# Using git

Lukas Mueller
Boyce Thompson Institute

The Practitioners Club, November 22, 2013

Most slides in this presentation are based on the Pro Git Book available on line at http://git-scm.com

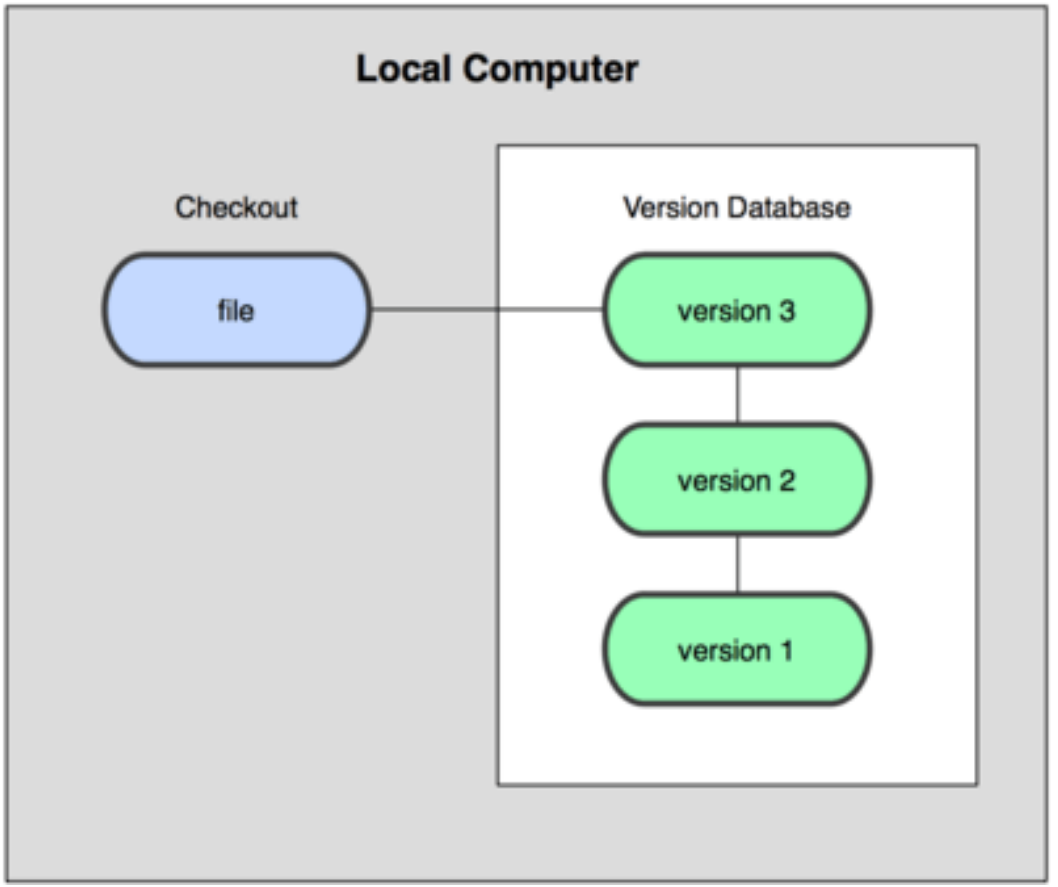
# Version Control Systems (VCS)

- Why is it useful?
  - Backup of your code
  - Help in debugging (track changes)
  - Better organization
  - Collaborate on code
  - Share code

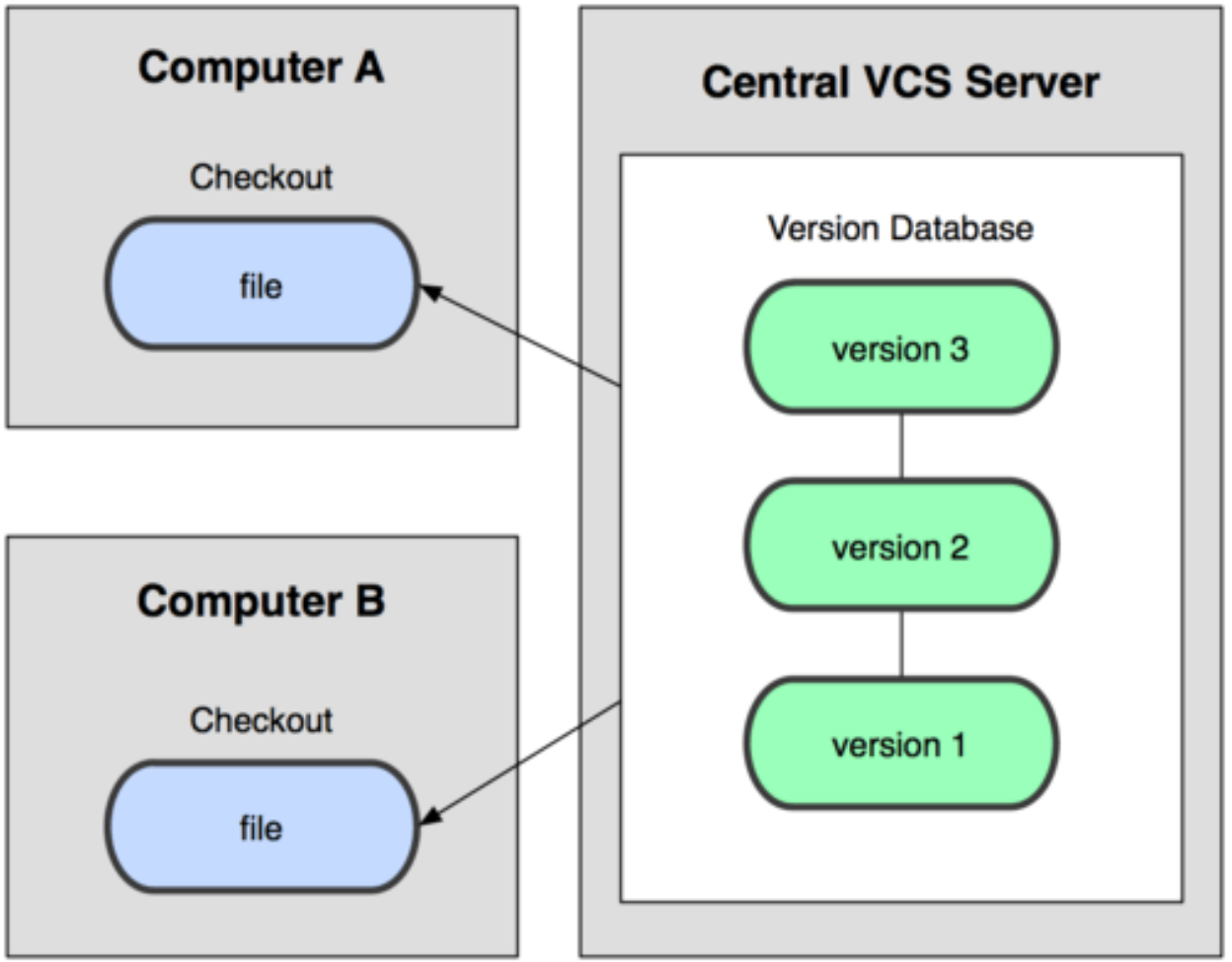
# VCS systems

- Cvs (concurrent version system)
- Subversion (svn)
- BitKeeper
- Mercurial
- Git (since 2005)

