Week 10
Version Control

3 December 2018

CS 35L Lab 4

Jeremy Rotman

Announcements

- ➔ Assignment #9 is due Friday by **11:55pm**
  - ◆ **LATE ASSIGNMENTS WILL NOT BE ACCEPTED**
- ➔ Remember to submit both your slides and report to CCLE for assignment #10
  - ◆ Both of these should be PDF files
  - ◆ I will post the submission link tonight
- ➔ Assignment #2 grades are uploaded
  - ◆ I may be changing them a little bit, in a positive way
  - ◆ If your grade was significantly lower than you expected
    - Feel free to email me and I can look into what was wrong

Outline

- ➔ Version Control
- ➔ Git
- ➔ Assignment 8

Questions?

Software Development Process

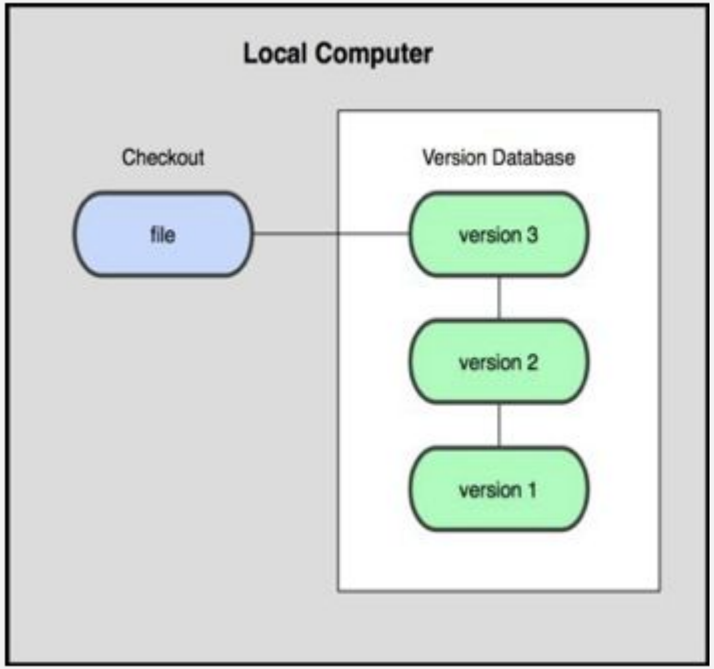
- ➔ Involves many changes to code
  - ◆ New features
  - ◆ Bug fixes
  - ◆ Etc.
- ➔ Many people are working on the same/different parts of code
- ➔ Many versions of software are released

Source/Version Control

- Track changes to code and other files related to the software
- Track entire history of the software
- Various Version Control Software (VCS) to do this
    - Git
    - Subversion
    - Perforce

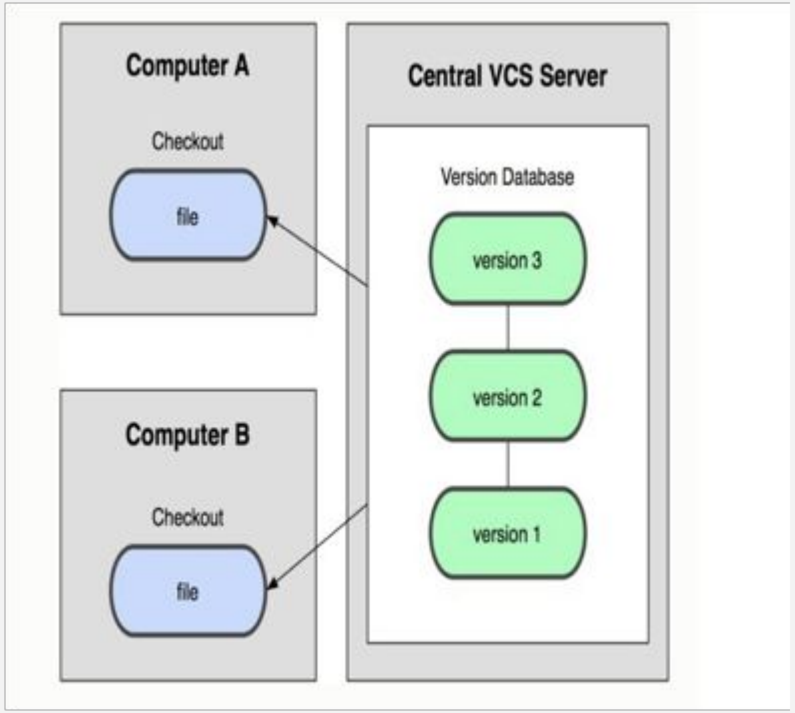
Local VCS

- ➔ Different versions of the software are in different folders in the local machine
- ➔ No server
- ➔ Other users should copy via disk/network



