Master in High Performance Computing

# An introduction to modern tools for collaborative science

**Luca Heltai <luca.heltai@sissa.it>**

Mathematical Analysis, Modeling, and Applications (math.sissa.it)
Theoretical and Scientific Data Science (datascience.sissa.it)

sissa mathLab innovating with mathematics

The Abdus Salam International Centre for Theoretical Physics

SISSA

# Target Audience

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

sissa mathLab innovating with mathematics
ICTP The Abdus Salam International Centre for Theoretical Physics
SISSA

# Syllabus

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

sissa mathLab innovating with mathematics
The Abdus Salam International Centre for Theoretical Physics
SISSA

Master in High Performance Computing

# An introduction to modern tools for collaborative science

## Lecture 1 - Introduction to GIT

**Luca Heltai <luca.heltai@sissa.it>**

Mathematical Analysis, Modeling, and Applications (math.sissa.it)
Theoretical and Scientific Data Science (datascience.sissa.it)

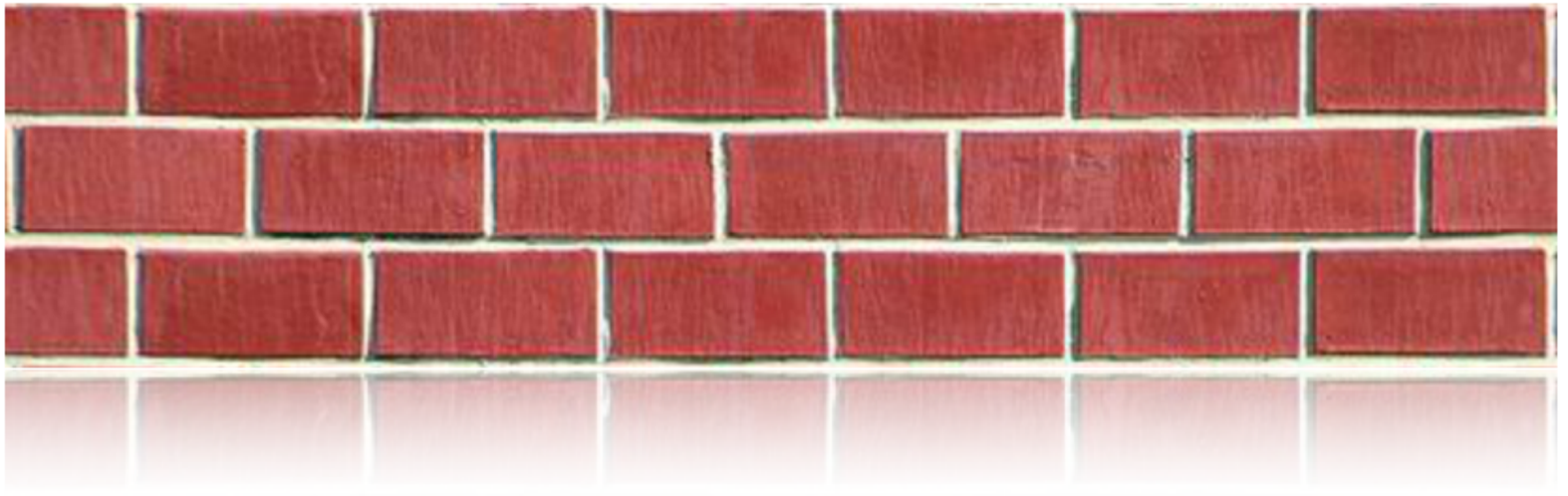
sissa mathLab innovating with mathematics

The Abdus Salam International Centre for Theoretical Physics

SISSA

MHPC
Master in High Performance Computing

# Version Control

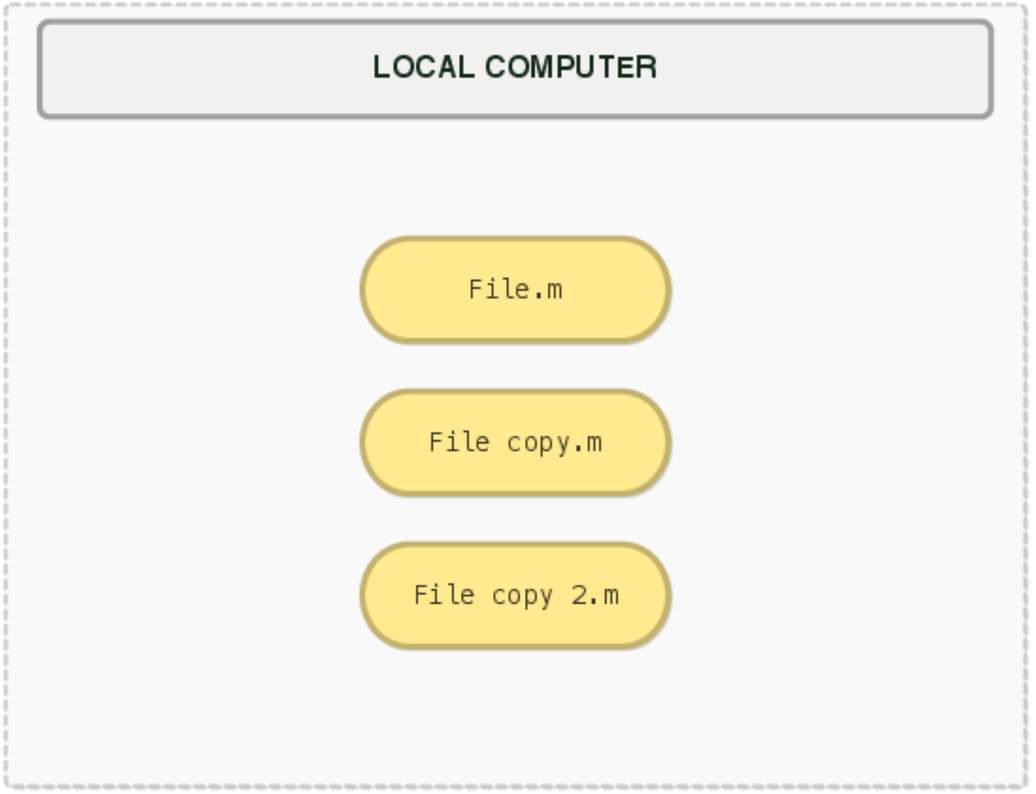


# Workflow

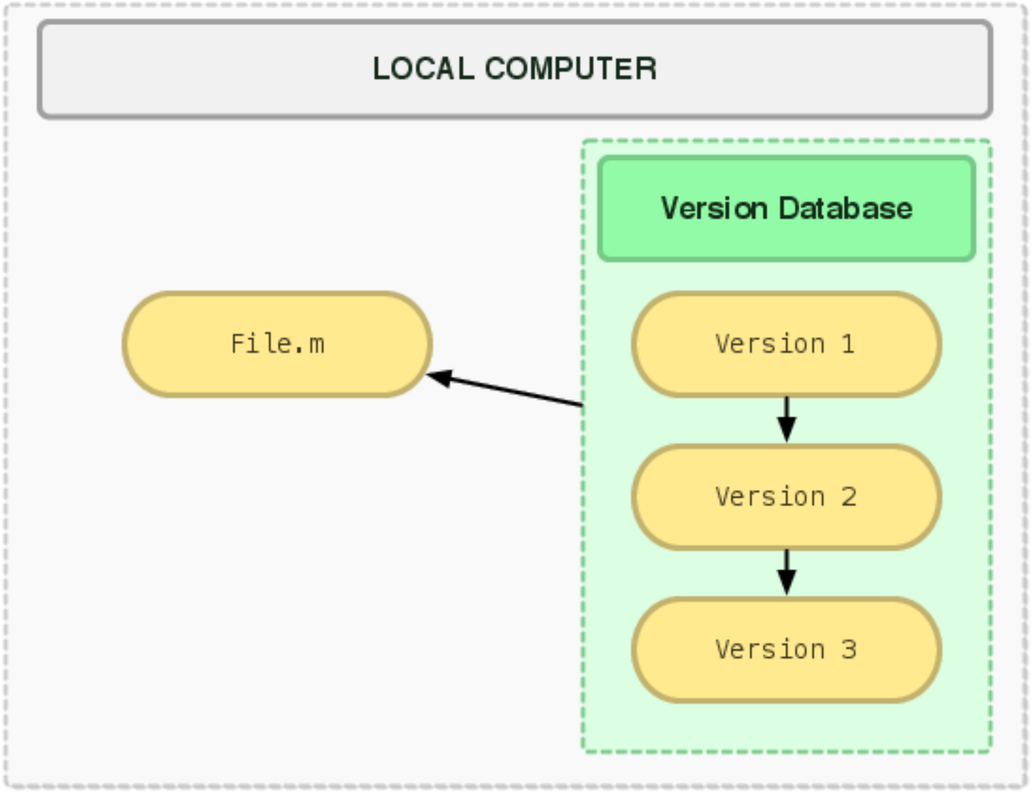
MHPC
Master in High Performance Computing

# History

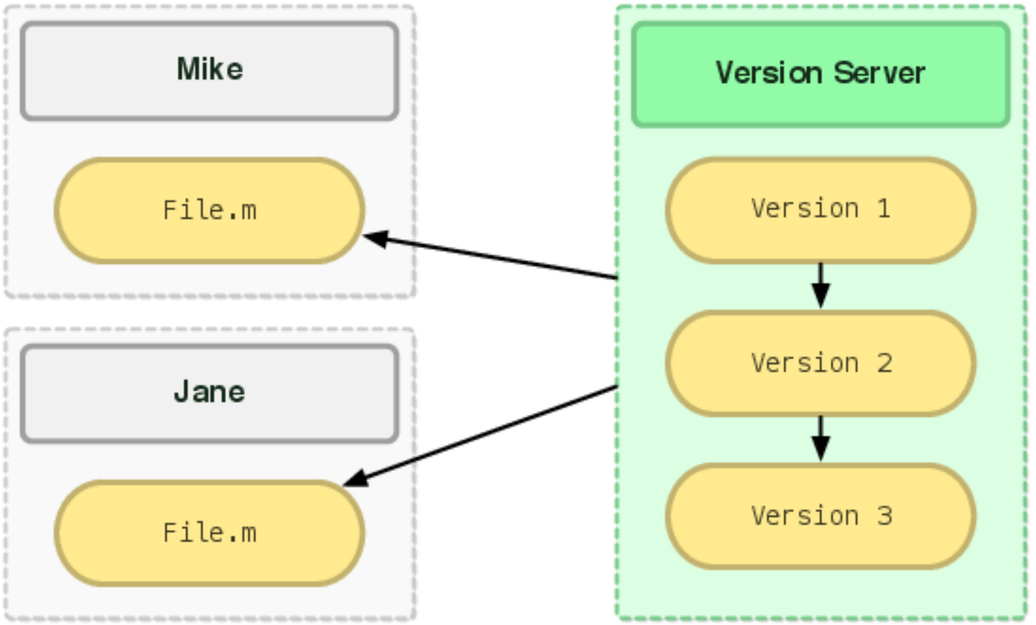
# Local Filesystem



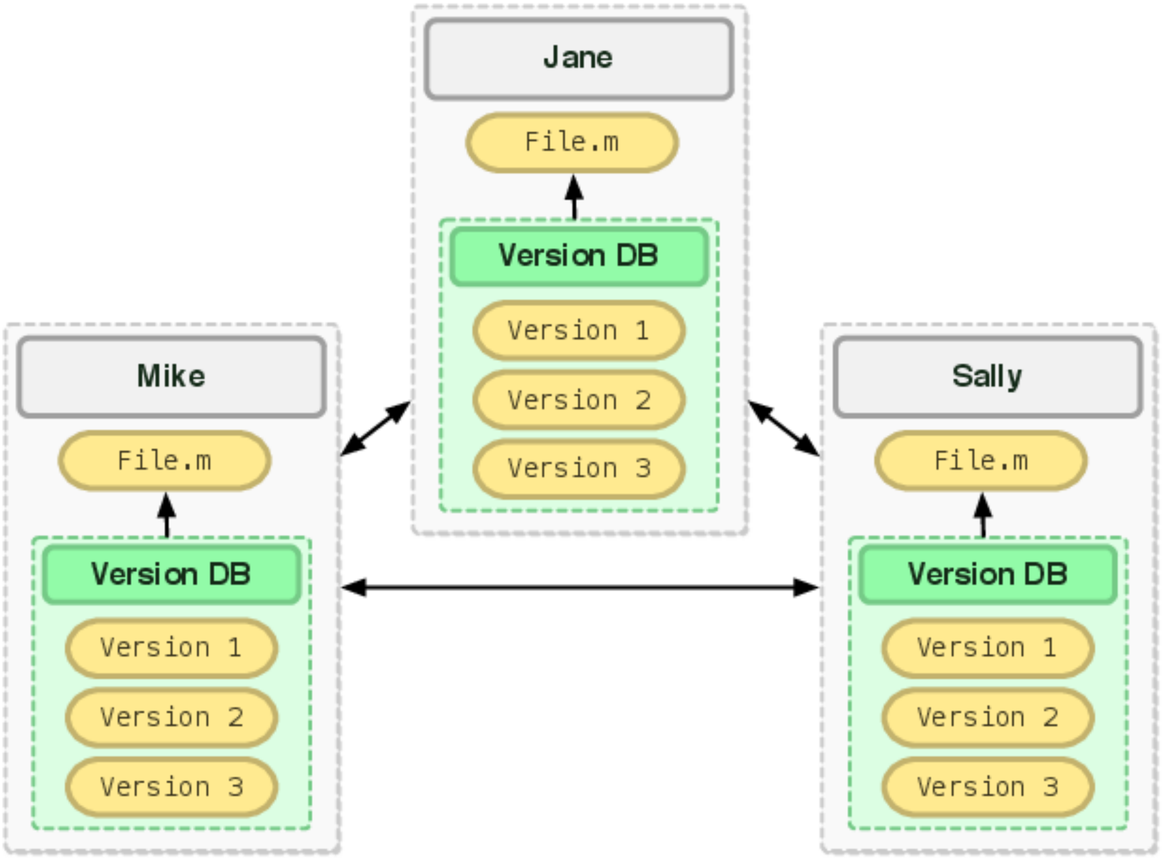
# LOCAL
# Version Control System



# CENTRALIZED
# Version Control System



# DISTRIBUTED
# Version Control System



MHPC
Master in High Performance Computing

# Everything is Local

## (Almost)

# No Network Required

| | |
|---|---|
| Create Repo | Status |
| Commit | Revisions |
| Merge | Diff |
| Branch | History |
| Rebase | Bisect |
| Tag | Local Sync |

