & Merging

**Delete branch with extreme prejudice.**

```
$ git branch -D <name>
```