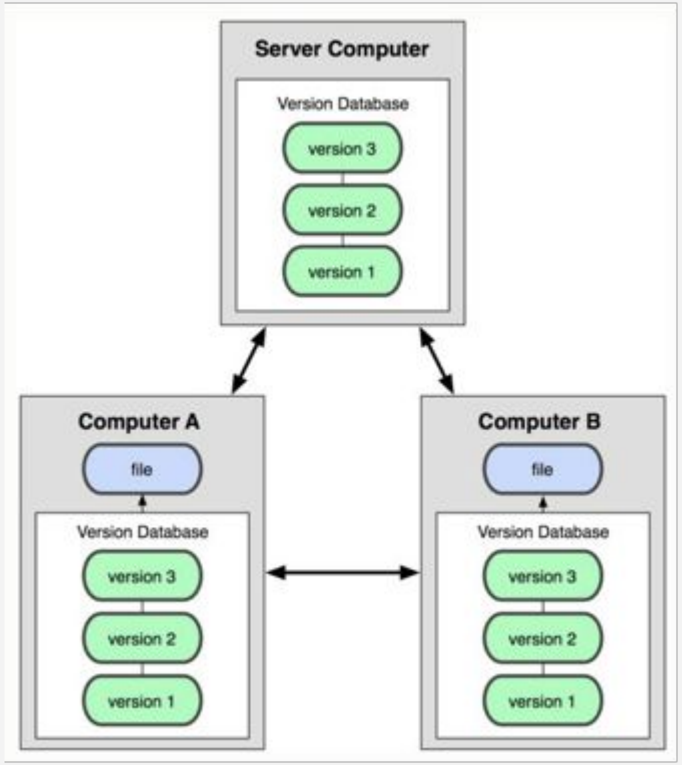
Centralized VCS

- ➔ Version history on a central server
- ➔ Users get working copy of files
- ➔ Changes must be committed to the server
- ➔ All users can get changes



Distributed VCS

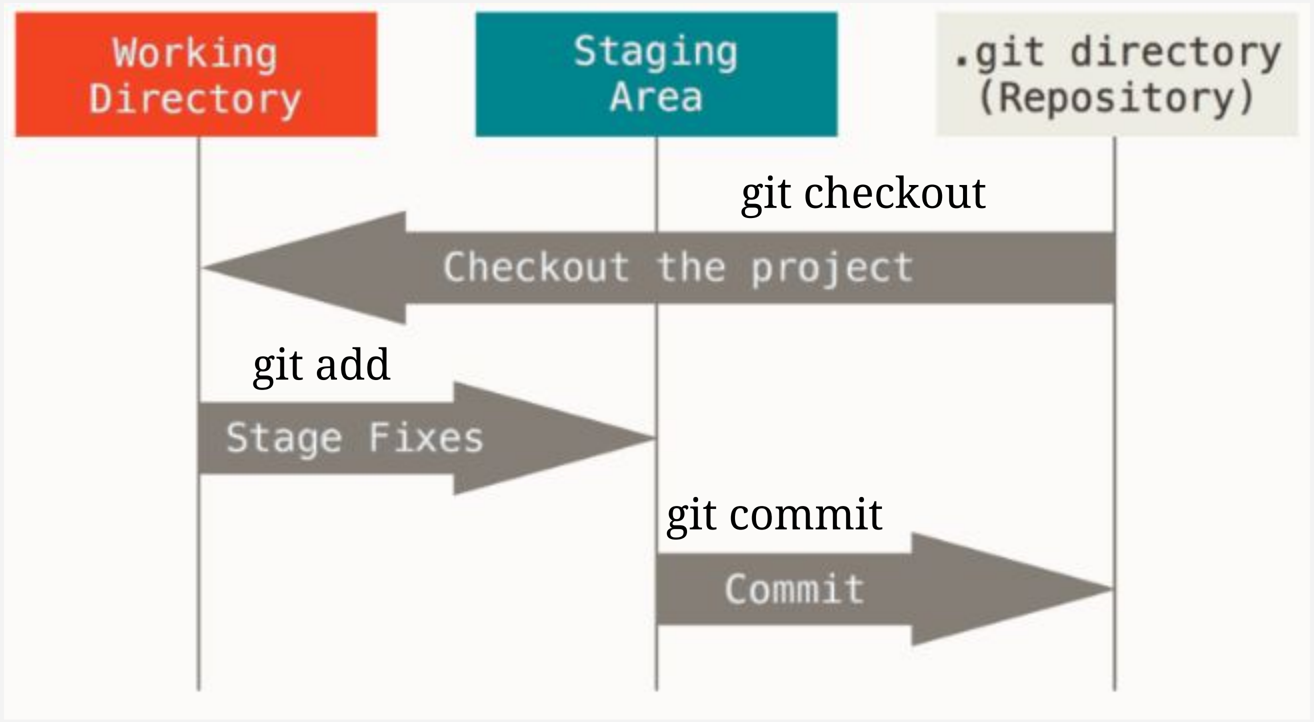
- ➔ All users mirror the entire version history
- ➔ Users have version control at all times
- ➔ Changes can be communicated between users
- ➔ Git is distributed