MHPC
Master in High Performance Computing

# Advantages

Everything is Fast

Everything is Transparent

Every Clone is a Backup

You Can Work Offline

MHPC
Master in High Performance Computing

# Storage

MHPC
Master in High Performance Computing

# Delta Storage

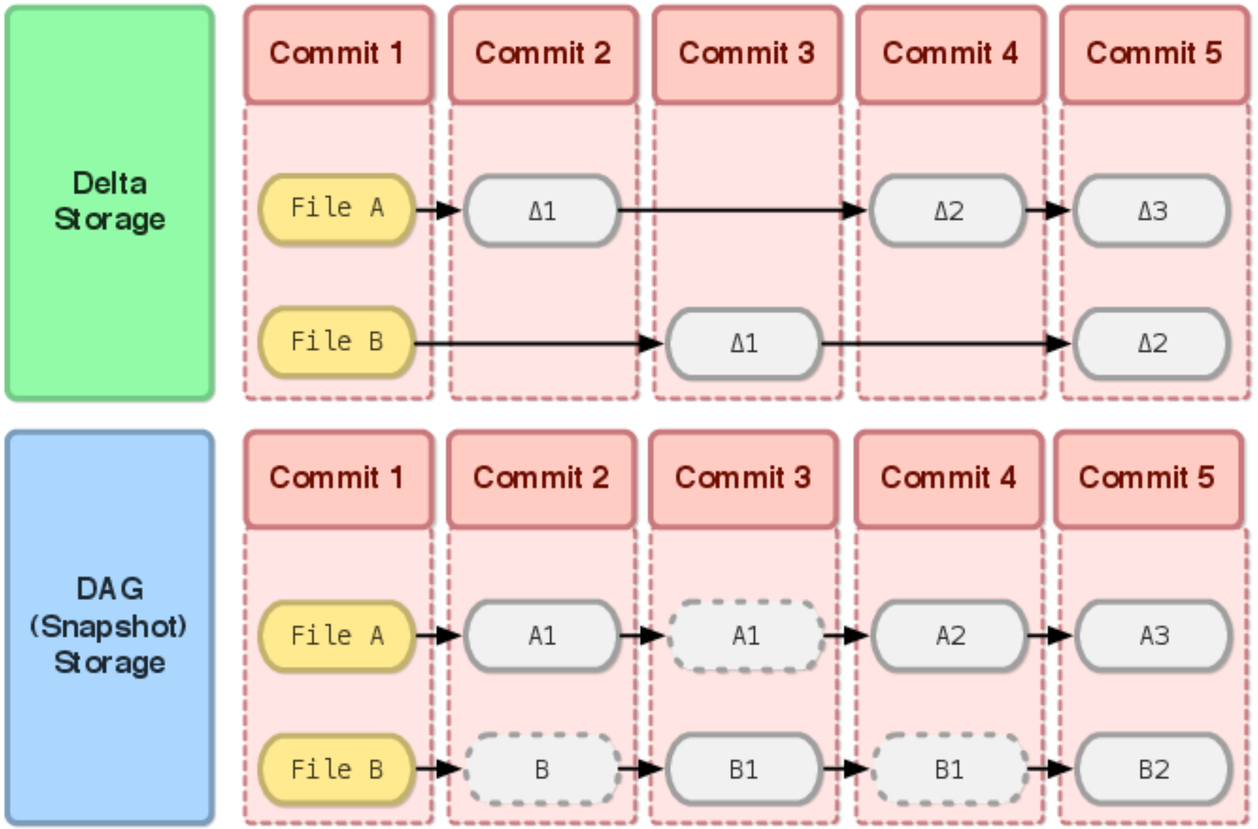
MHPC
Master in High Performance Computing

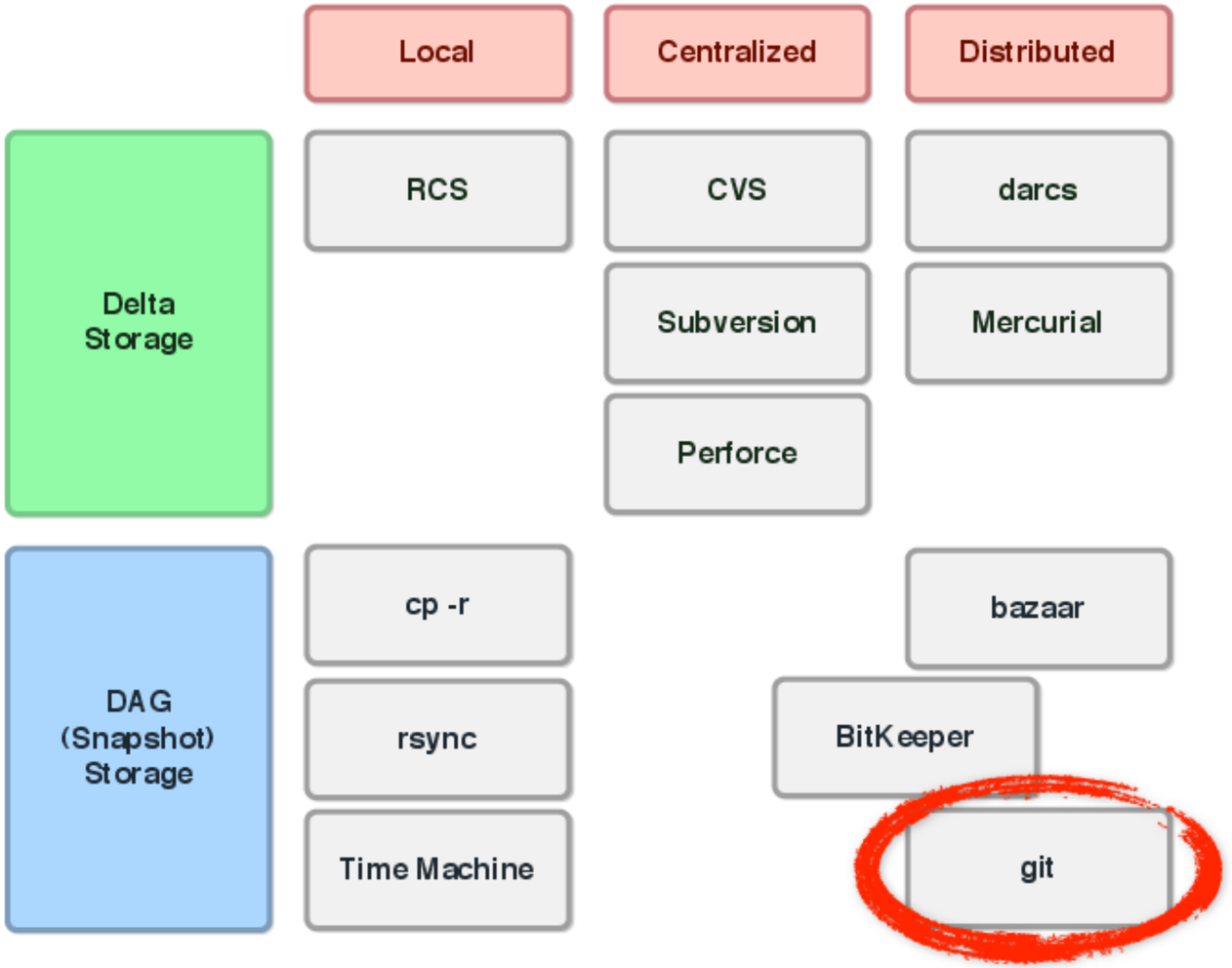
# Snapshot Storage

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

# Delta vs. Snapshot

MHPC
Master in High Performance Computing




Local
Centralized
Distributed
Delta Storage
RCS
CVS
darcs
Subversion
Mercurial
Perforce
DAG (Snapshot) Storage
cp -r
bazaar
rsync
BitKeeper
Time Machine
git

MHPC
Master in High Performance Computing

# About Git

MHPC
Master in High Performance Computing

# 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 | Rails |
| Linux | Android |
| Perl | PostgreSQL |
| Eclipse | KDE |
| Qt | Gnome |

MHPC
Master in High Performance Computing

# 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 14:36 .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
...
```

MHPC
Master in High Performance Computing

# Git Directory

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

MHPC
Master in High Performance Computing

# .git only in root of Working Directory

## (unlike Subversion)

MHPC
Master in High Performance Computing

# Git Directory

Configuration File

Hooks

Object Database

References

Index

MHPC
Master in High Performance Computing

# Git Directory

~~Configuration File~~

~~Hooks~~

Object Database

References

Index

MHPC
Master in High Performance Computing

# Object Database

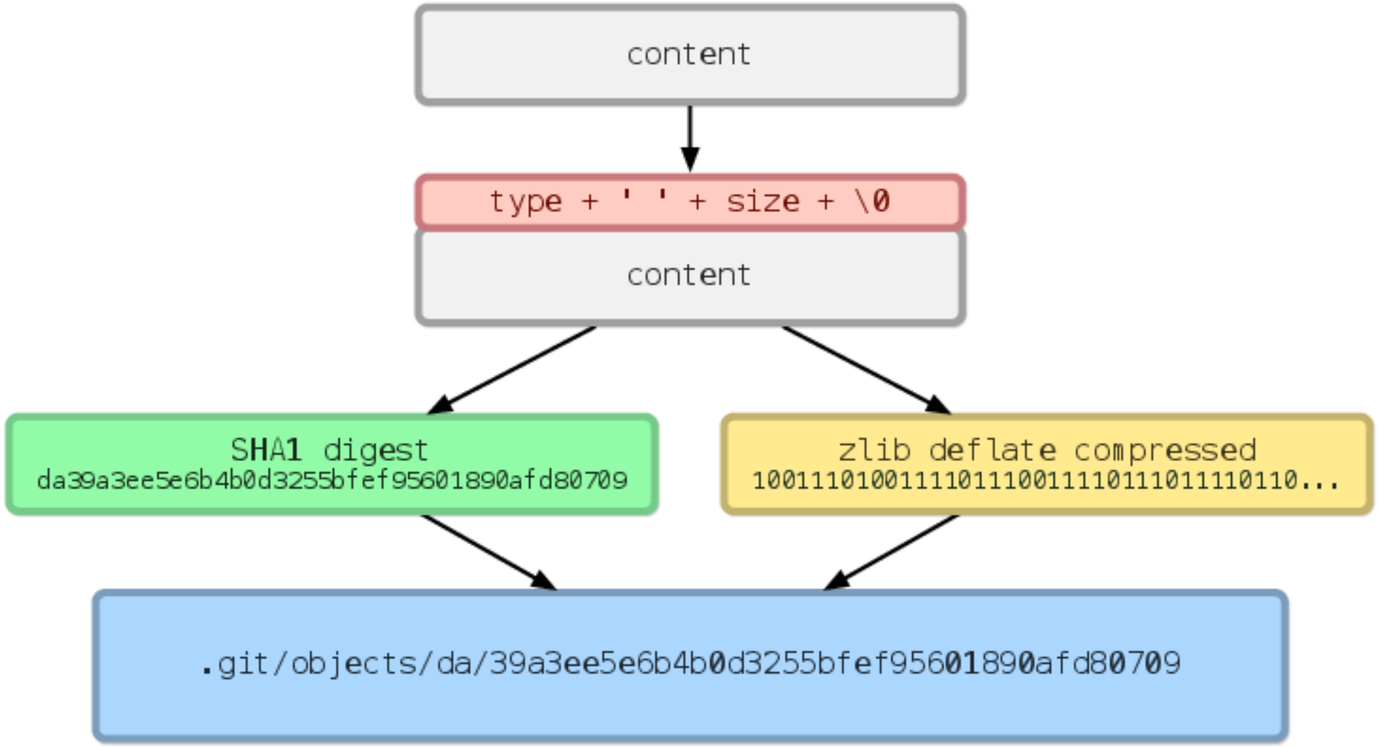
# Object Database

≈ NSDictionary

(Hash Table / Key-Value Store)

# Object Database

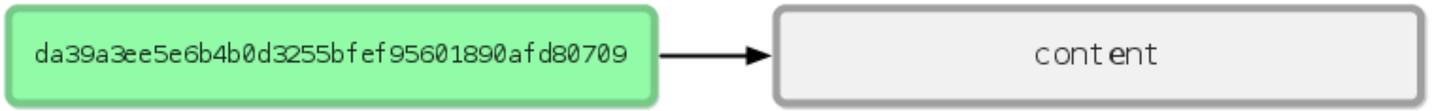
# Object Database




"loose format"

# Object Database

## ≈ NSDictionary




**Pointer / Key** **Object / Value**

MHPC
Master in High Performance Computing

# Object Database


**Equivalent if common prefix is unique!**

MHPC
Master in High Performance Computing

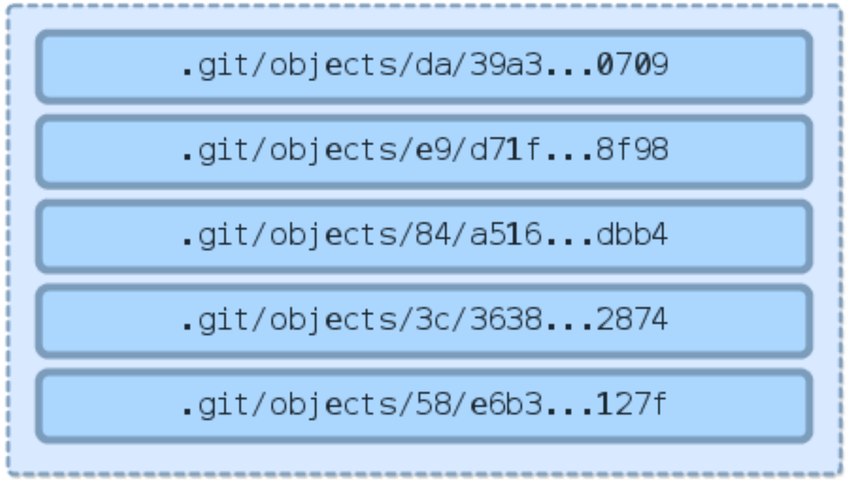
# Object Database

## Garbage Collection

MHPC
Master in High Performance Computing

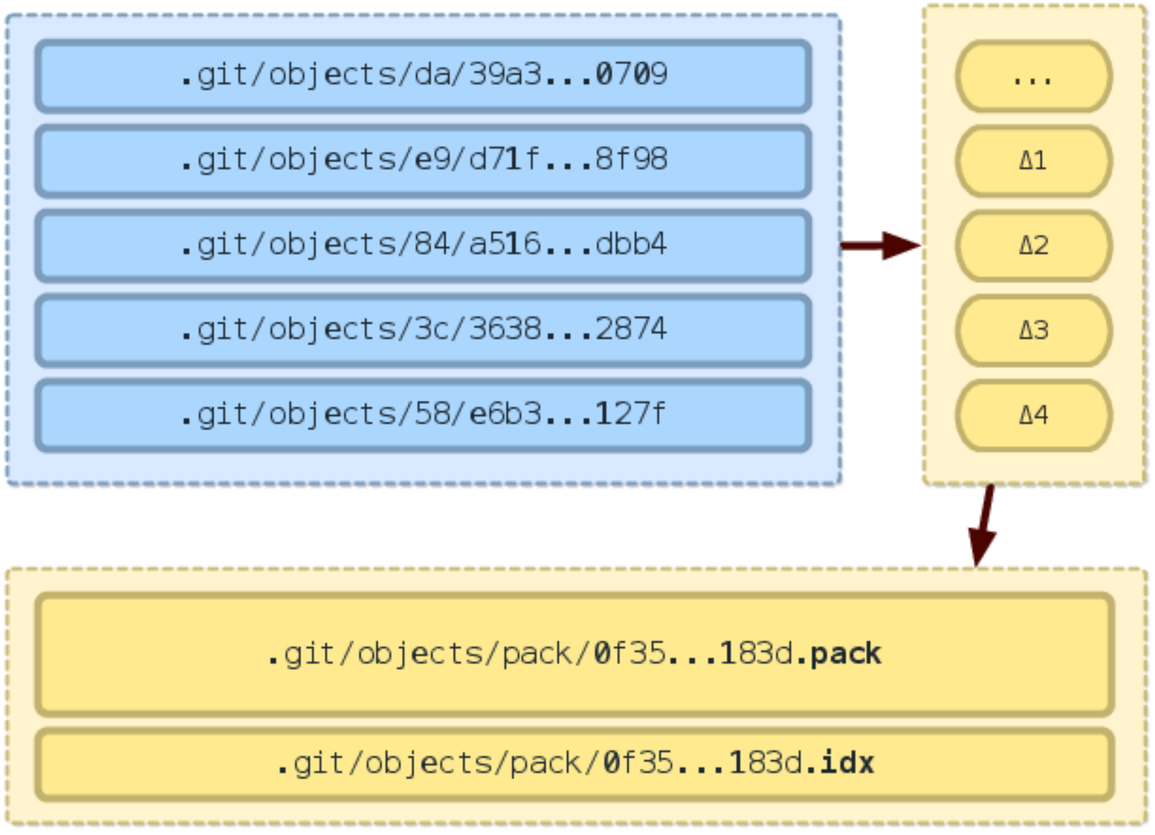
# Object Database

## Similar Objects




`git gc`

# Object Database




"packed format"

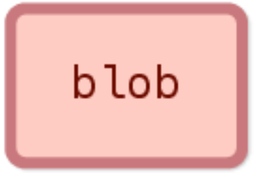
MHPC
Master in High Performance Computing

# Object Database

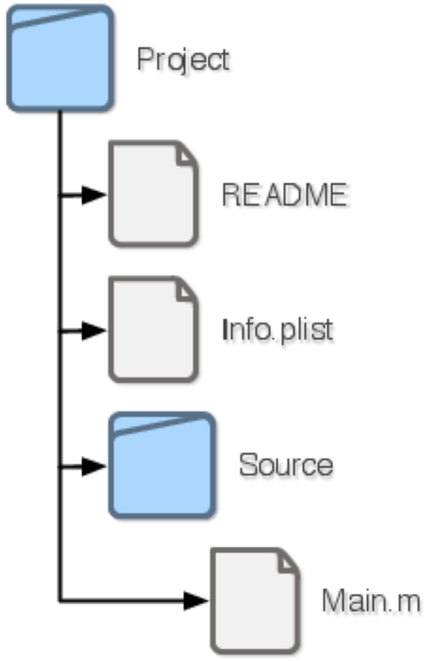
## Four Object Types

# Object Database

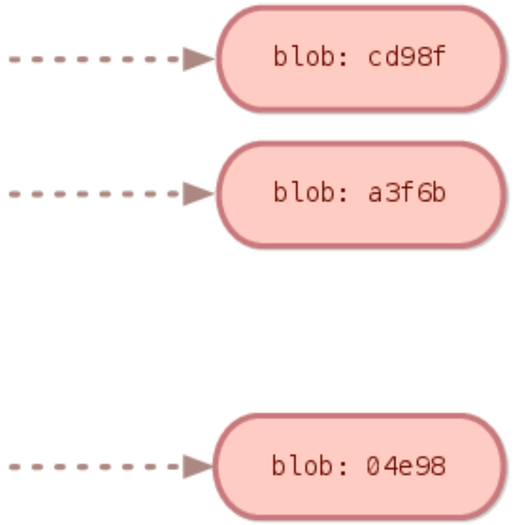
# Object Database




## Working Directory




## Git Directory



MHPC
Master in High Performance Computing

# Object Database

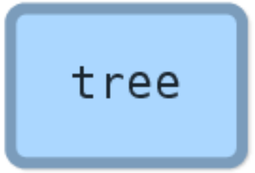
blob

blob 109\0

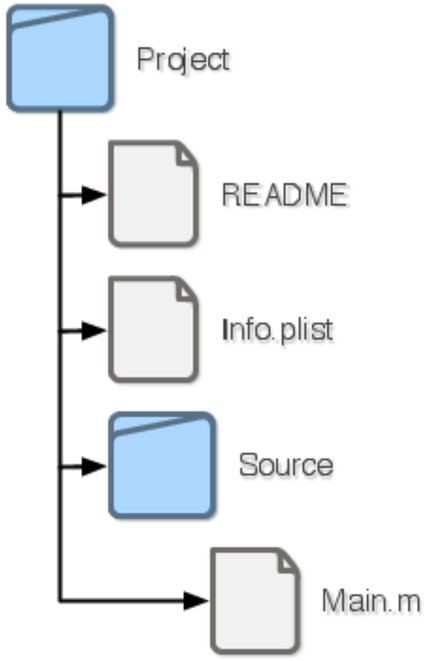
```
#import <Cocoa/Cocoa.h>