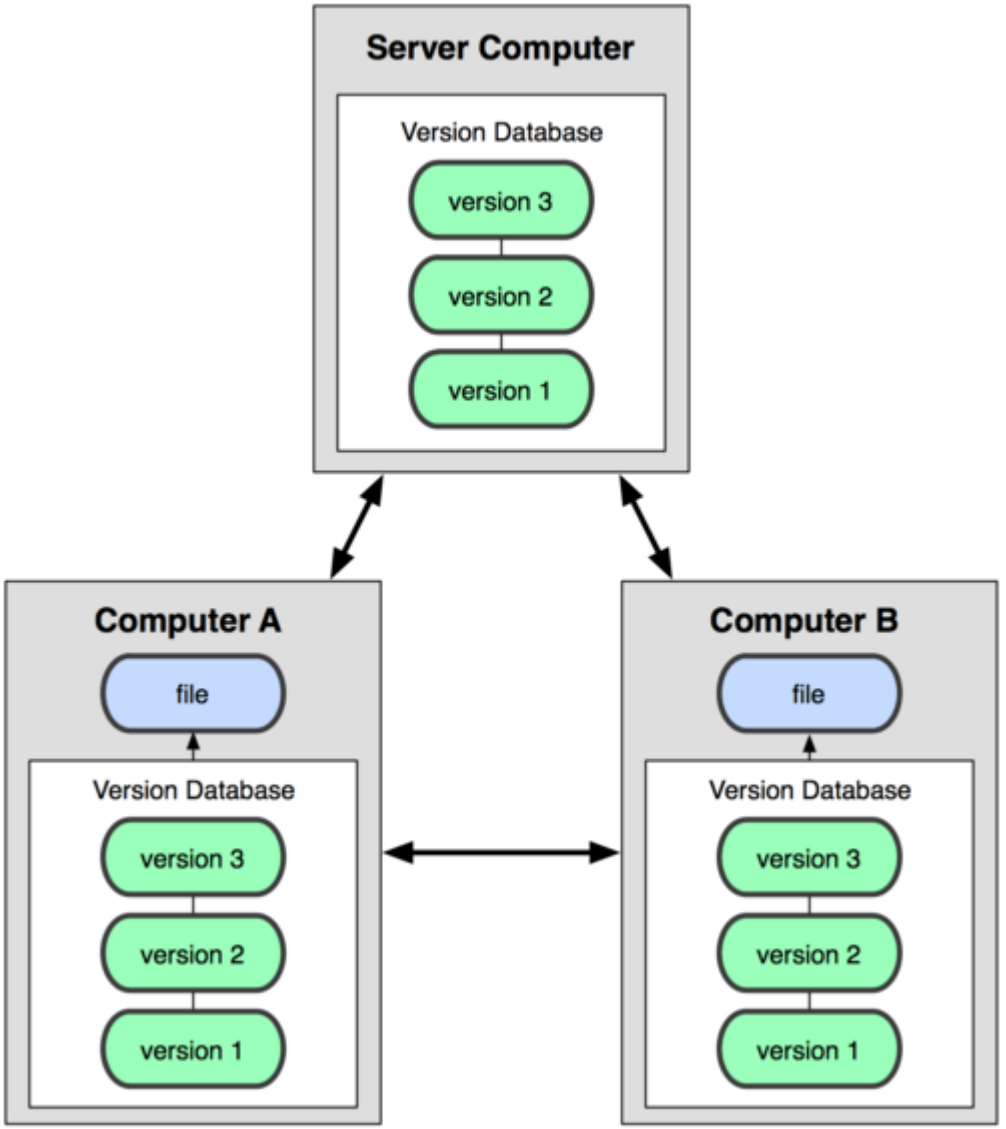
# Local version control



# Central


Computer A
Checkout
file
Computer B
Checkout
file
Central VCS Server
Version Database
version 3
version 2
version 1

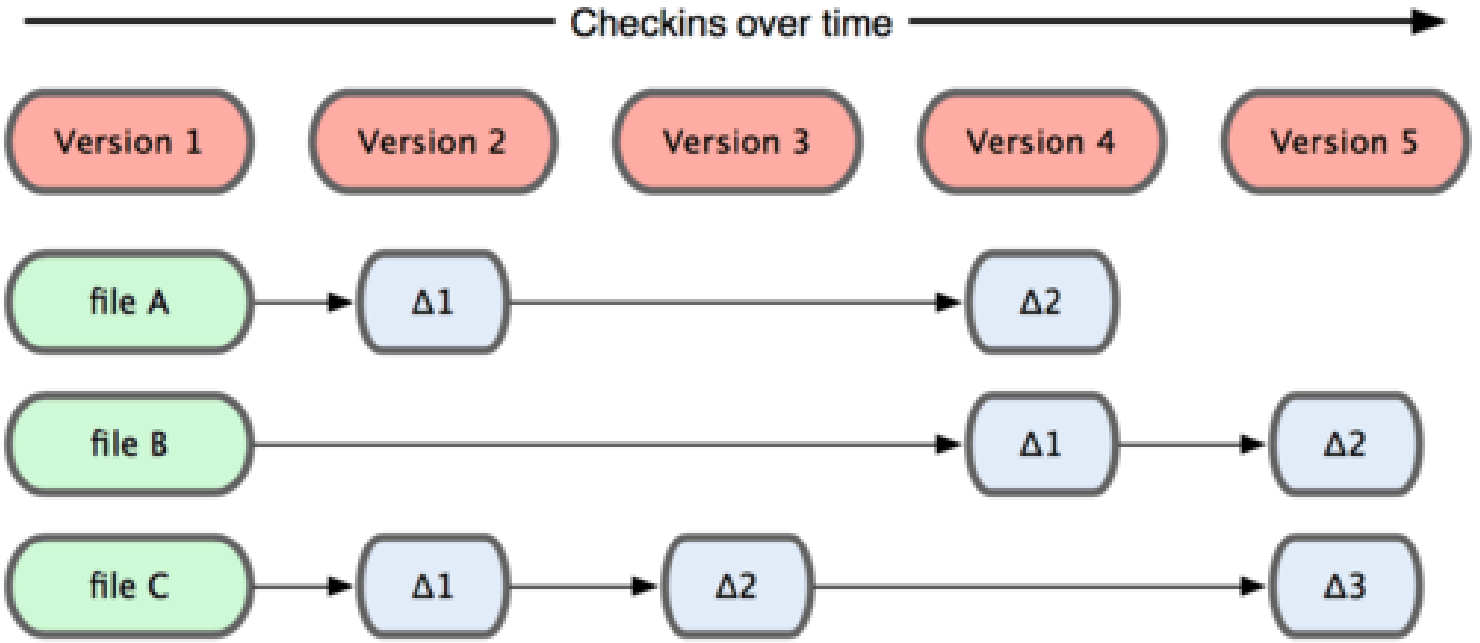
# Distributed



# Advantages of the distributed model

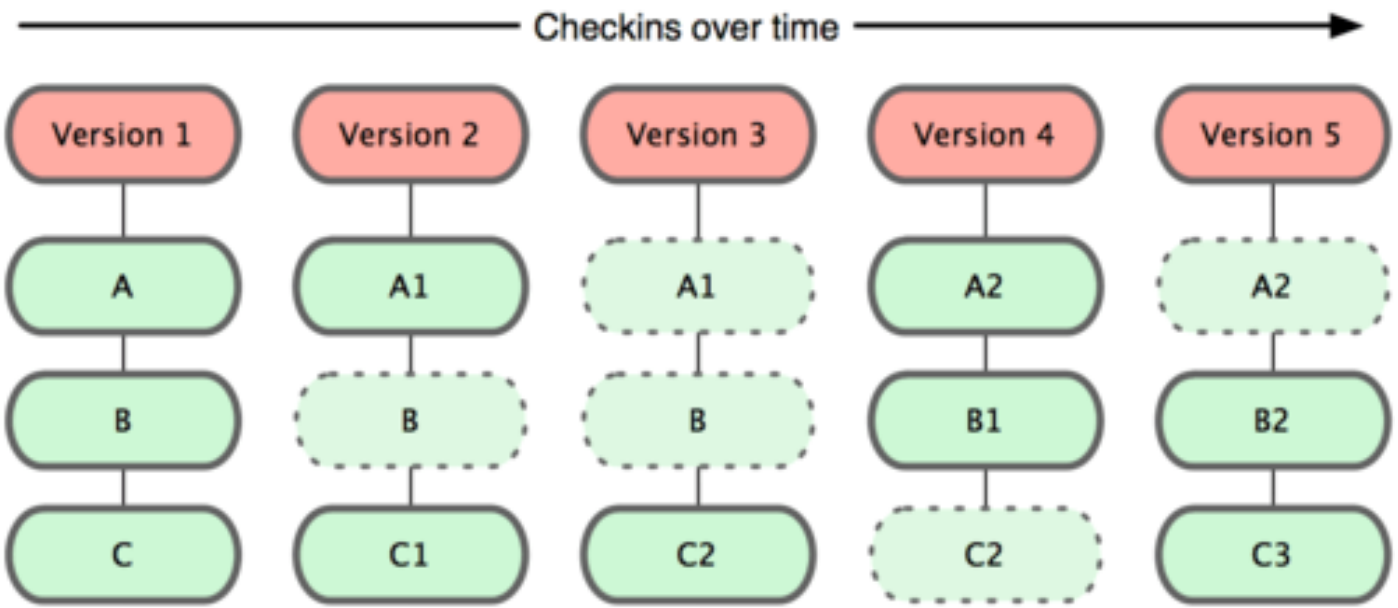
- Every repository contains entire history
- Fast access to all history-related functions
- Work offline and commit!