Useful Terms

- ➔ Repository (repo)
  - ◆ Files and folders related to the software code
  - ◆ Full history of the software
- ➔ Working copy
  - ◆ Copy of the software’s files in the repository
- ➔ Check-out
  - ◆ To create a working copy of the repository
- ➔ Check-in / Commit
  - ◆ Write the changes made in the working copy to the repository
  - ◆ Commits are recorded by the VCS

Git States

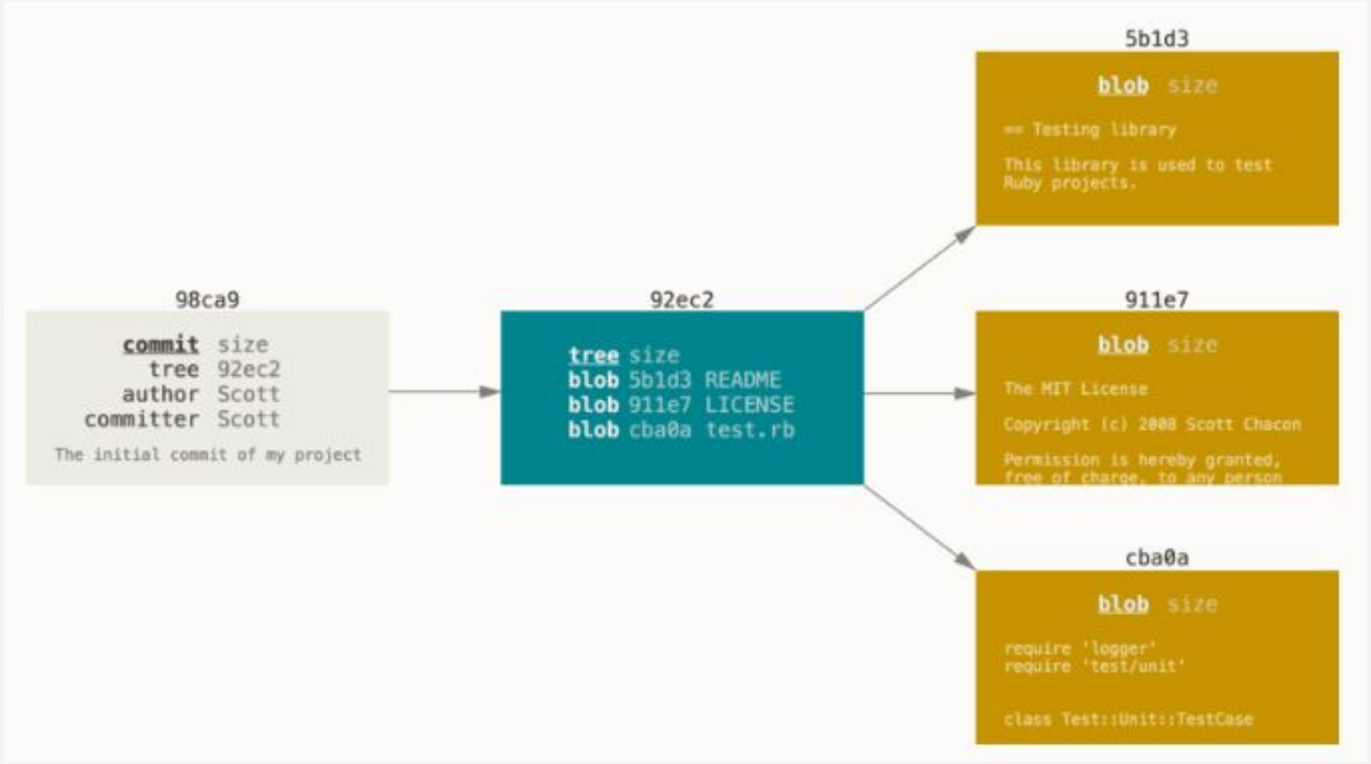

Working Directory
Staging Area
.git directory (Repository)
git checkout
Checkout the project
git add
Stage Fixes
git commit
Commit

Git Repository Objects

- ➔ Blobs
  - ◆ Binary object used to store the contents of each file
- ➔ Trees
  - ◆ The hierarchy between files stored in a repository
- ➔ Commit
  - ◆ Refers to a particular “git commit”
  - ◆ A snapshot of the git tree and blobs in a repository
- ➔ Tags
  - ◆ Essentially attached names for specific commits