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

# Object Database

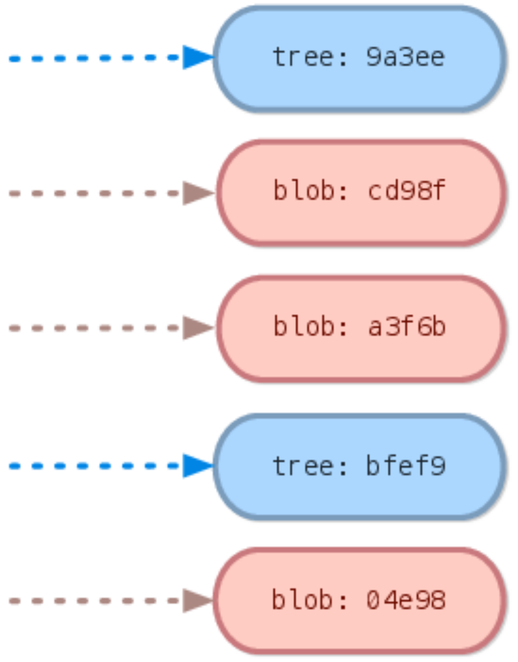
# Object Database



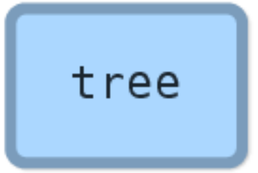

## Working Directory




## Git Directory



# Object Database

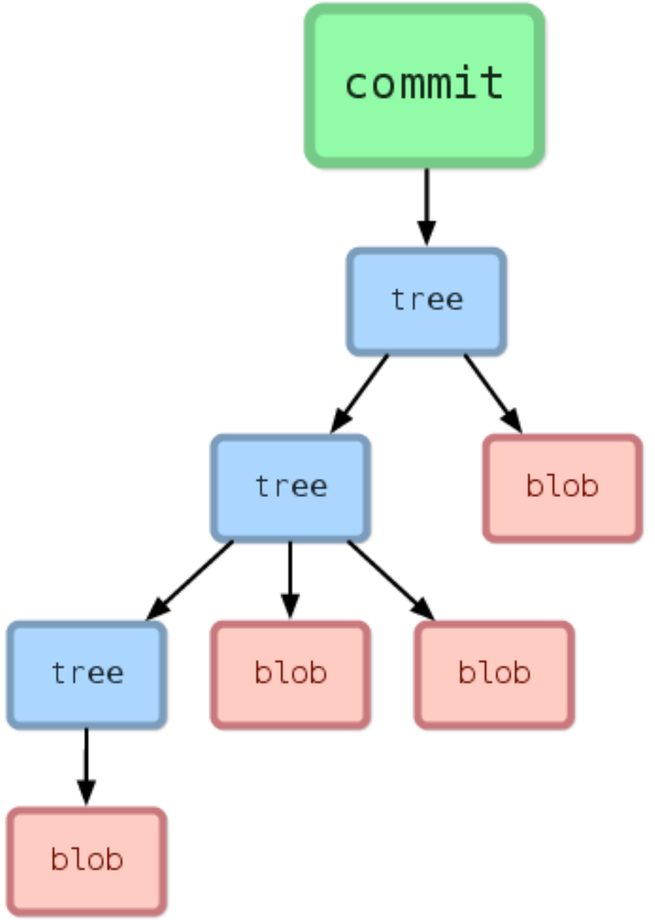



```
tree 84\0
```

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

# Object Database

# Object Database



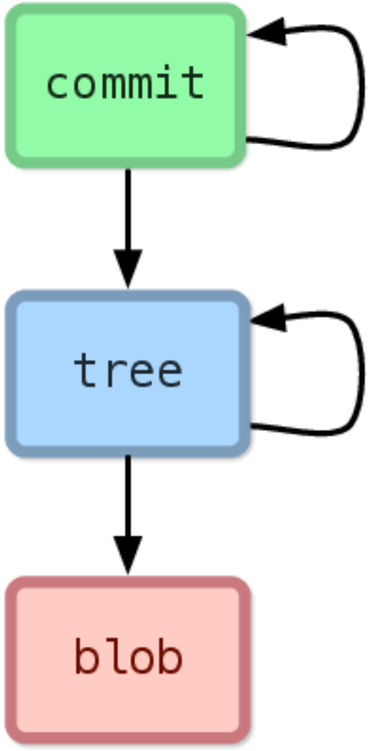
# Object Database

commit

```
commit 155\0
```

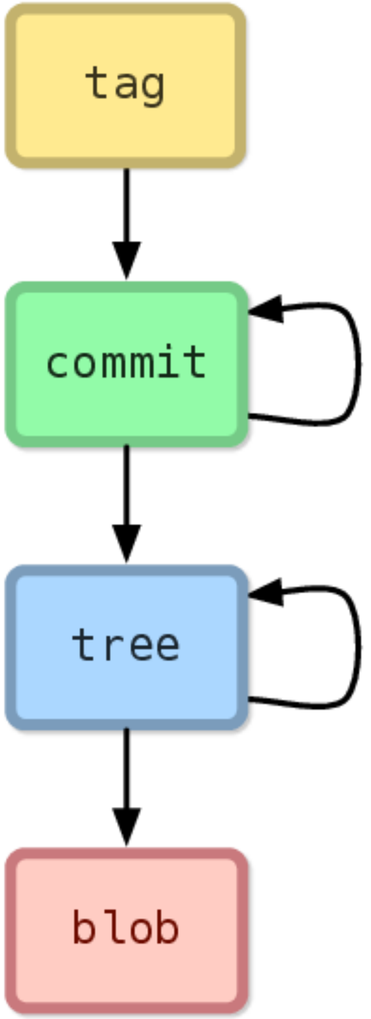
```
tree 9a3ee
parent fb39e
author Patrick Hogan <pbhogan@gmail.com> 1311810904
committer Patrick Hogan <pbhogan@gmail.com> 1311810904

Fixed a typo in README.
```

# Object Database



# Object Database

# Object Database



MHPC
Master in High Performance Computing

# Object Database


```
tag 121\0
```

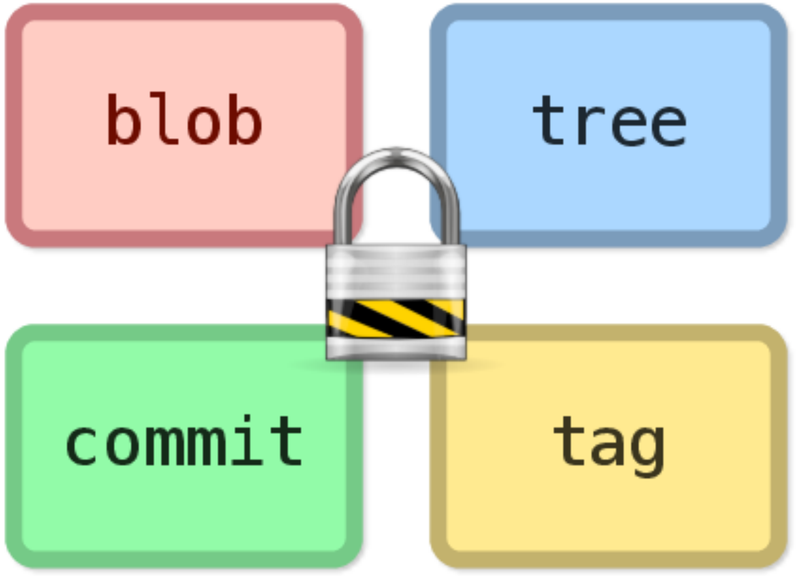
```
object e4d23e
type commit
tag v1.2.0
tagger Patrick Hogan <pbhogan@gmail.com> 1311810904

Version 1.2 release -- FINALLY!
```

**.git/objects/20/c71174453dc760692cd1461275bf0cffeb772f**

**.git/refs/tags/v1.2.0**

MHPC
Master in High Performance Computing

# Object Database




**Immutable!**

MHPC
Master in High Performance Computing

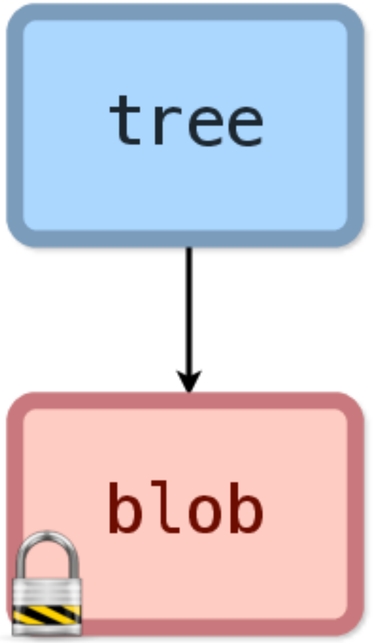
# Never Removes Data

## (Almost)

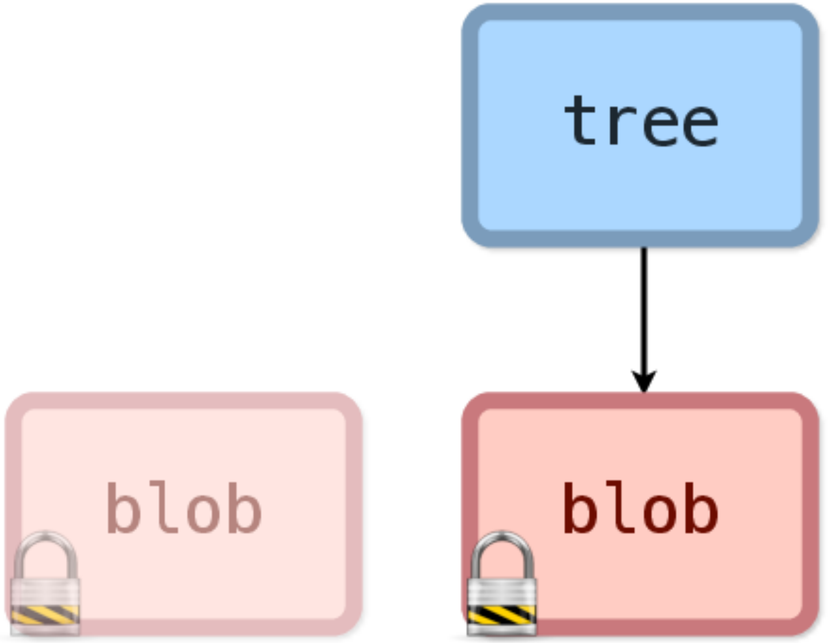
MHPC
Master in High Performance Computing