- Most VCS deal with diffs

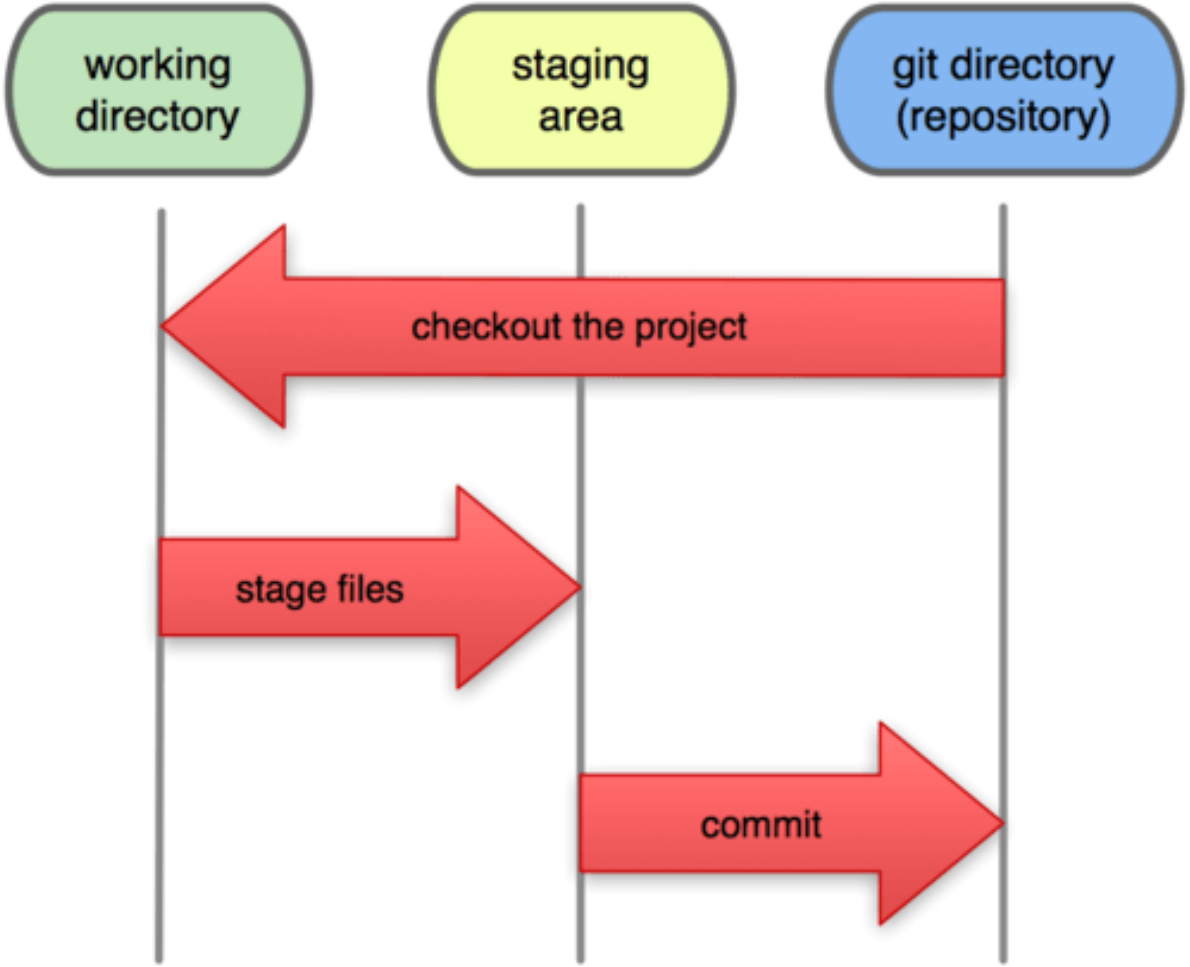


# Integrity

- Git deals with snapshots instead of diffs



# Local Operations



# Installing Git

- Debian/Ubuntu
  - `apt-get install git`
- Other platforms:
  - http://git-scm.com/book/en/Getting-Started-Installing-Git

# Configuration

- `/etc/gitconfig` – system level config (`--system`)
- `~/.gitconfig` – per user config (`--global`)
- `.git/config` – per repo config

- Add your identity

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

- Favorite editor, etc.

```
$ git config --global core.editor emacs
```

# Getting help

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

- IRC
  - irc.freenode.net #git or #github

# Initializing a project

- Create project directory and initialize
  - `mkdir project`
  - `cd project`
  - `git init`
- Edit a file
  - `emacs README.txt`
  - `git add README.txt`
  - `git commit -m 'add a README file'`

# GitHub

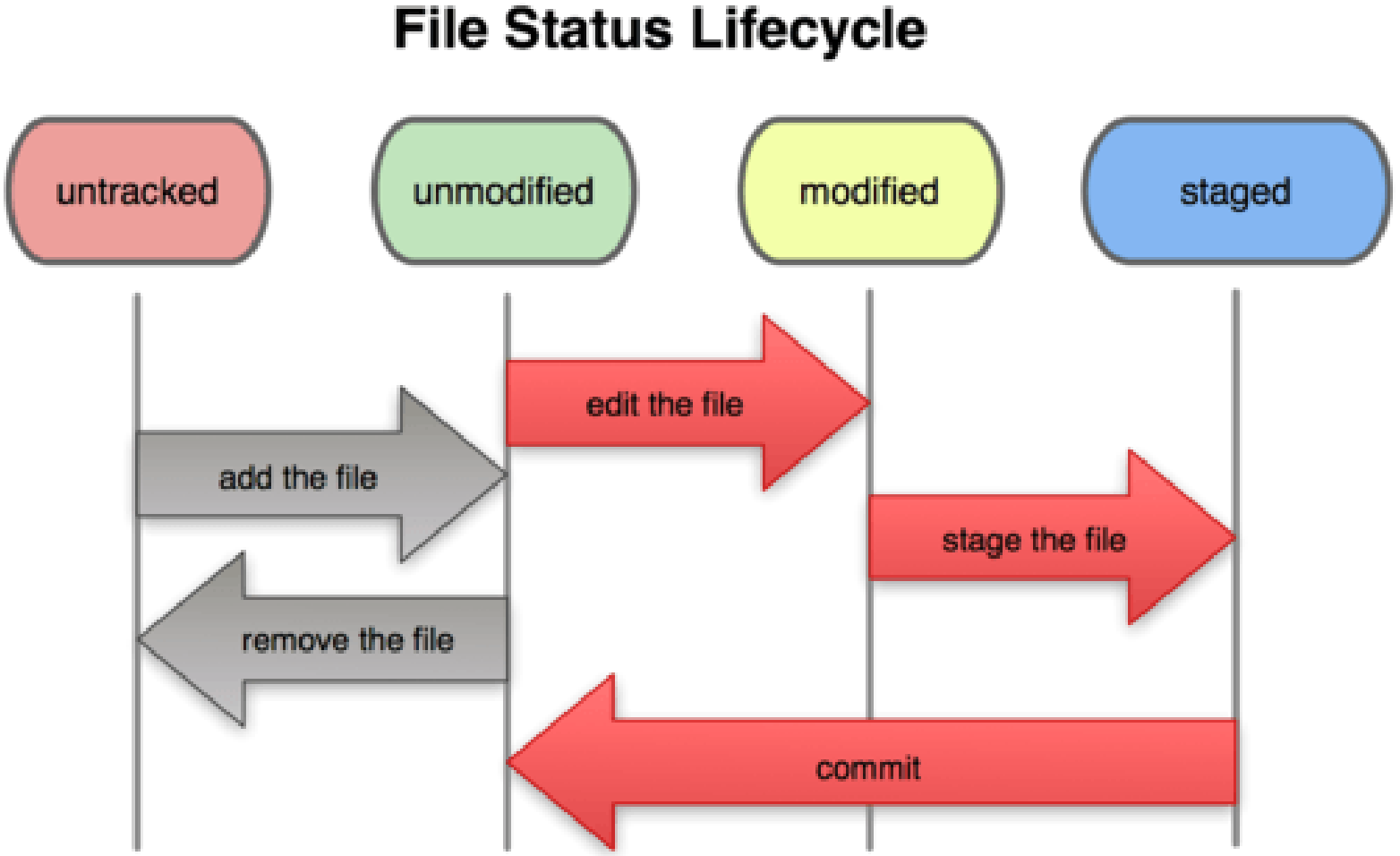
- Github (http://github.com) is a commercial website providing project hosting based on git
- Easy to use, web-based user interface and native apps for OSX and Windows
- Basic accounts with up to 600MB of public data are free
- More storage or private repositories require a monthly subscription

# Pushing to a remote repo

- Using our github account as remote (“origin”)
  - `git remote add origin https://github.com/lukasmueller/test2.git`
  - `git push -u origin master`

# Using github to create a project

- Create a github account
- Log in
- Click on new repo logo (next to username)
- Fill in form
- Use `git clone` to clone the repo to your local machine


File Status Lifecycle
untracked
unmodified
modified
staged
edit the file
add the file
stage the file
remove the file
commit