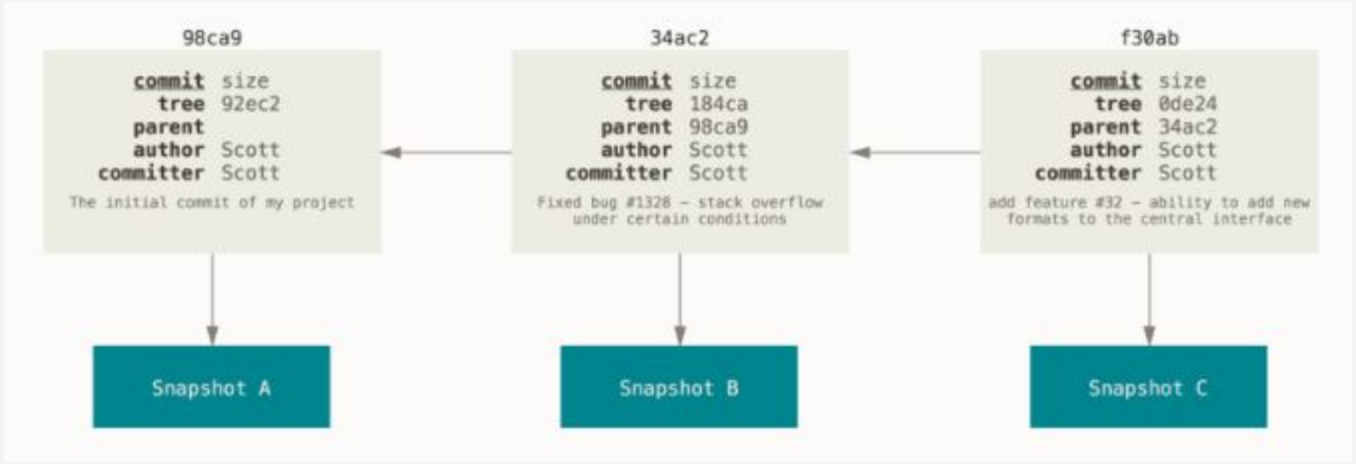
Git Repository Objects

A git commit is a snapshot of the repository, a version



Git Repository Objects

A git commit is a snapshot of the repository, a version

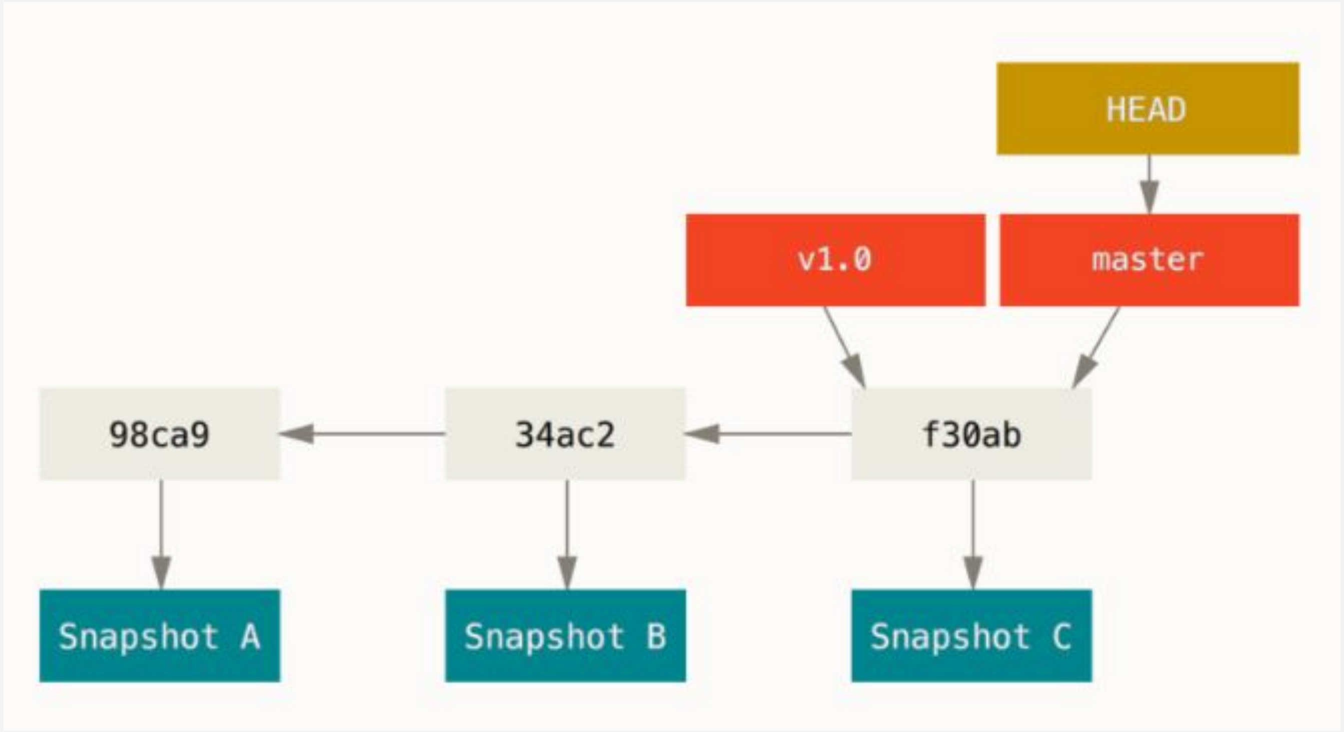


More Terms!

- Branch
  - An active line of development
- Head
  - Named reference to a commit at the tip of a branch
- HEAD
  - Your currently active branch
- Detached HEAD
  - Occurs when you check-out a commit that is not necessarily part of any branch
- Master
  - Default branch