# "Rewriting History" Writes Alternate History

# Object Database



# Object Database



MHPC
Master in High Performance Computing

# Git Directory

Configuration File

Hooks

Object Database

References

Index

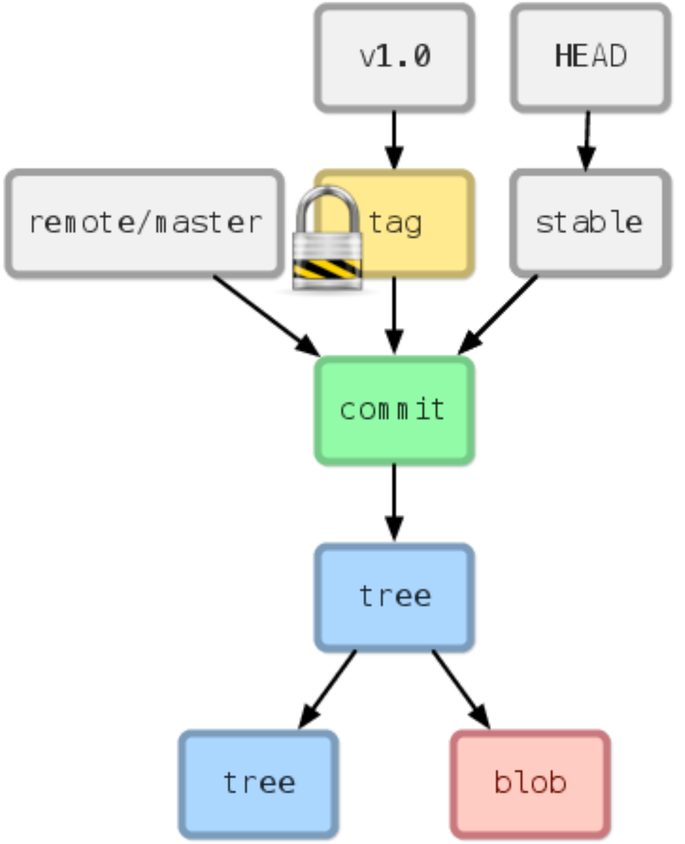
# References

# References

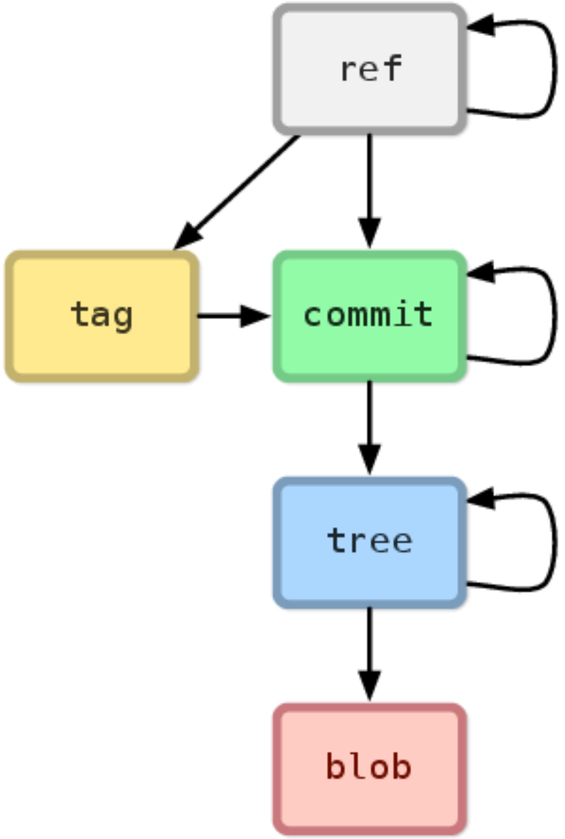
## Lightweight, Movable Pointers to Commits

(and other things)

# References



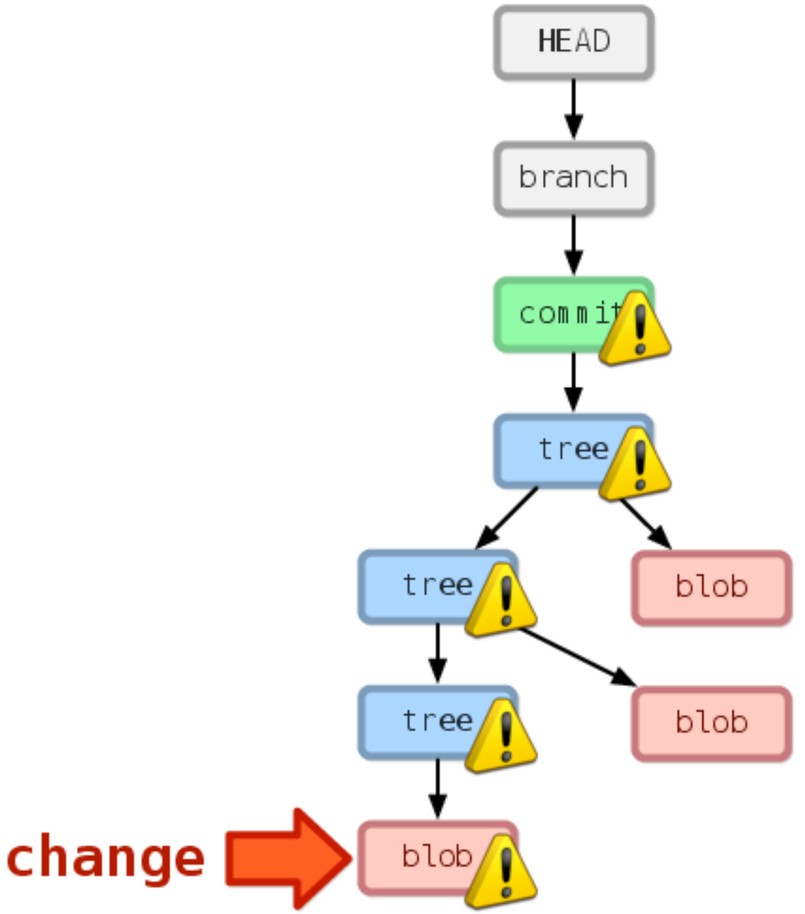
# References



# Scenario

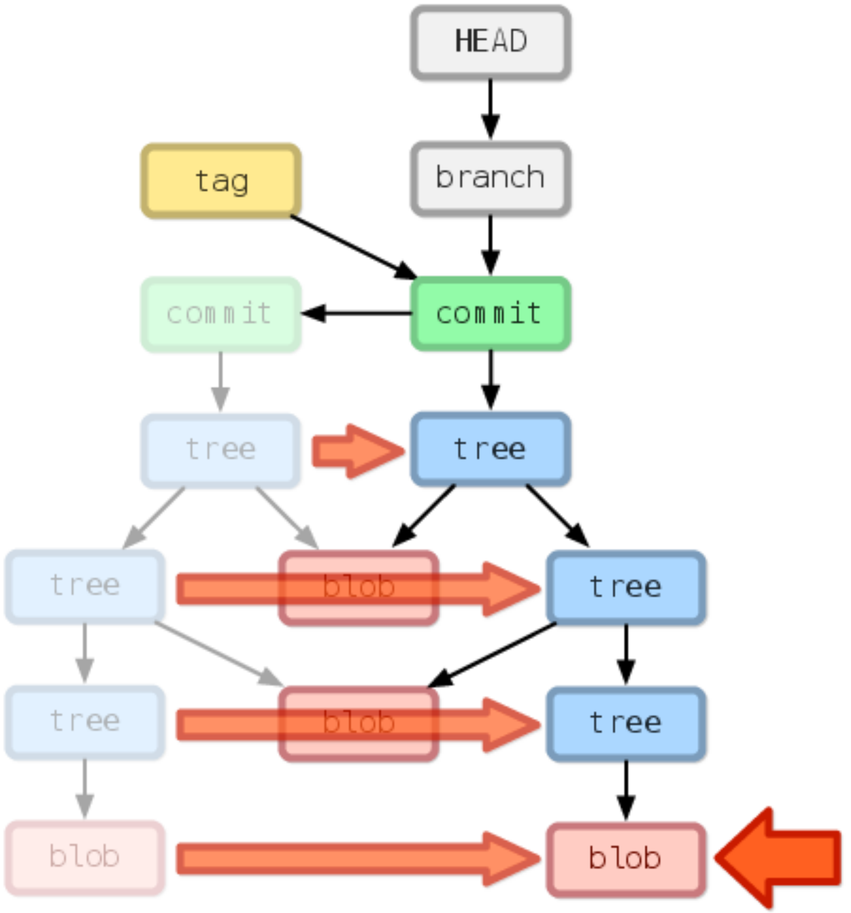
# Scenario

MHPC
Master in High Performance Computing



# Scenario

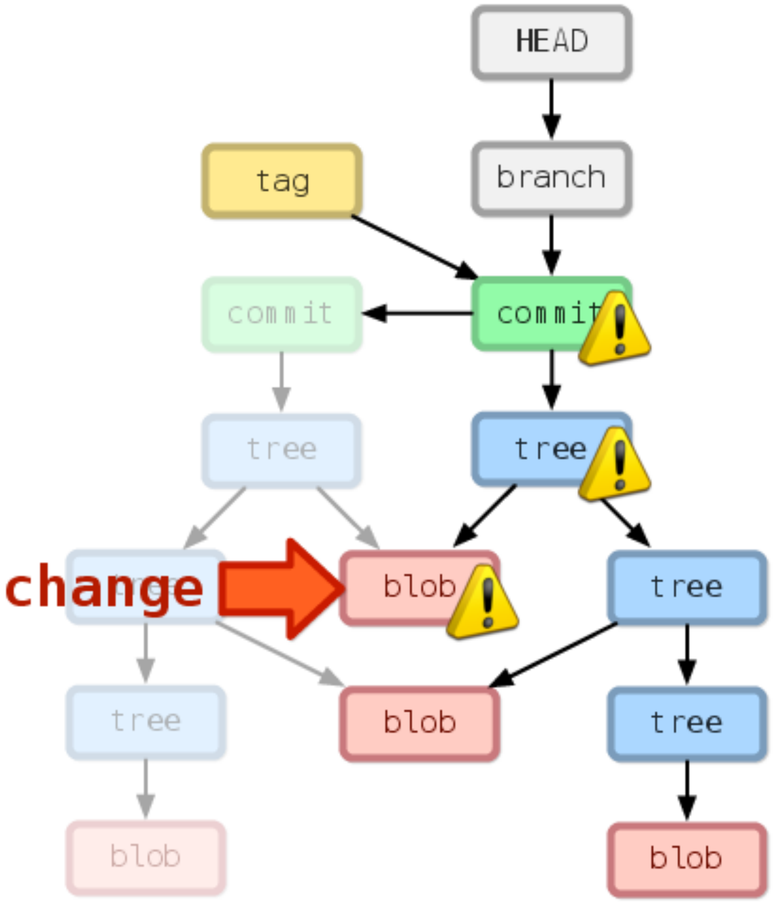
MHPC
Master in High Performance Computing




**new objects**

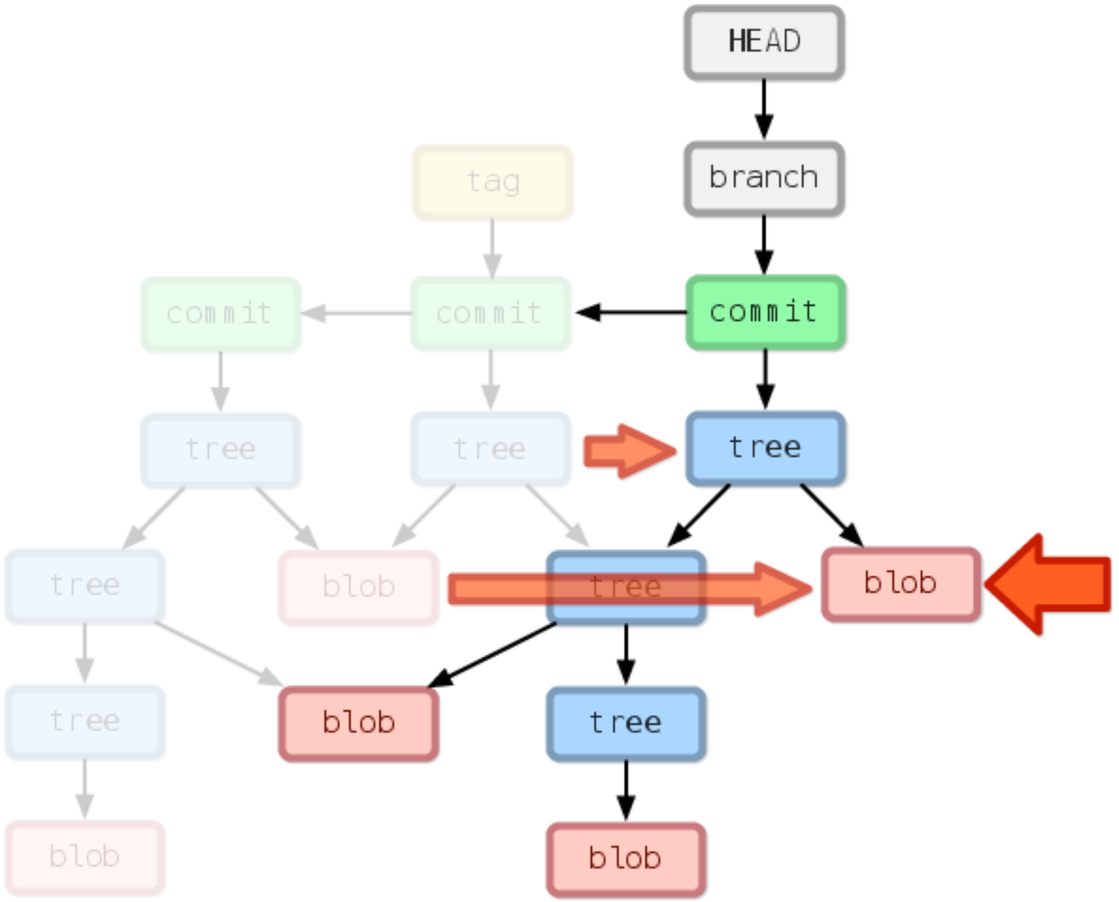
# Scenario




HEAD
tag
branch
commit
commit
tree
tree
change
blob
tree
tree
blob
tree
blob
blob

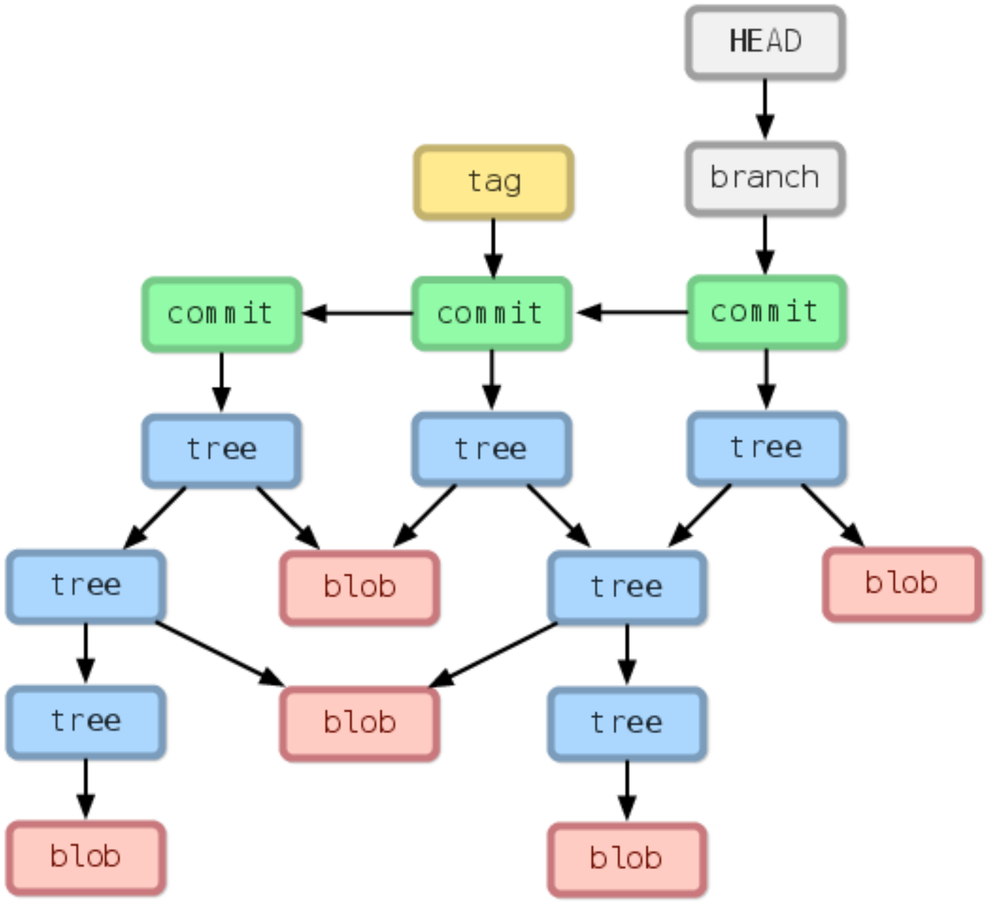
# Scenario




HEAD
branch
tag
commit
commit
commit
tree
tree
tree
tree
blob
tree
blob
tree
blob
tree
tree
blob
blob