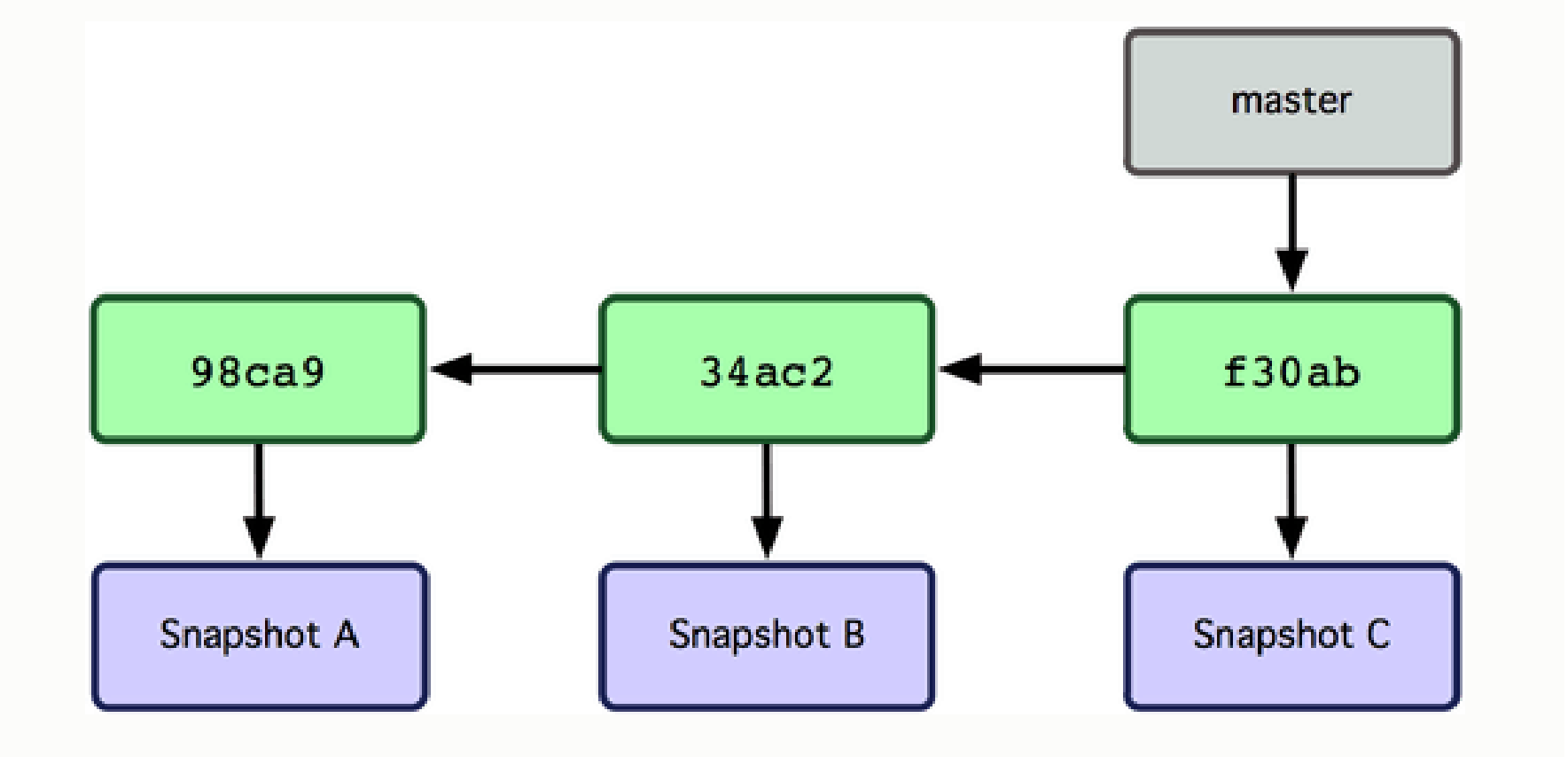
# Interactive demo

- Topics
  - Editing a document
  - `git status`
  - `git diff`
  - `git add, git commit`
  - `git rm`
  - `git mv`
  - `git log –stat`
  - `git reset HEAD <file>`
  - `git checkout <file>`
- gitignore