Branching Example

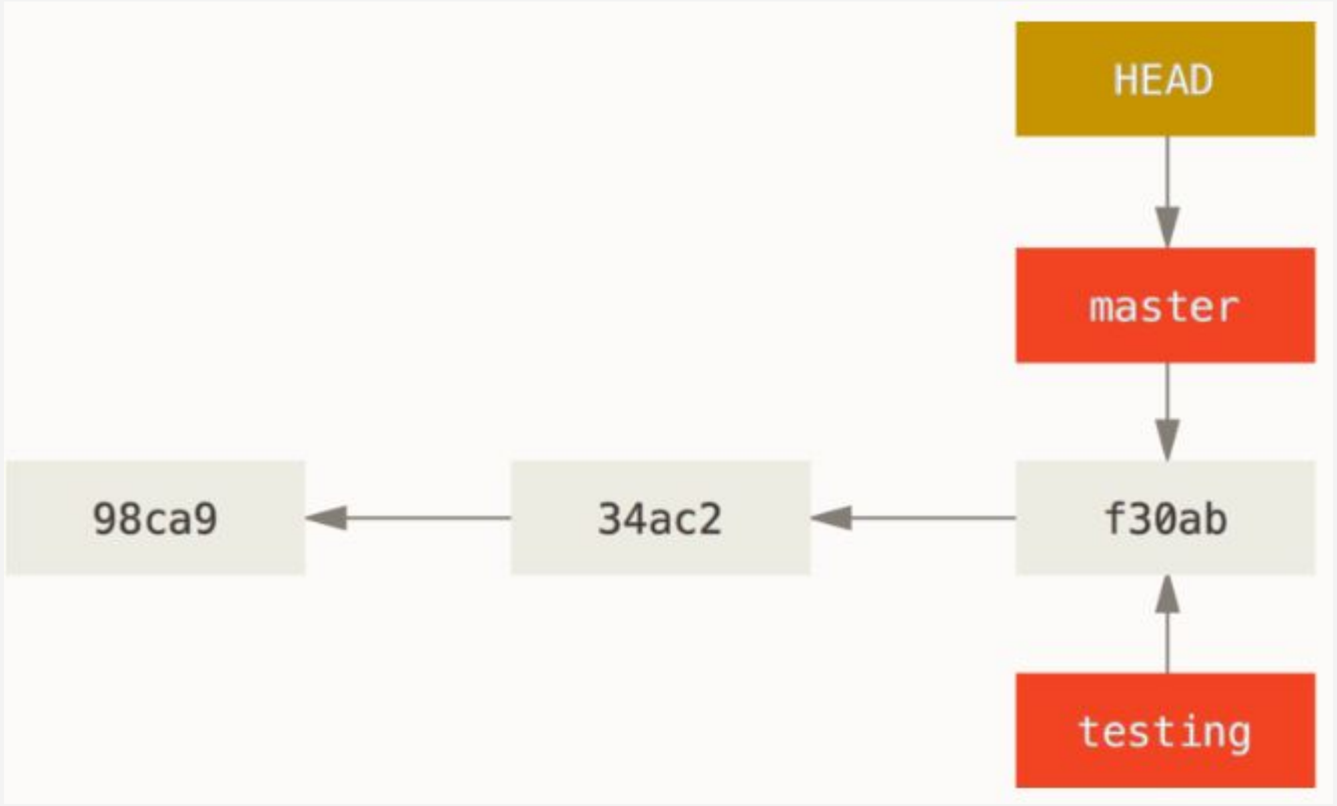
Going back to our previous tree, we now see that we are on the default master branch