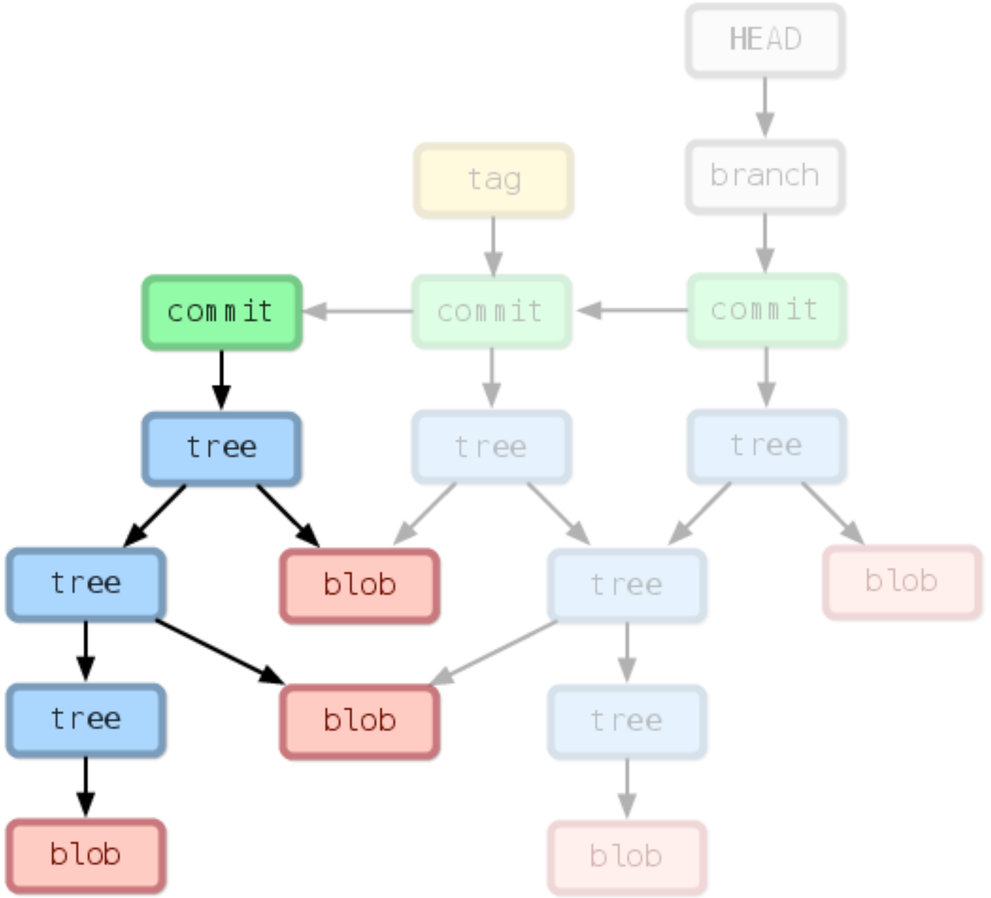
# Scenario




HEAD
tag
branch
commit
commit
commit
tree
tree
tree
tree
blob
tree
blob
tree
blob
tree
blob
blob

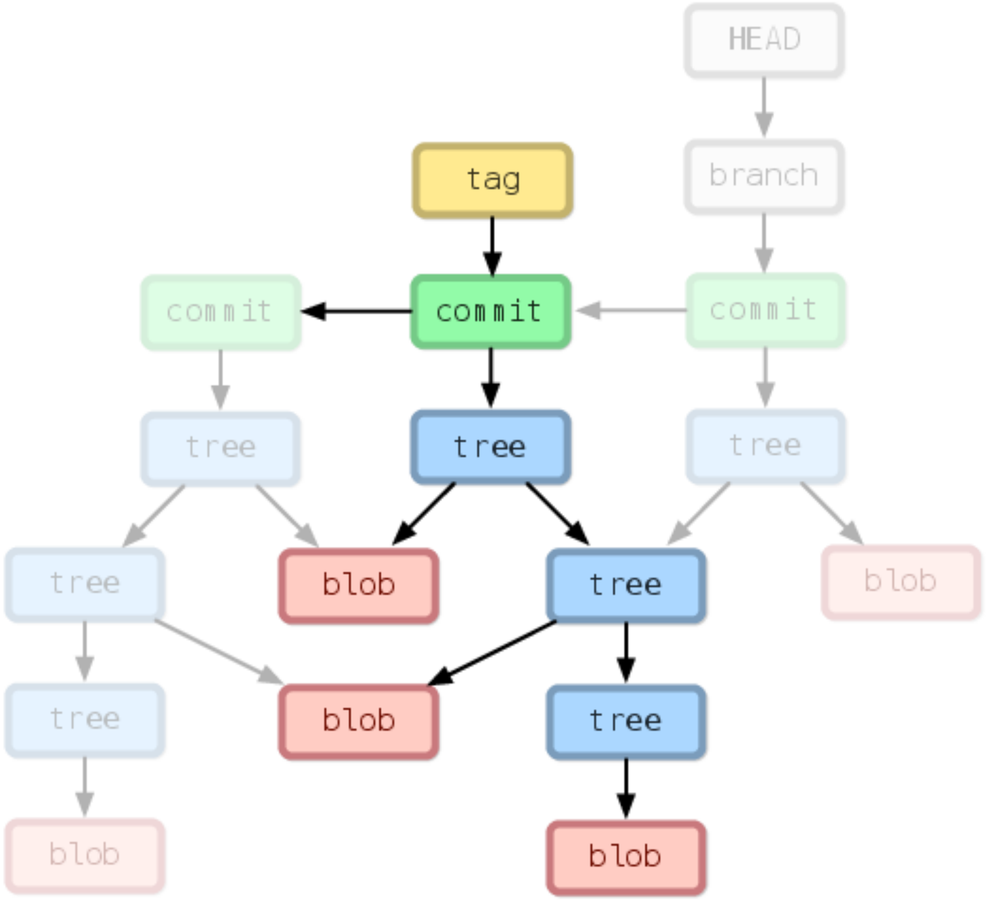
# Scenario

MHPC
Master in High Performance Computing



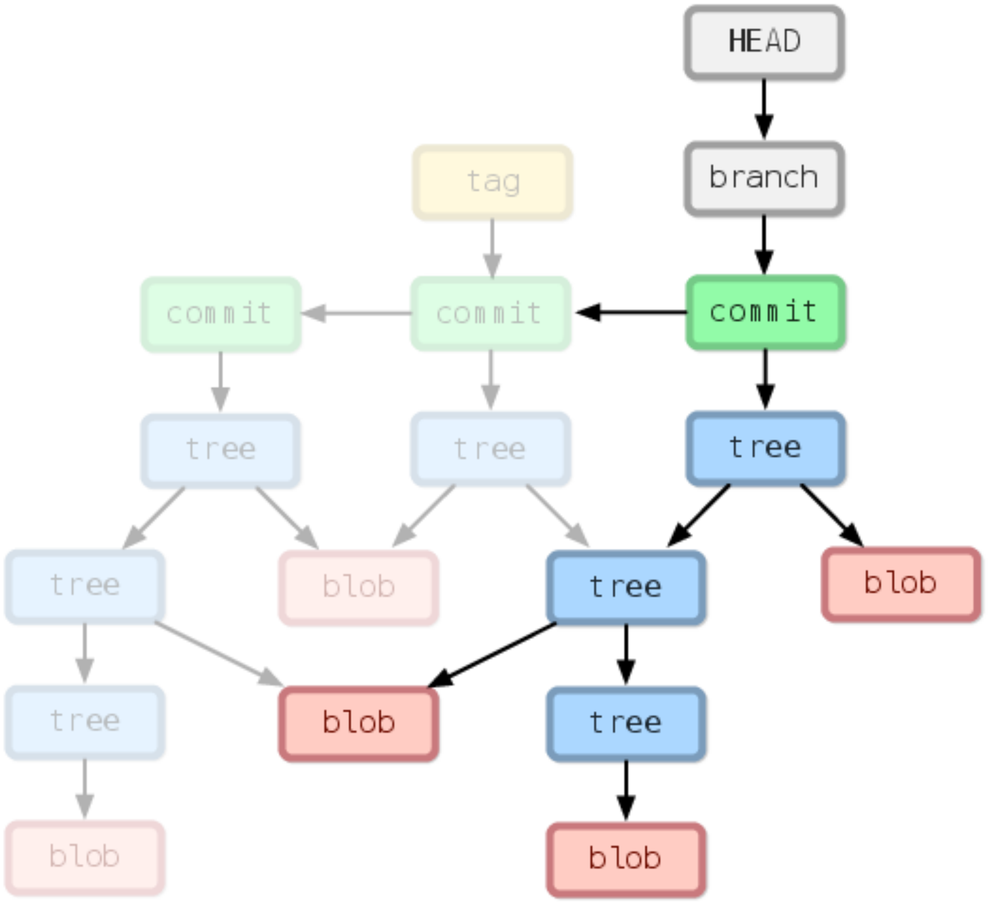
# Scenario




HEAD
tag
branch
commit
commit
commit
tree
tree
tree
tree
blob
tree
blob
tree
blob
tree
blob

# Scenario




HEAD
branch
tag
commit
commit
commit
tree
tree
tree
tree
blob
tree
blob
tree
tree
blob
blob

MHPC
Master in High Performance Computing

# Git Directory

Configuration File

Hooks

Object Database

References

Index

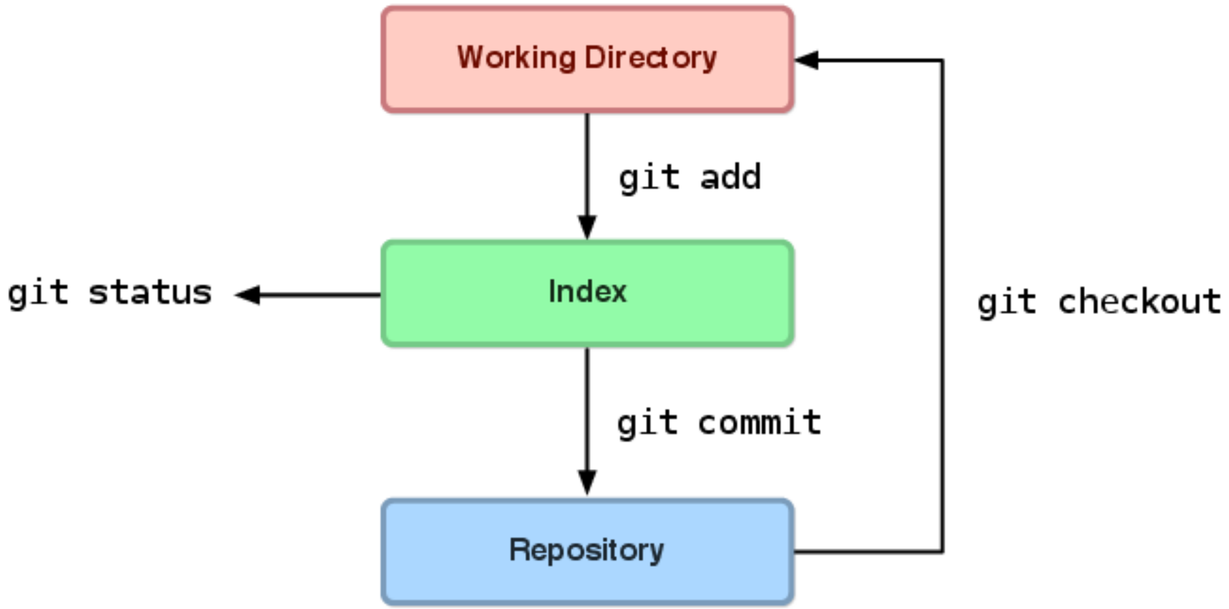
MHPC
Master in High Performance Computing

# Index

# Index

== Staging Area

# Index


Working Directory
git add
git status
Index
git checkout
git commit
Repository


# 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

MHPC
Master in High Performance Computing

# Git Started

# 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
```

MHPC
Master in High Performance Computing

# .gitignore

## Specify files which will be ignored by Git

```
$ cat .gitgnore
.svn
.DS_Store
build
*.pbxuser
*.perspective
*.perspectivev3
*.mode1v3
*.mode2v3
*.xcuserstate
*.xcworkspace
xcuserdata
```

MHPC
Master in High Performance Computing

# Staging

## Stage files to the index.

```
$ git add .
```

# Committing

## Create a commit tagged with a message.

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

MHPC
Master in High Performance Computing

# Branching & Merging

# Branching & Merging

## Create new branch (from current branch)

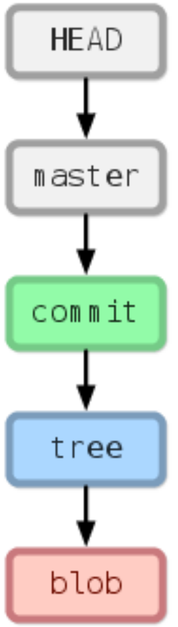
```
$ git branch <name>
```

# Branching & Merging

## Switch to branch (overwrites Working Dir!)

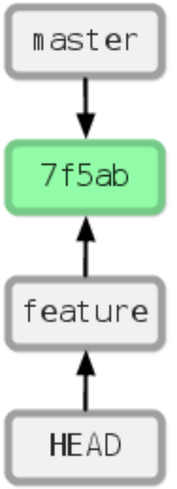
```
$ git checkout <name>
```

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

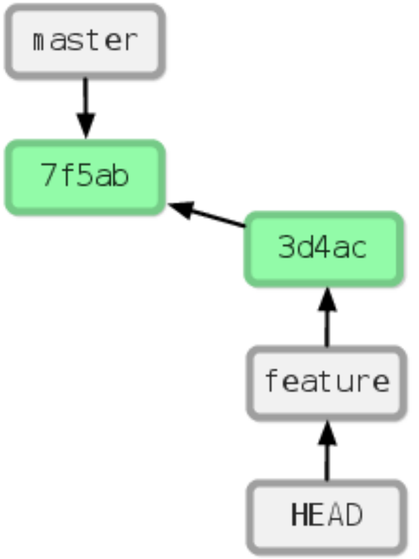


```
$ git commit -m "First commit!"
```

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

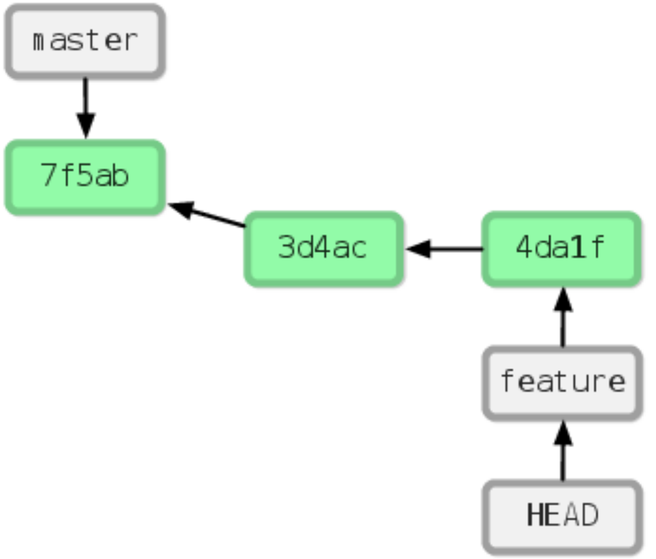