# Tags

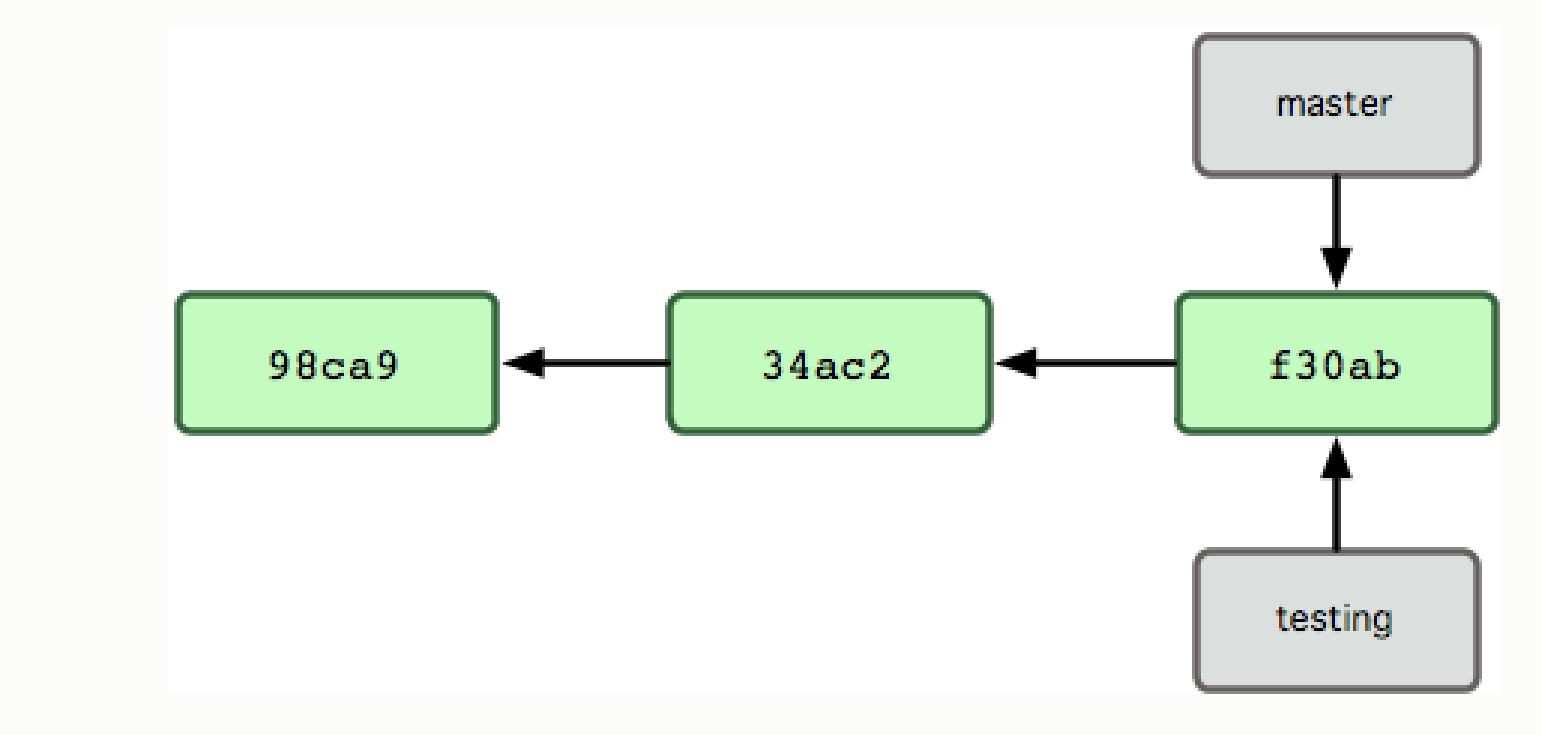
# Branches

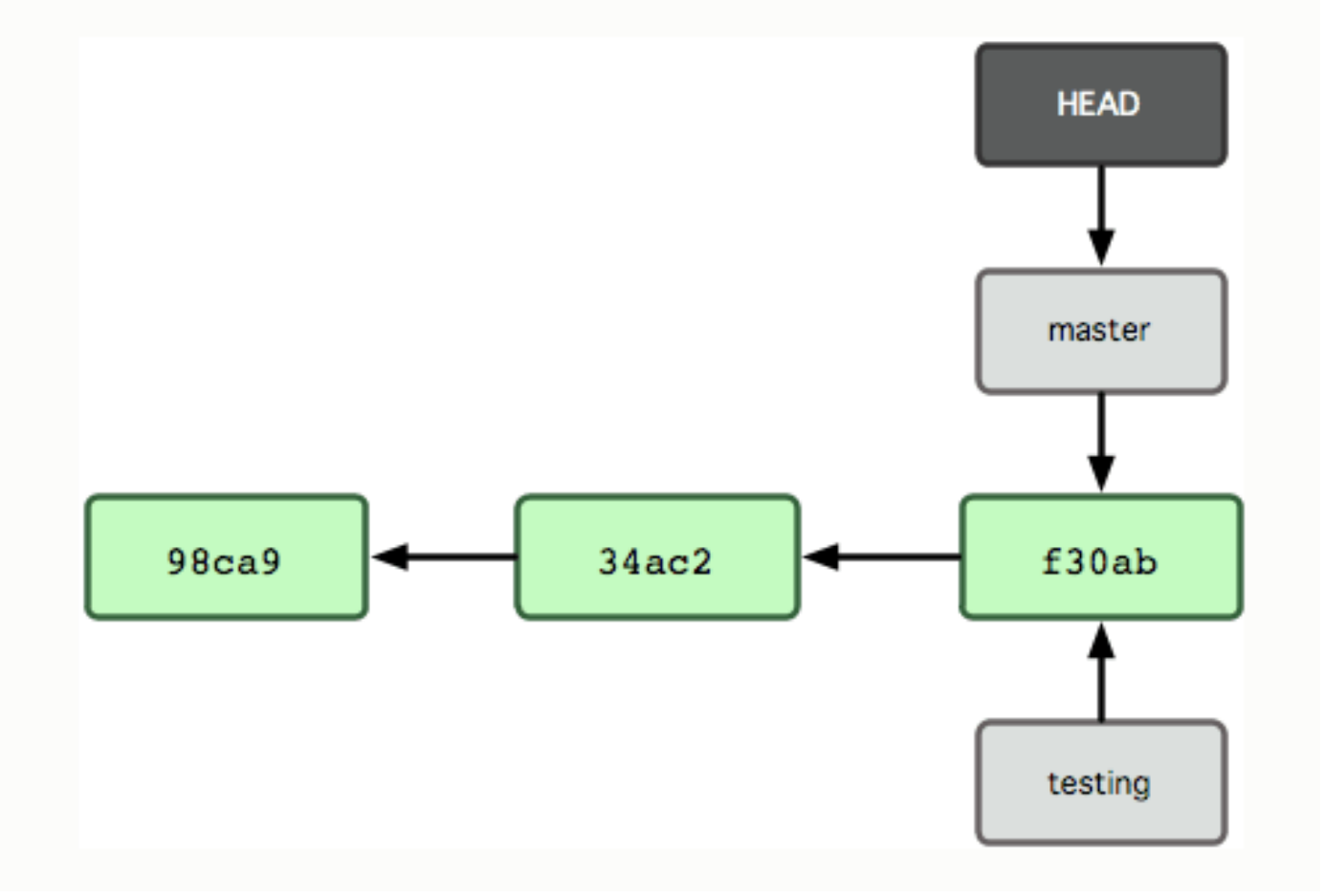
- A branch is a pointer to a commit



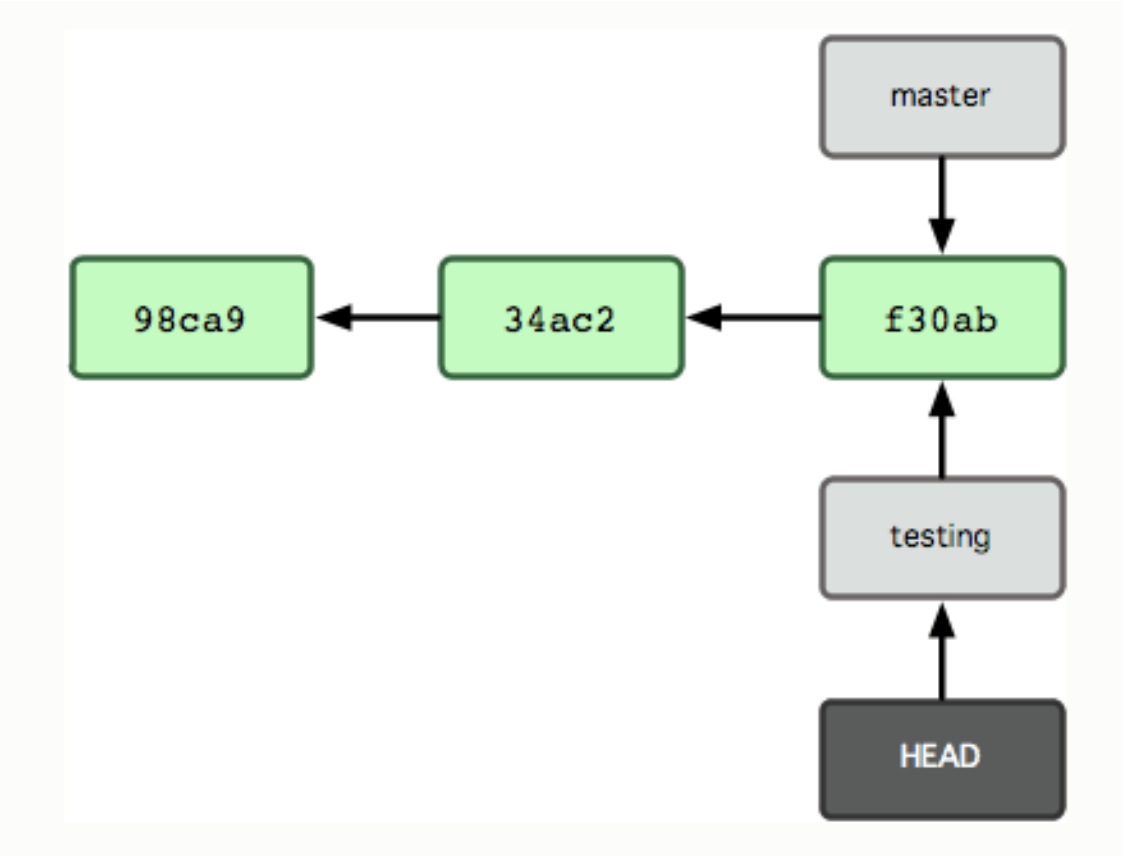
# Create a new branch

- `git branch testing`

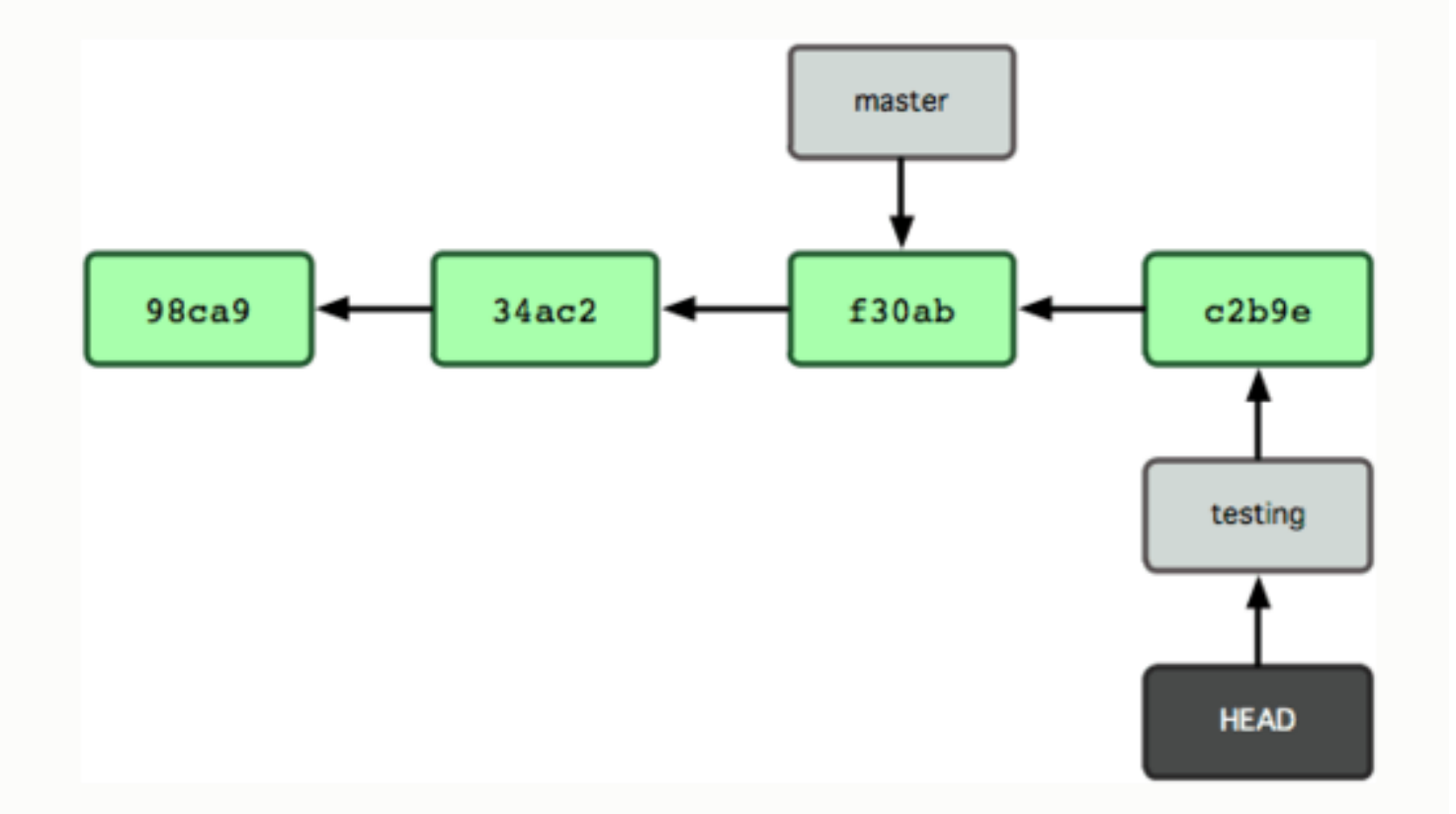



HEAD
master
98ca9
34ac2
f30ab
testing

- `git checkout testing`

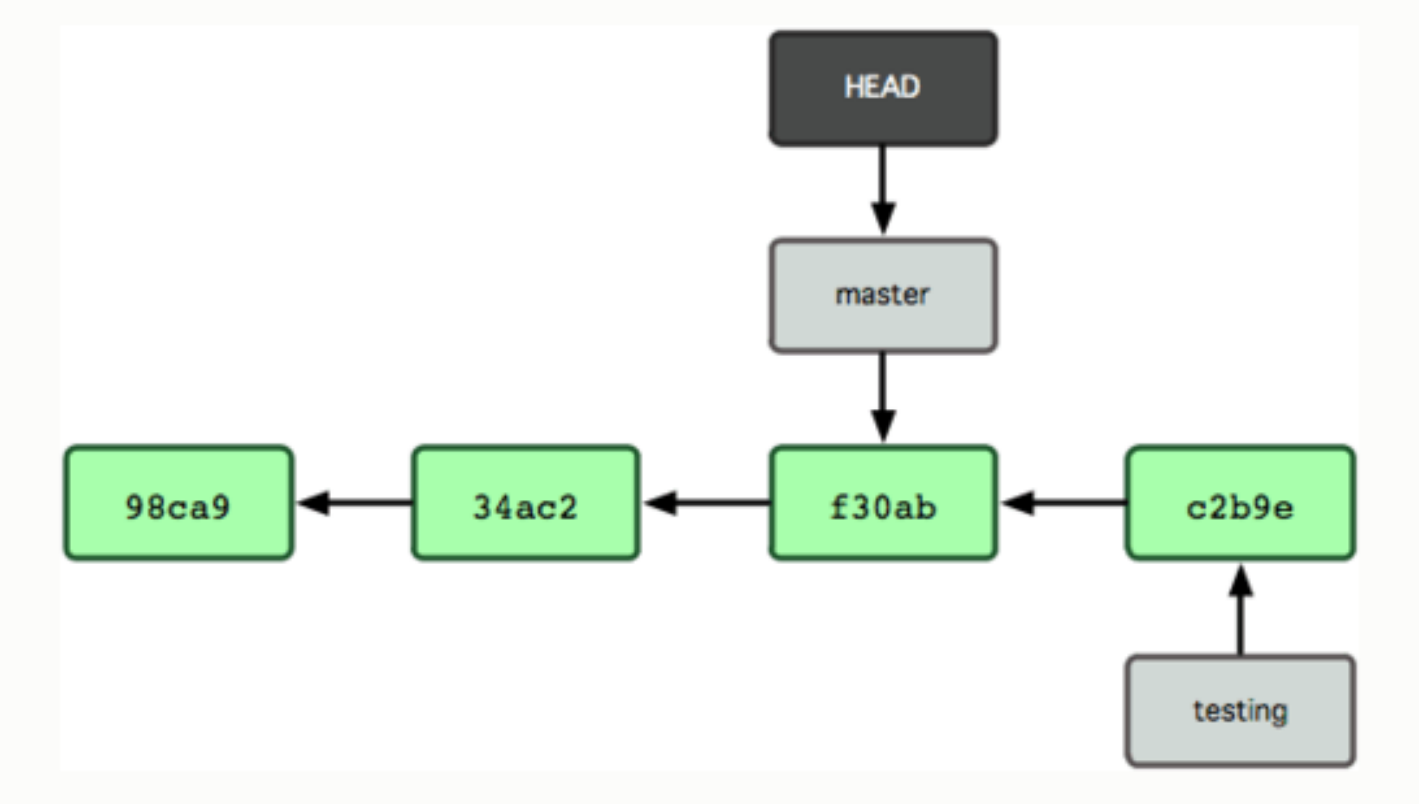


# Commit to testing

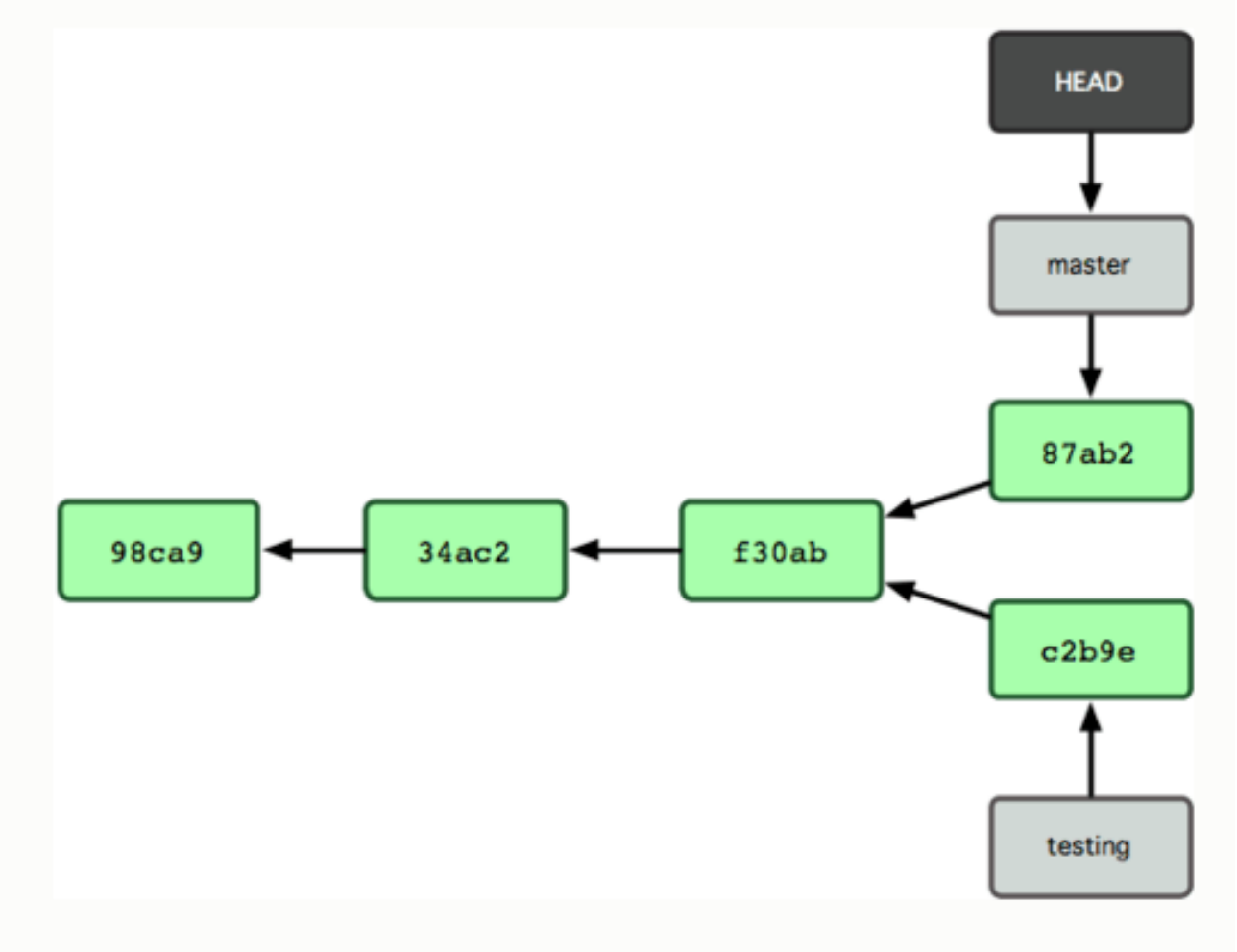