Branching Example

Now assume that we have made a new branch called testing

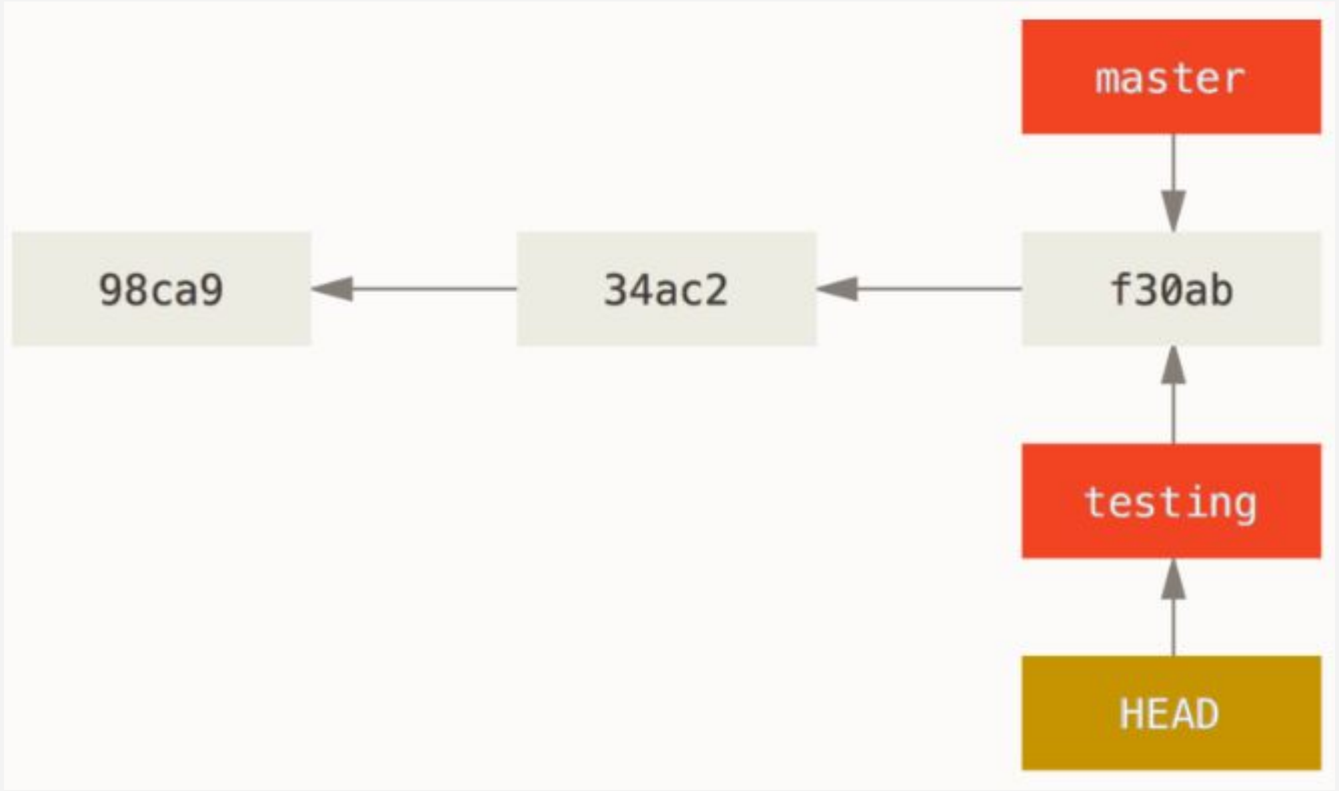
```
git branch
testing
```



Branching Example