```
$ git add feat.c
$ git commit -m “Added feat.c”
```



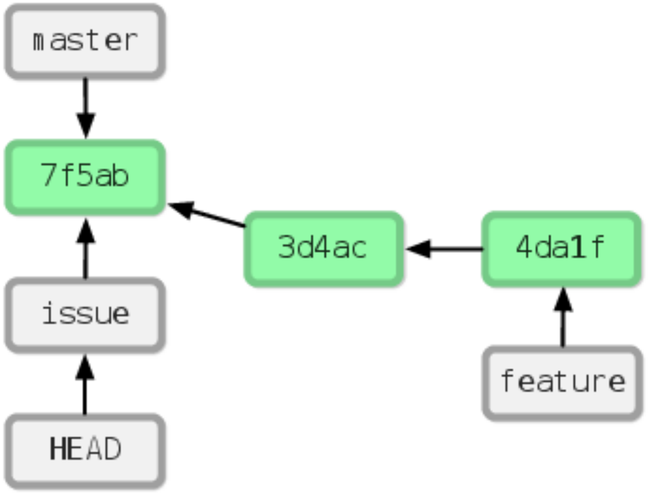
```
$ git commit -a -m "Updated feat.c"
```

feat.c already tracked so -a automatically stages.

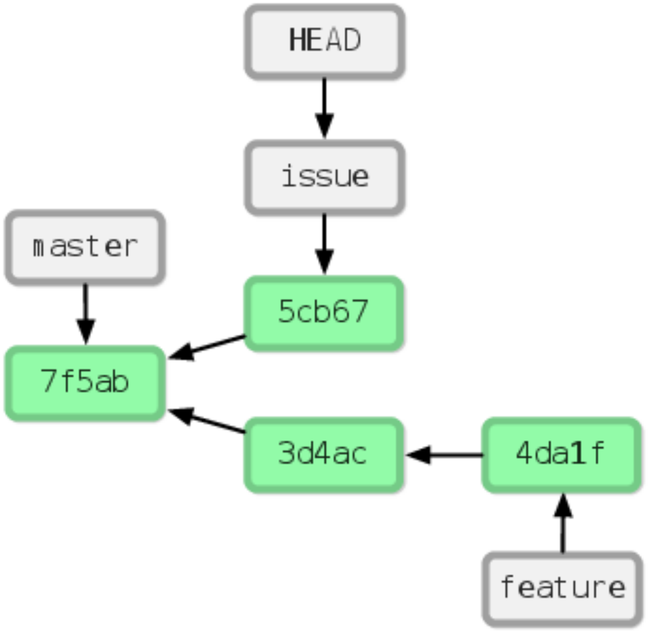


```
$ git checkout -b issue master
```

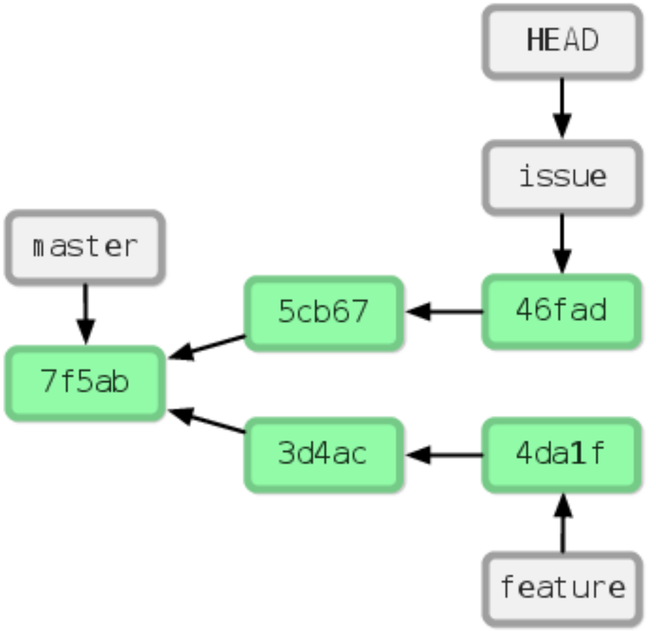
```
git checkout master
git branch issue
git checkout issue
```



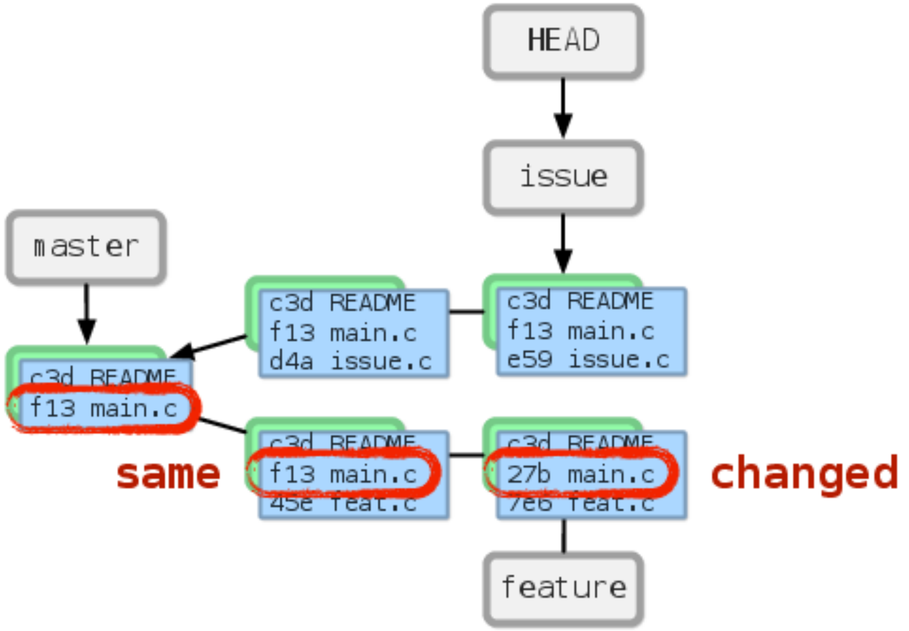
```
$ git add issue.c
$ git commit -m “Added issue.c”
```



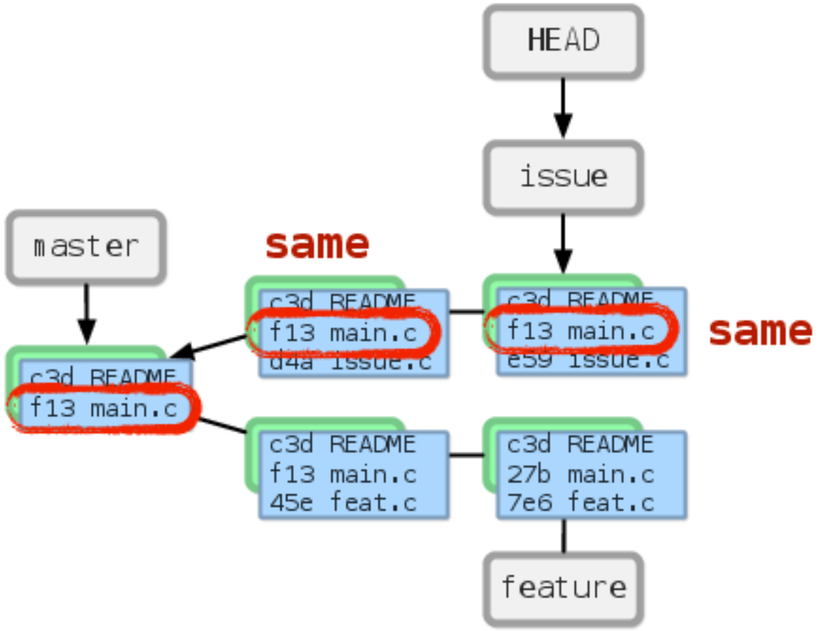
```
$ git commit -a -m "Updated issue.c"
```



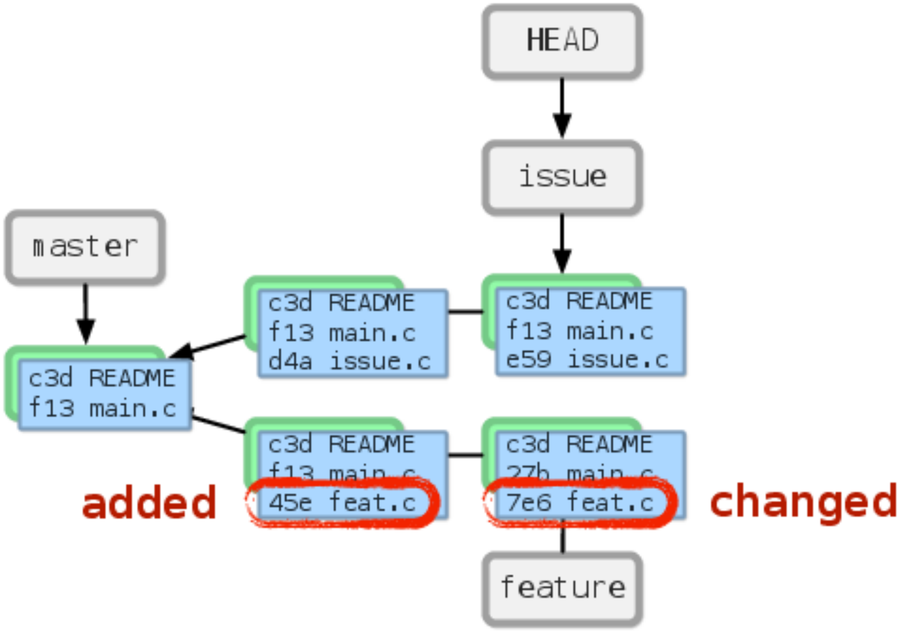
```
$ git log --stat
```


HEAD
issue
master
c3d README
f13 main.c
d4a issue.c
c3d README
f13 main.c
e59 issue.c
c3d README
f13 main.c
c3d README
f13 main.c
45e feat.c
c3d README
27b main.c
7e6 feat.c
same
changed
feature

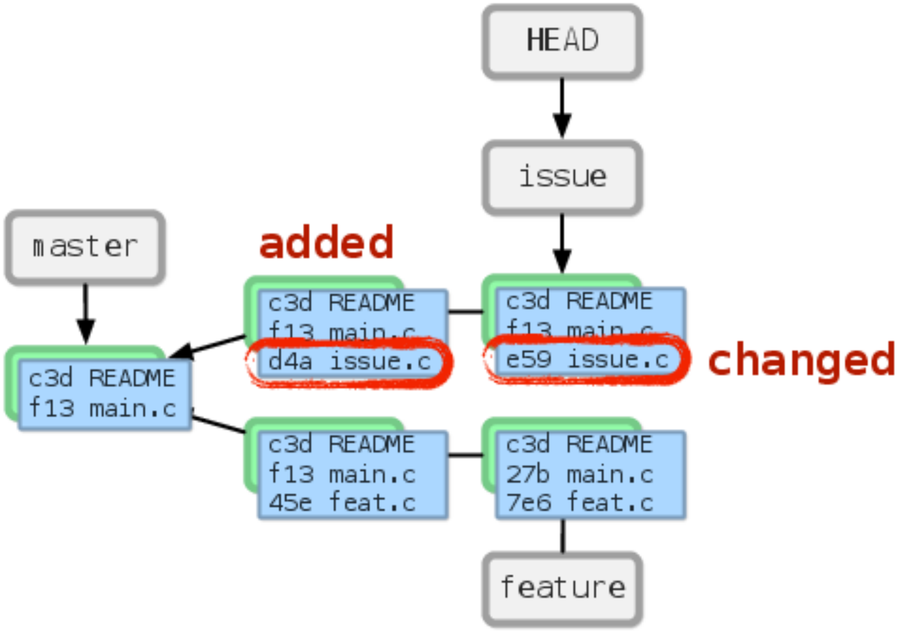
```
$ git log --stat
```

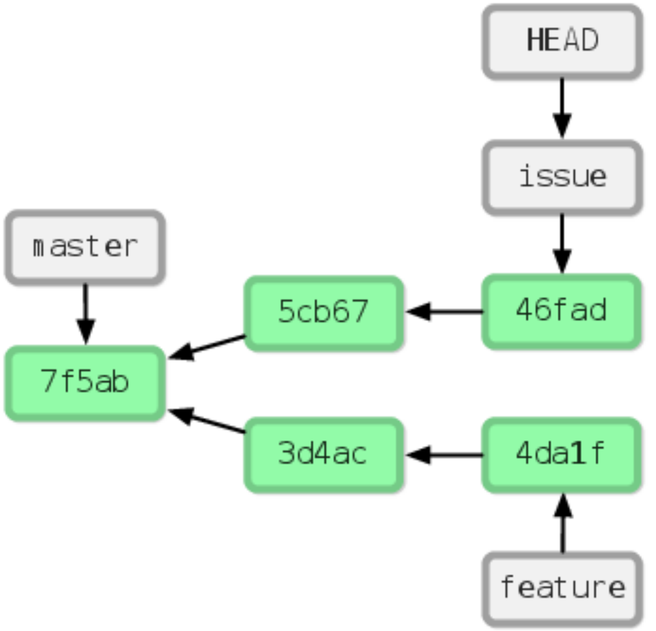

HEAD
issue
master
same
c3d README
f13 main.c
d4a issue.c
c3d README
f13 main.c
e59 issue.c
same
c3d README
f13 main.c
c3d README
f13 main.c
45e feat.c
c3d README
27b main.c
7e6 feat.c
feature

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

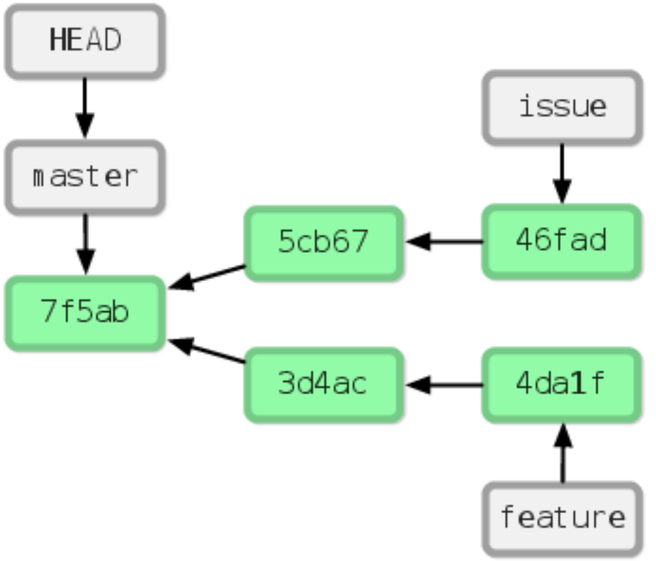

HEAD
issue
master
c3d README
f13 main.c
d4a issue.c
c3d README
f13 main.c
e59 issue.c
c3d README
f13 main.c
c3d README
f13 main.c
45e feat.c
c3d README
27b main.c
7e6 feat.c
added
changed
feature

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

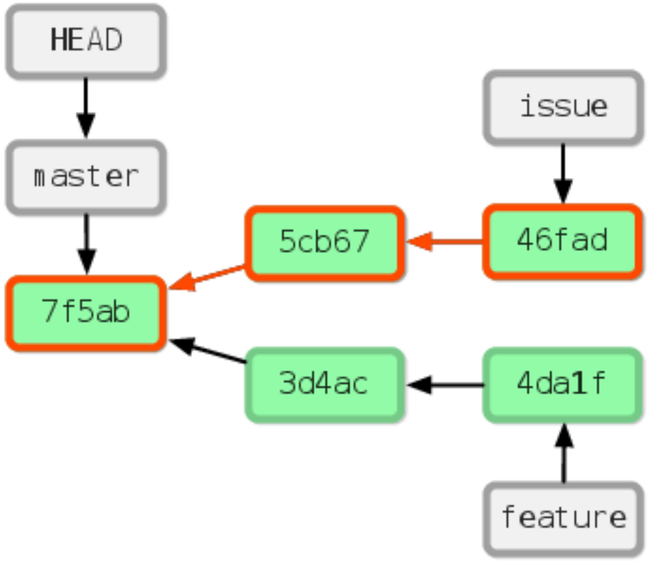

HEAD
issue
master
added
c3d README
f13 main.c
d4a issue.c
c3d README
f13 main.c
e59 issue.c
changed
c3d README
f13 main.c
c3d README
f13 main.c
45e feat.c
c3d README
27b main.c
7e6 feat.c
feature


HEAD
issue
master
5cb67
46fad
7f5ab
3d4ac
4da1f
feature