# Switch back to master

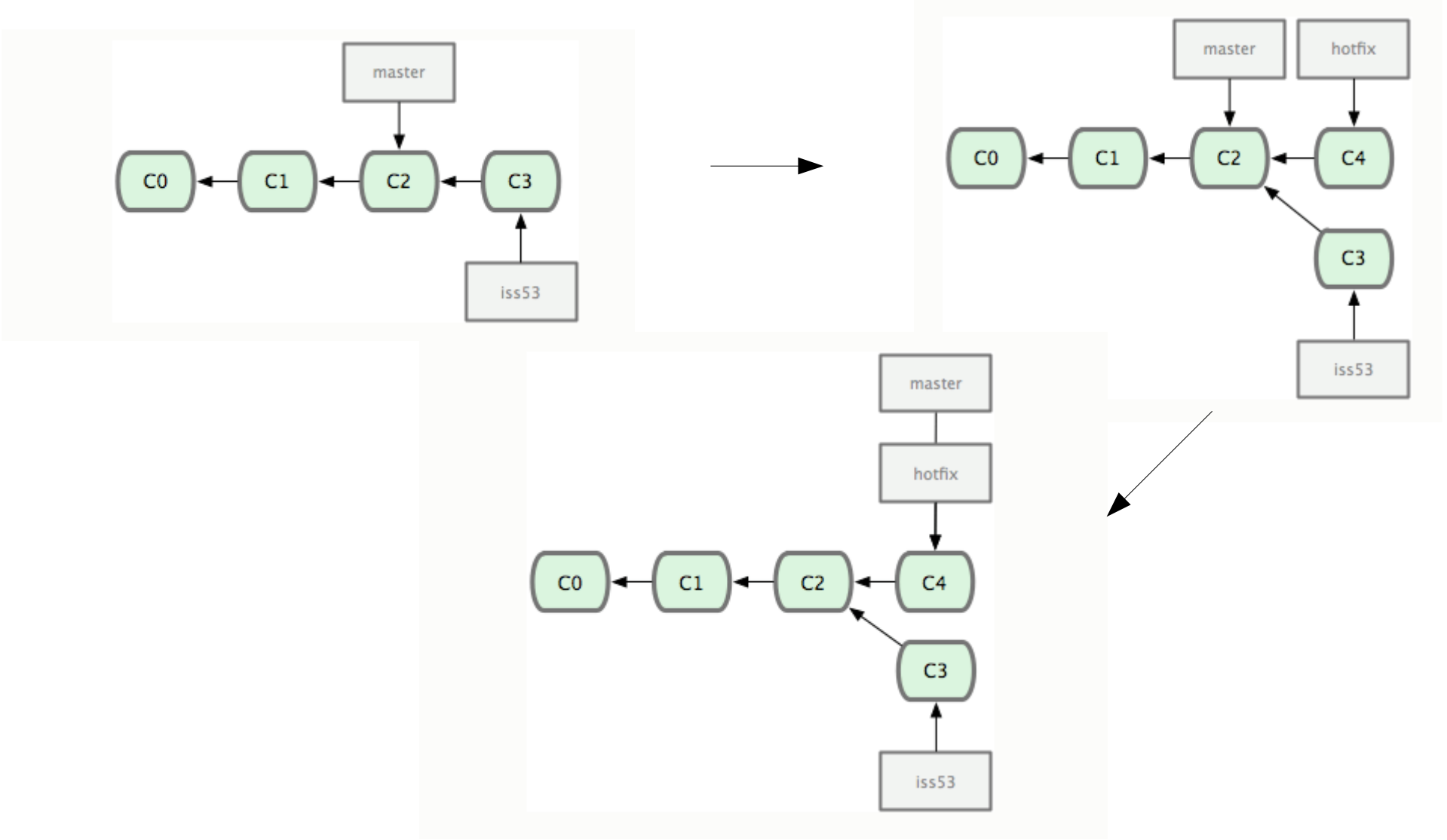
- git checkout master



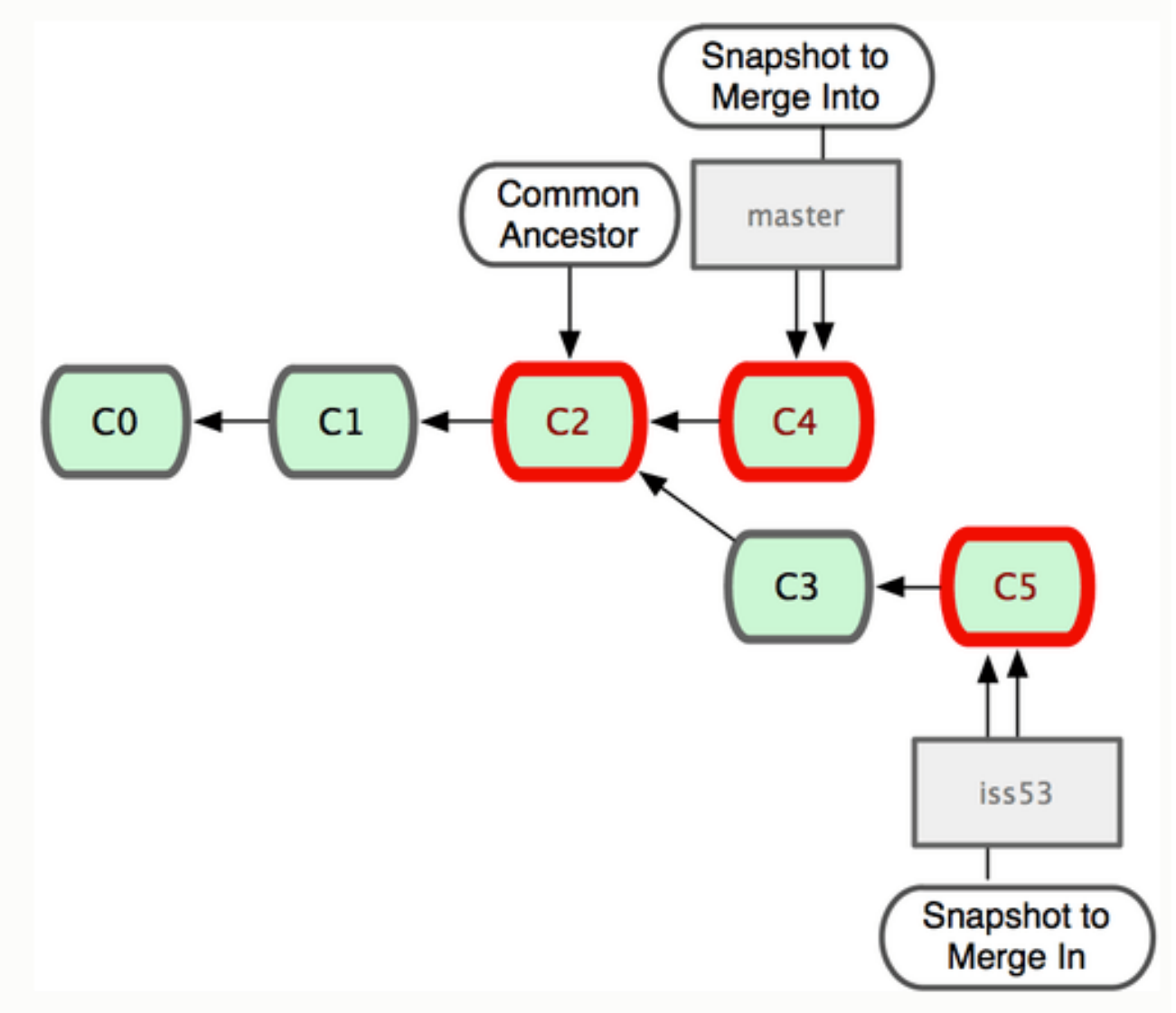
# Commit to master



# Fast forward merge

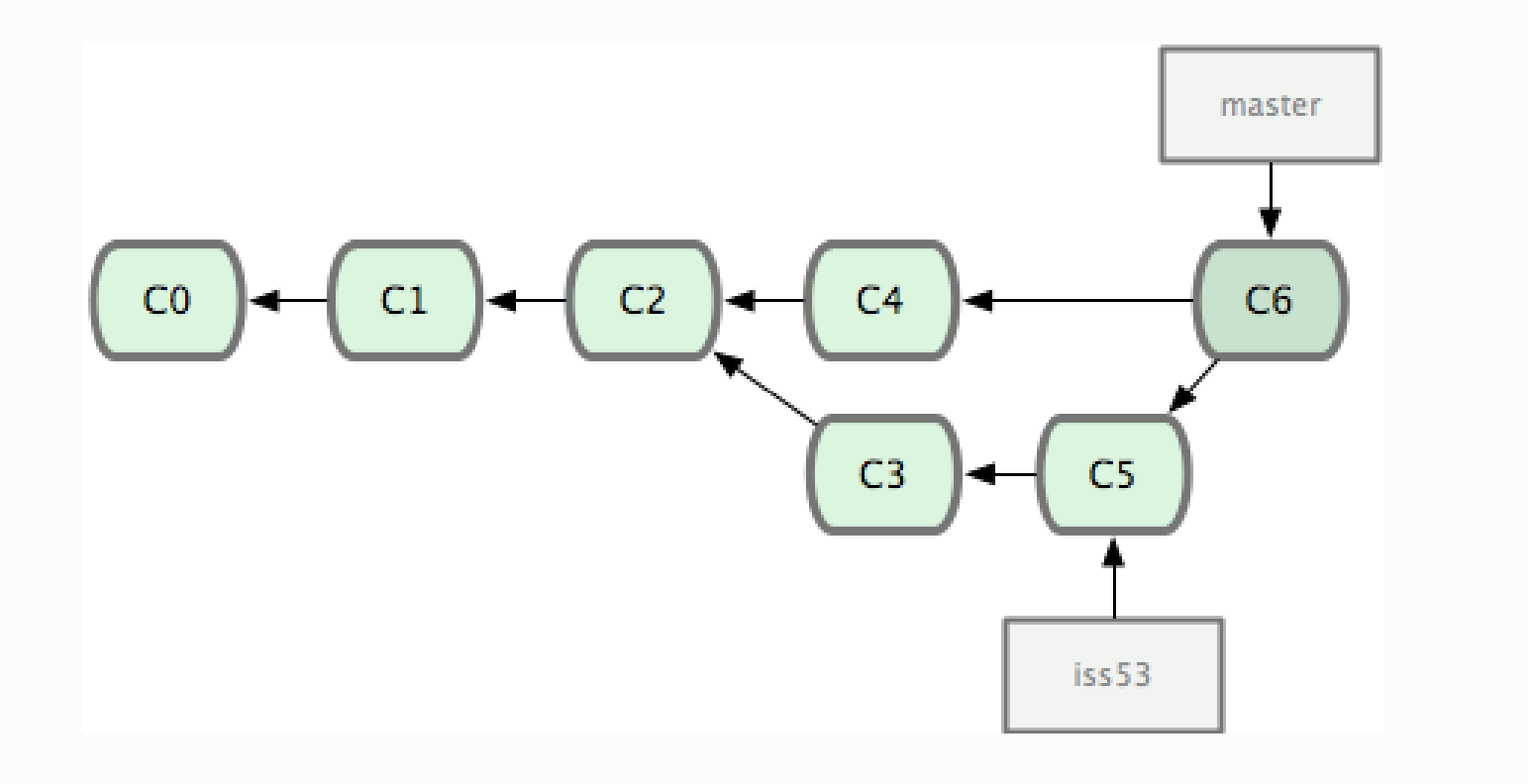


# Three way merge



# Merge commit

- Git generates a new merge commit automatically



# Merge conflicts

- Git will report a list of files with conflicts
- The conflicted areas in the file are denoted with markers
- Edit the file to the desired status, remove conflict markers
- `git add <file>`
- `git commit`

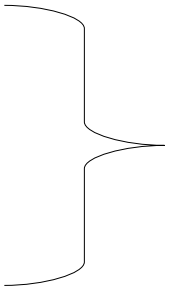
# Summary of branches

- `git branch <new-branch>`
- `git checkout <new-branch>`
- Edit files...
- `git add <file>`
- `git commit <file>` # now goes to branch
- `git checkout master`
- # changes are no longer present
- `git merge <new-branch>`
- `git branch` # list all the branches

# Remote branches

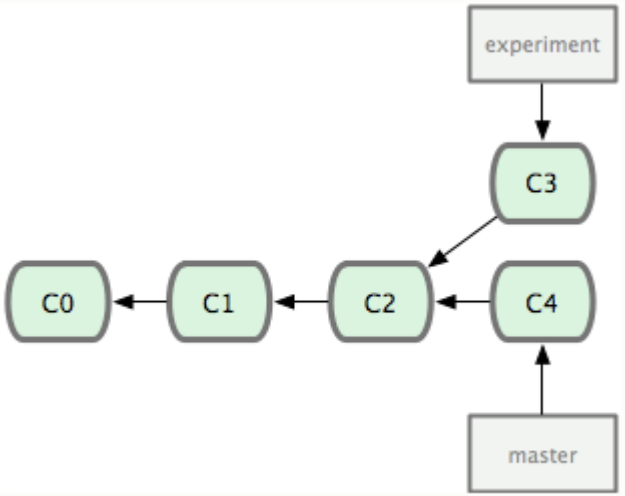
- Same as local branches, really.
- Note the shortcut:

- `git fetch`
- `git merge`



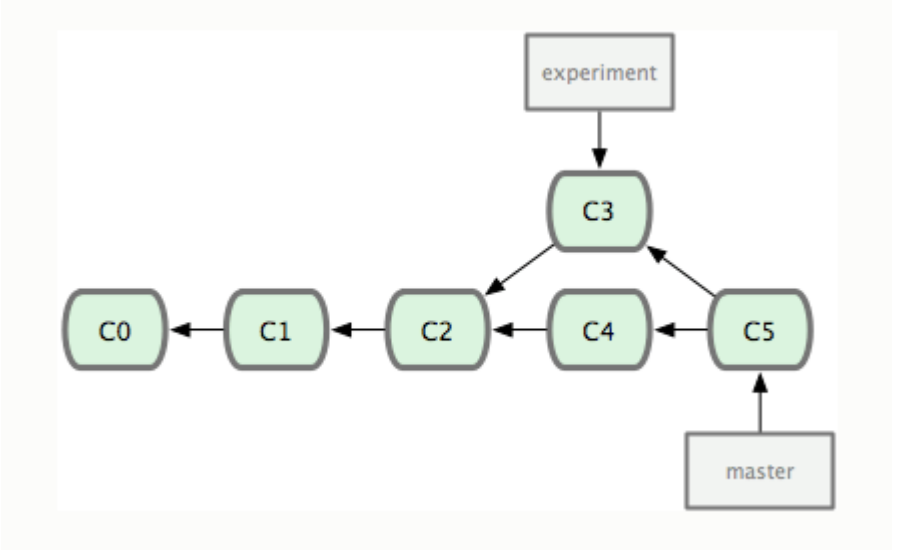
- `git pull`

# Rebasing

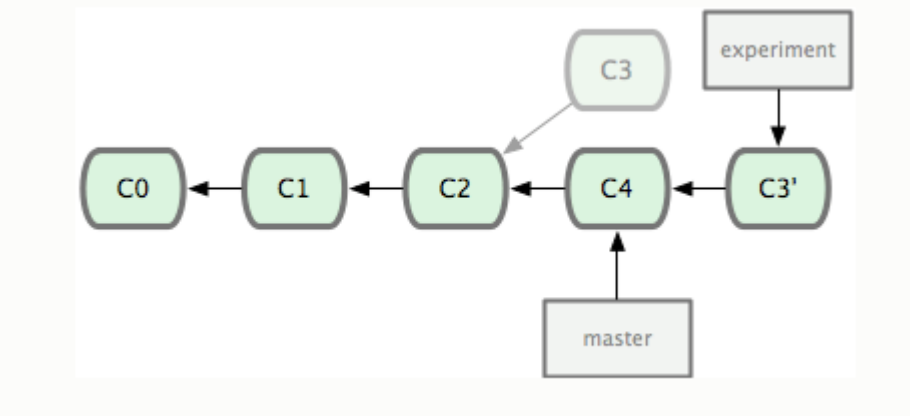


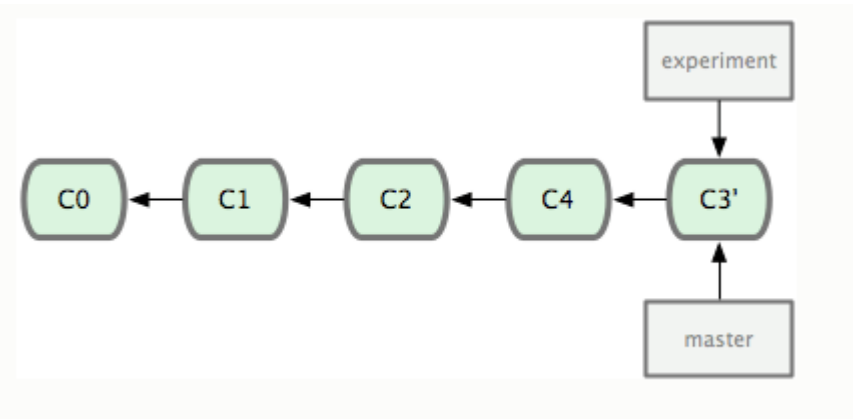

nerge, ca

Merge




rebase



# A better git pull

- Cleaner history when pulling:
  - `git pull –rebase`
- (this replaces the merge with the rebase command)

# Thanks!