```
$ git checkout master
```

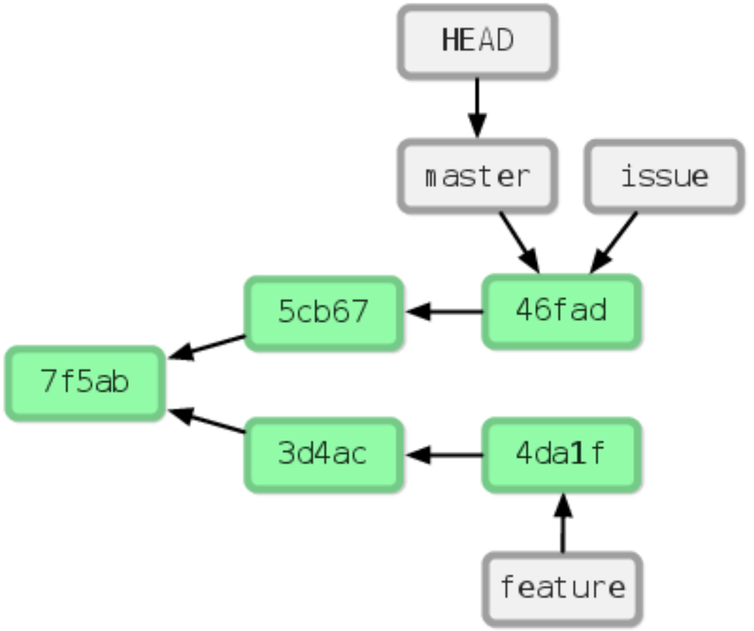


```
$ git merge issue
```

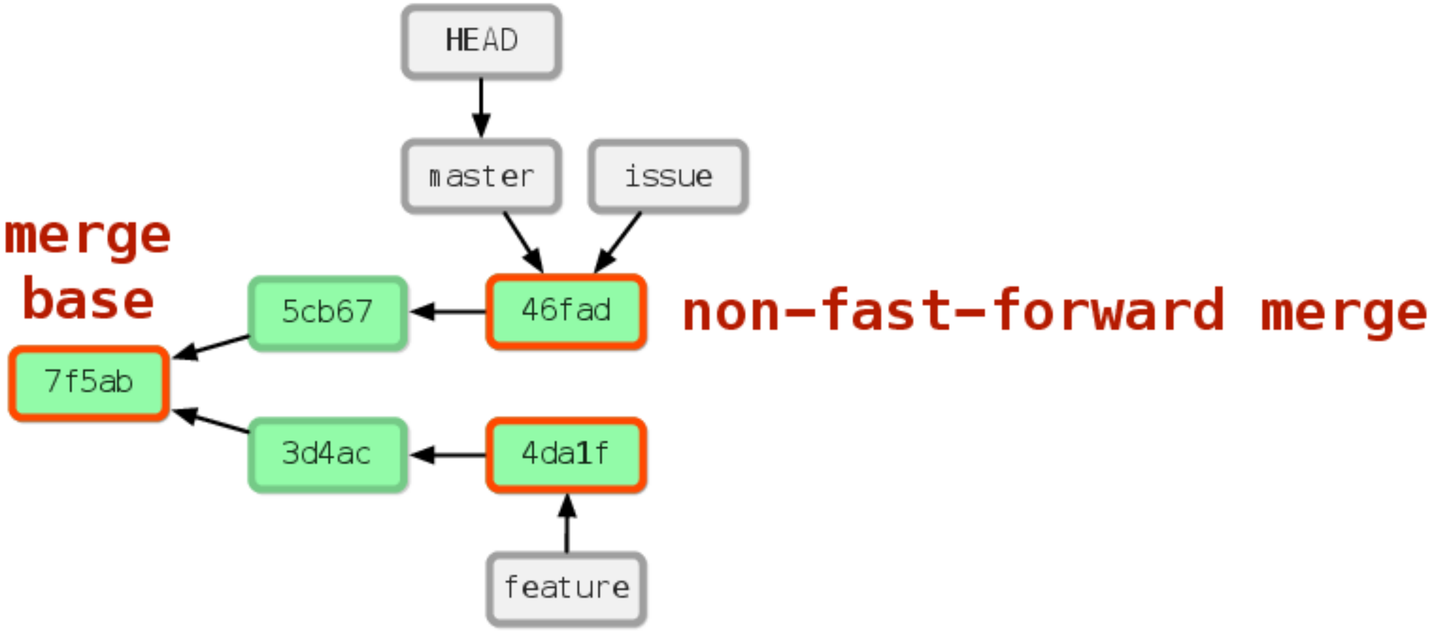



**fast-forward merge**

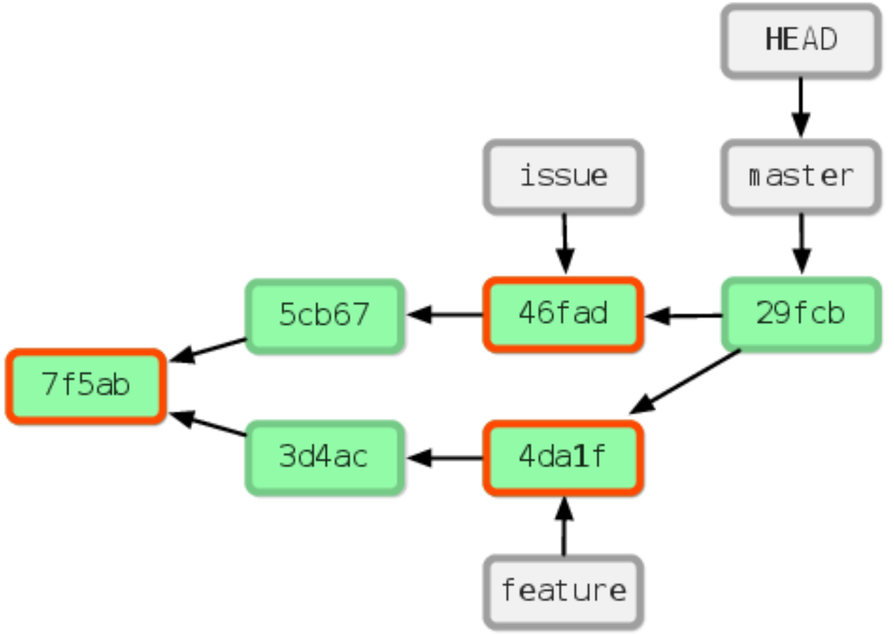
```
$ git merge issue
```



```
$ git merge feature
```

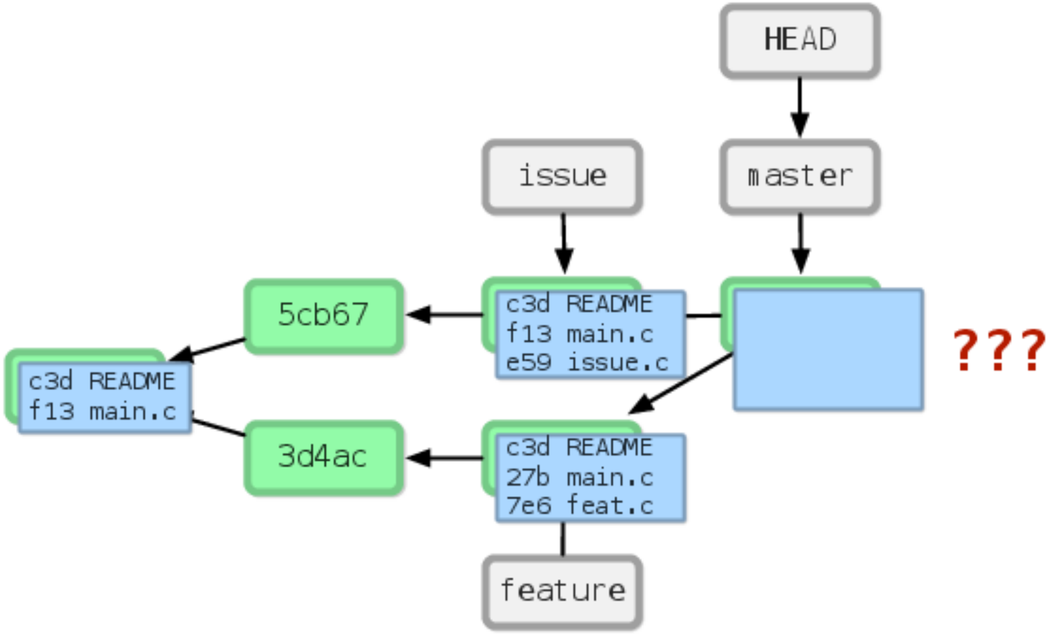


```
$ git merge feature
```

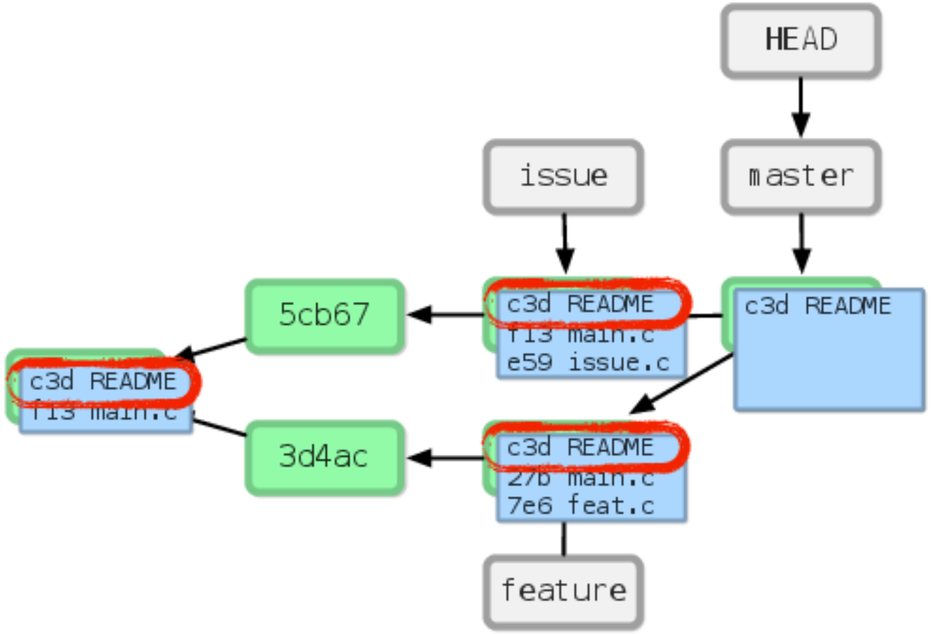



**3-way merge**

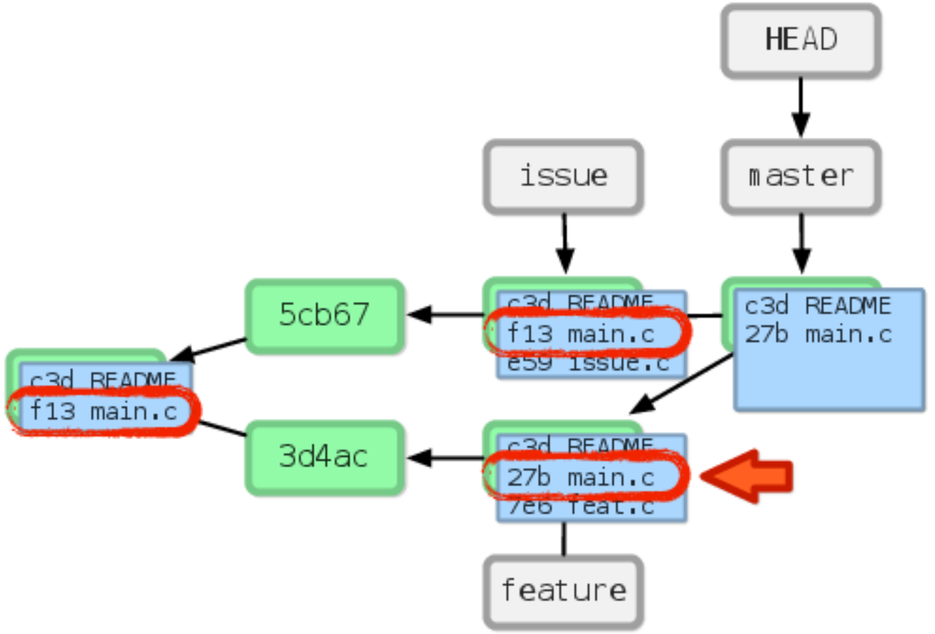
```
$ git merge feature
```



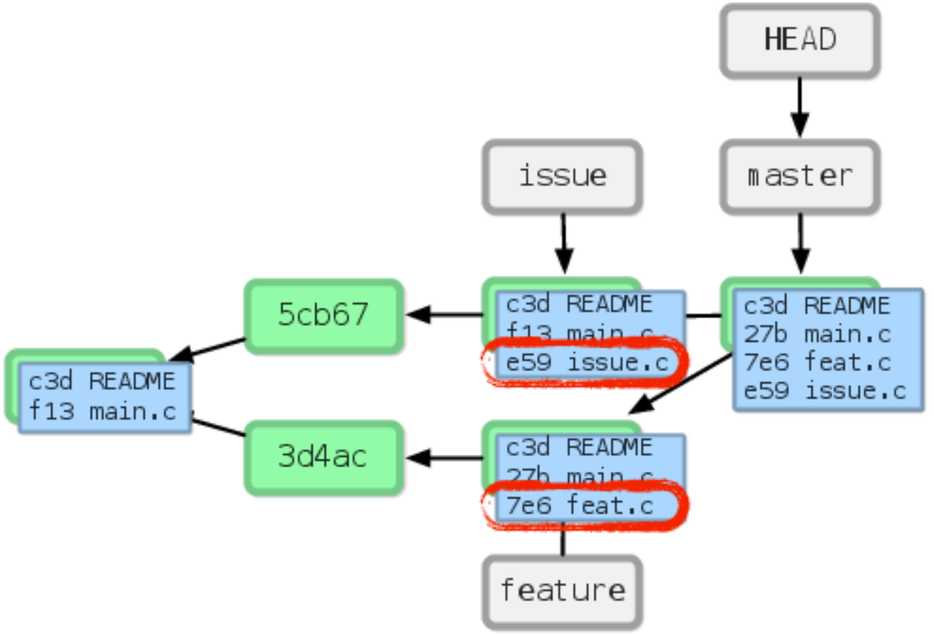
```
$ git merge feature
```



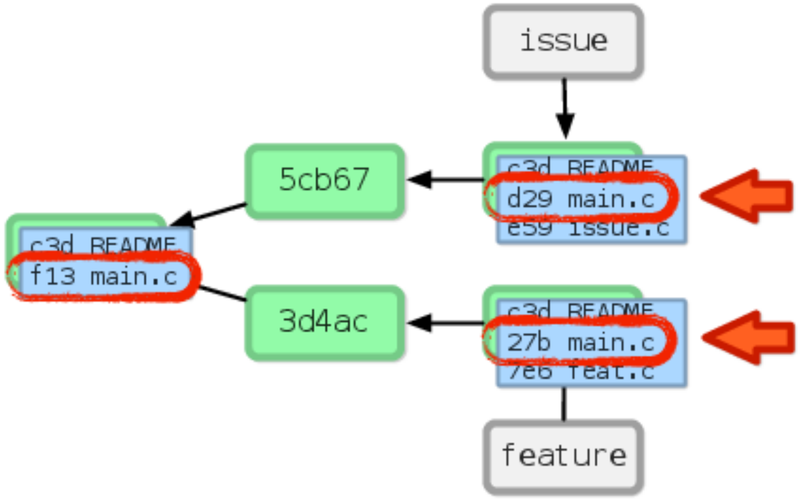
```
$ git merge feature
```



```
$ git merge feature
```



```
$ git merge feature
```




What if main.c
MERGE CONFLICT!
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")
```

MHPC
Master in High Performance Computing

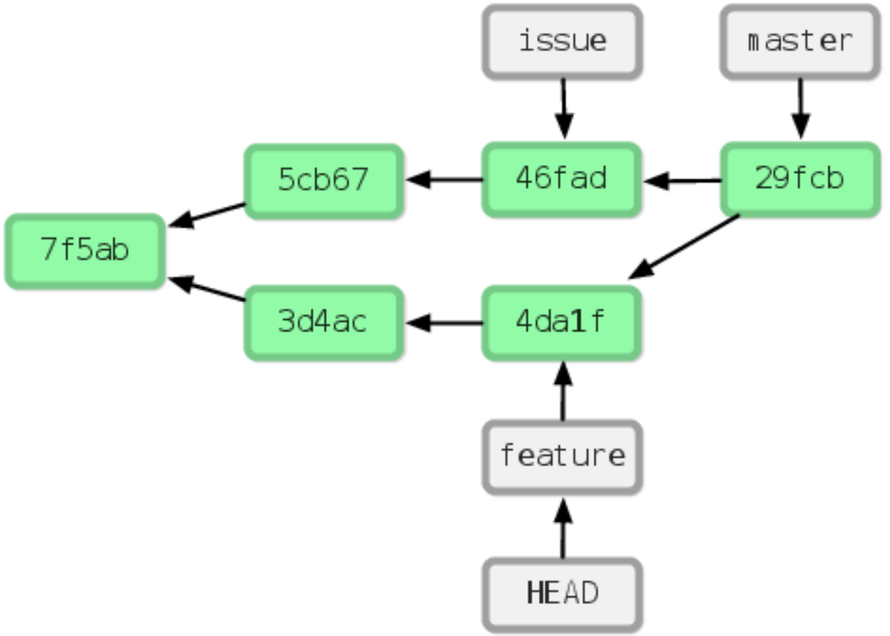
# Merge Conflict

```
$ cat main.c

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

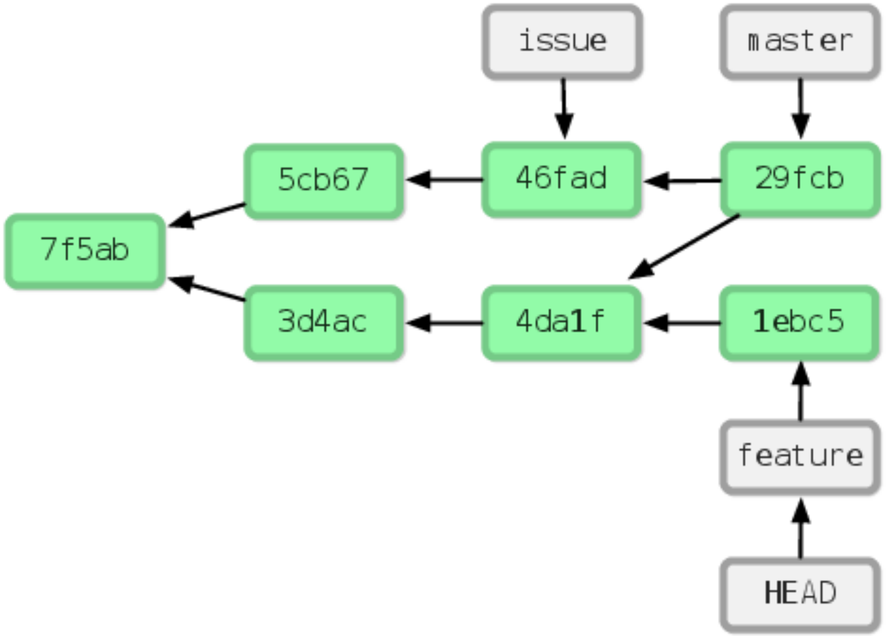
# Merge Conflict

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

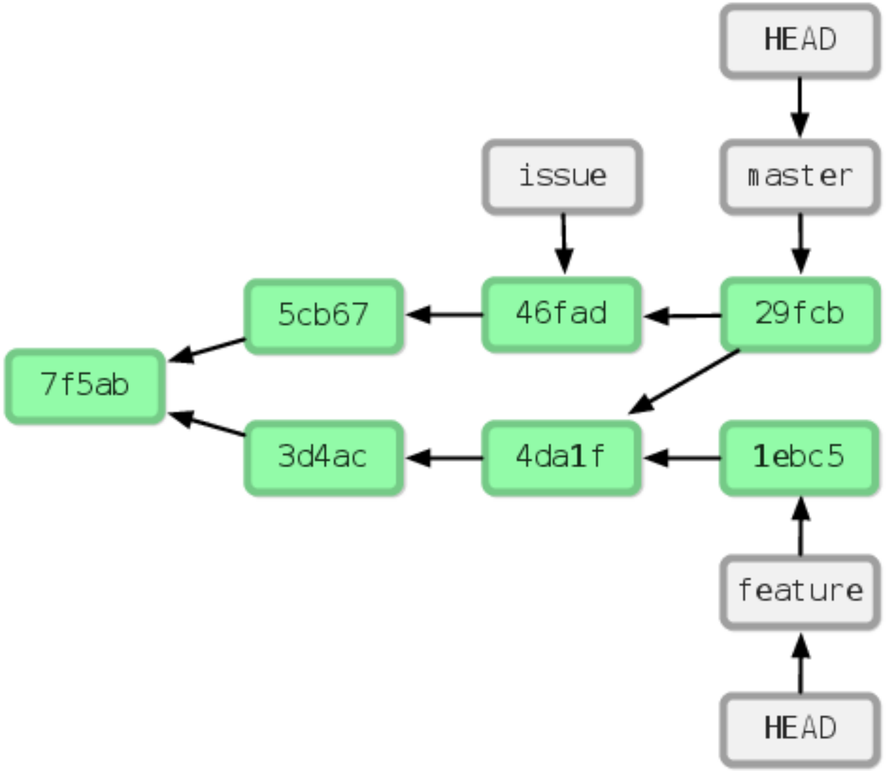
```
$ git checkout feature
```



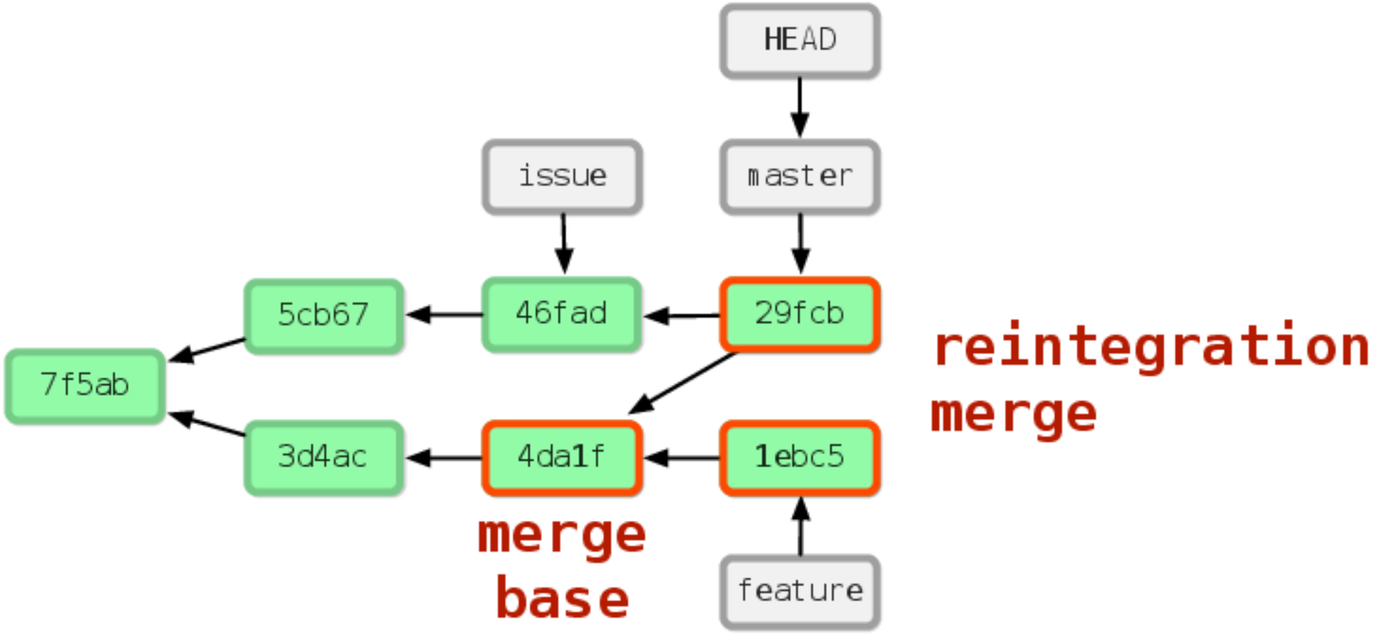
```
$ git commit -am "More on feature."
```



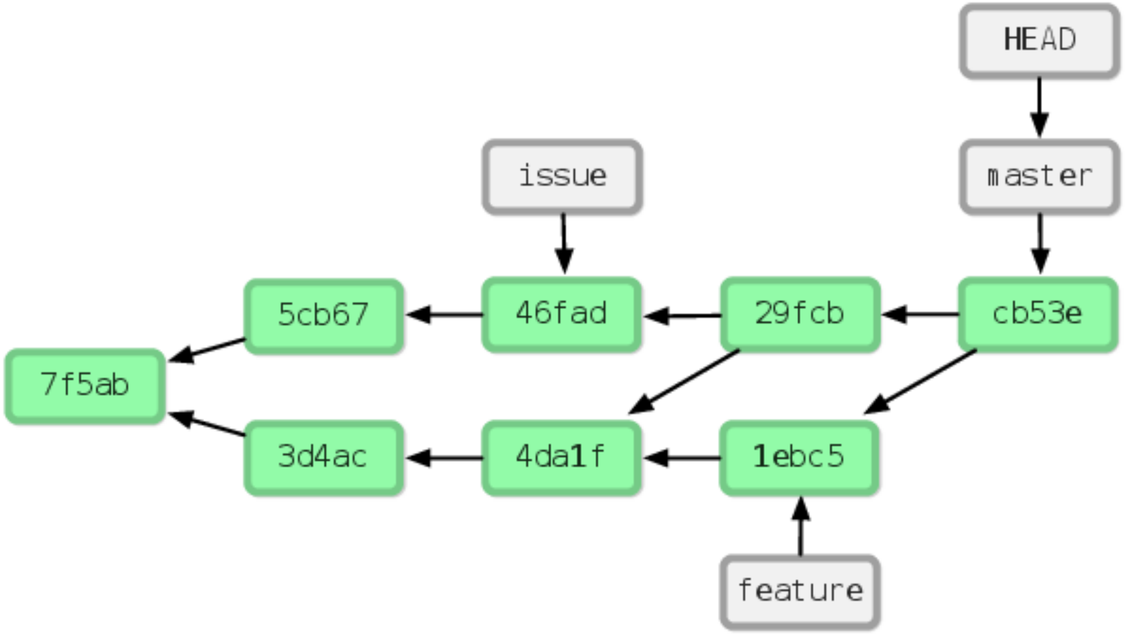
```
$ git checkout master
```



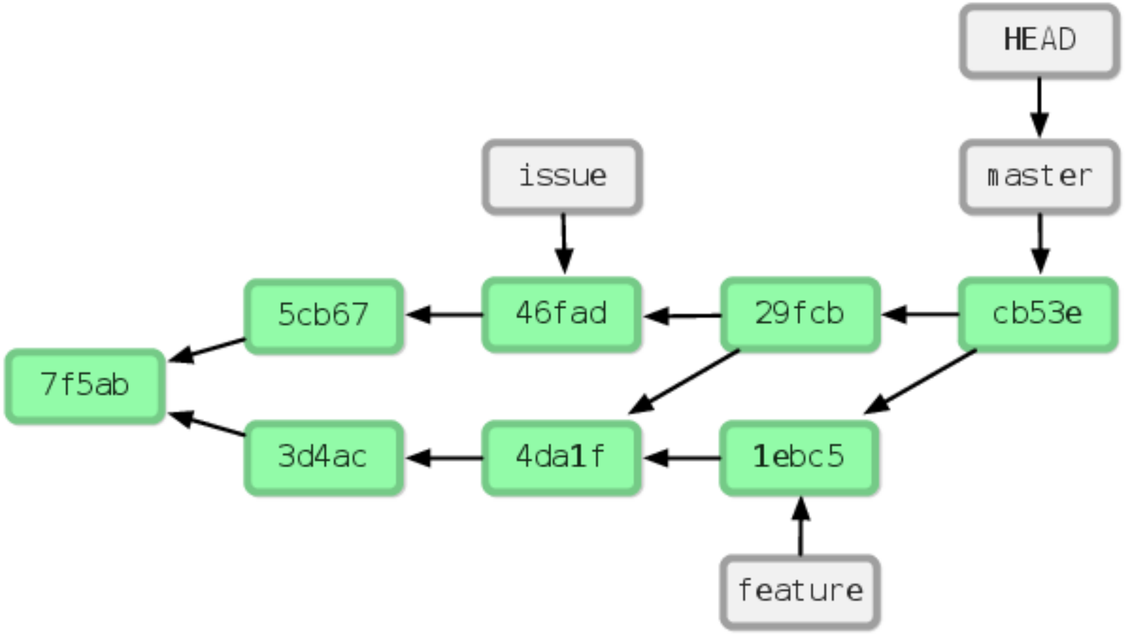
```
$ git merge feature
```



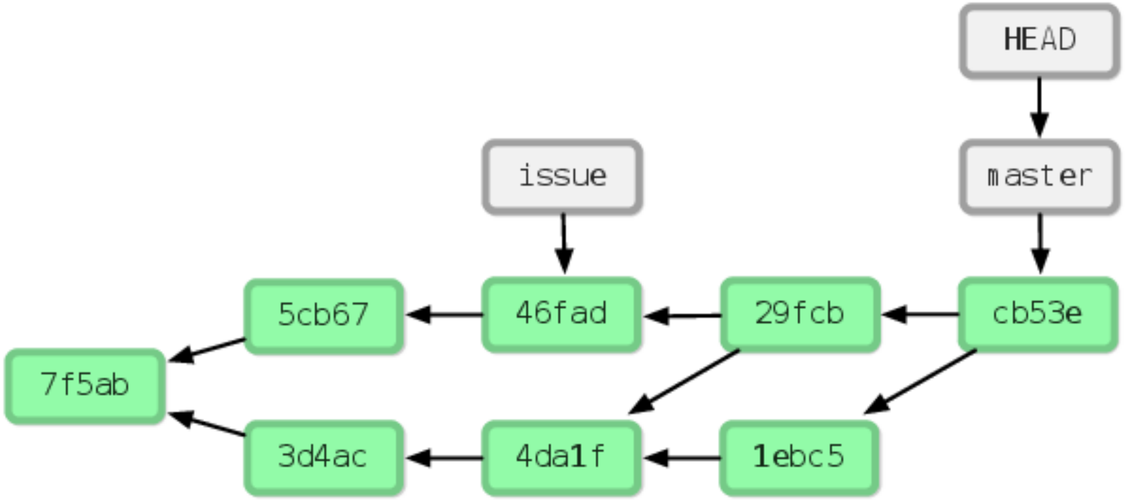
```
$ git merge feature
```



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



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



# Branching & Merging

## Delete branch with extreme prejudice.

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