We can now switch to that new branch

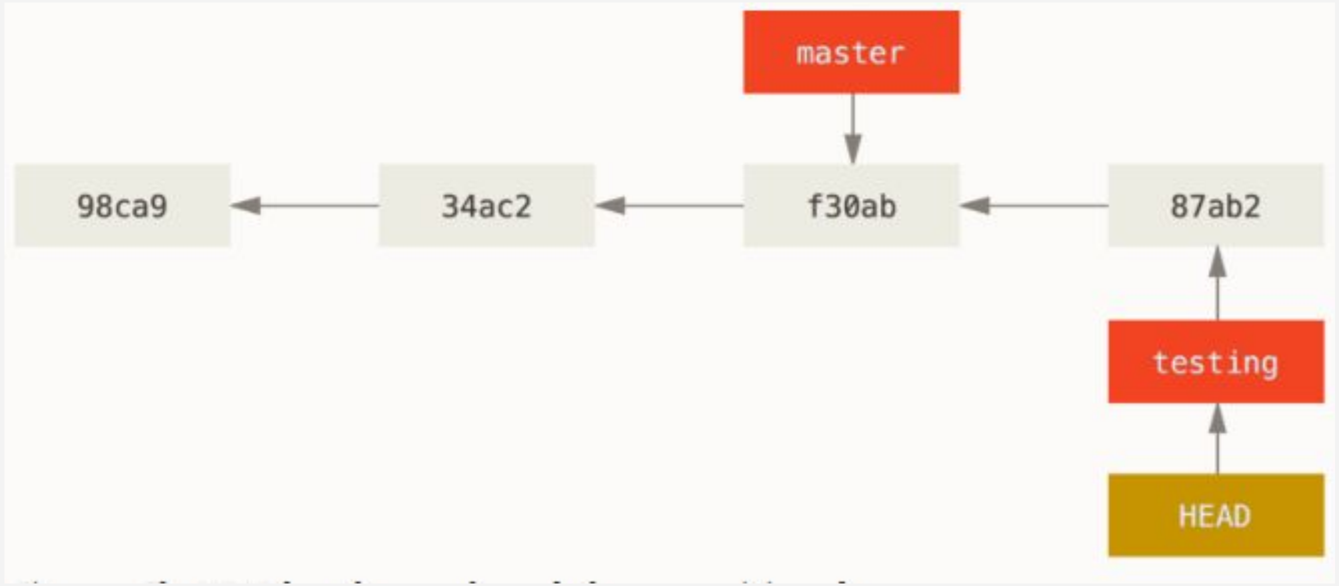
```
git checkout testing
```



Branching Example

Now if we make a new commit, we progress the testing branch

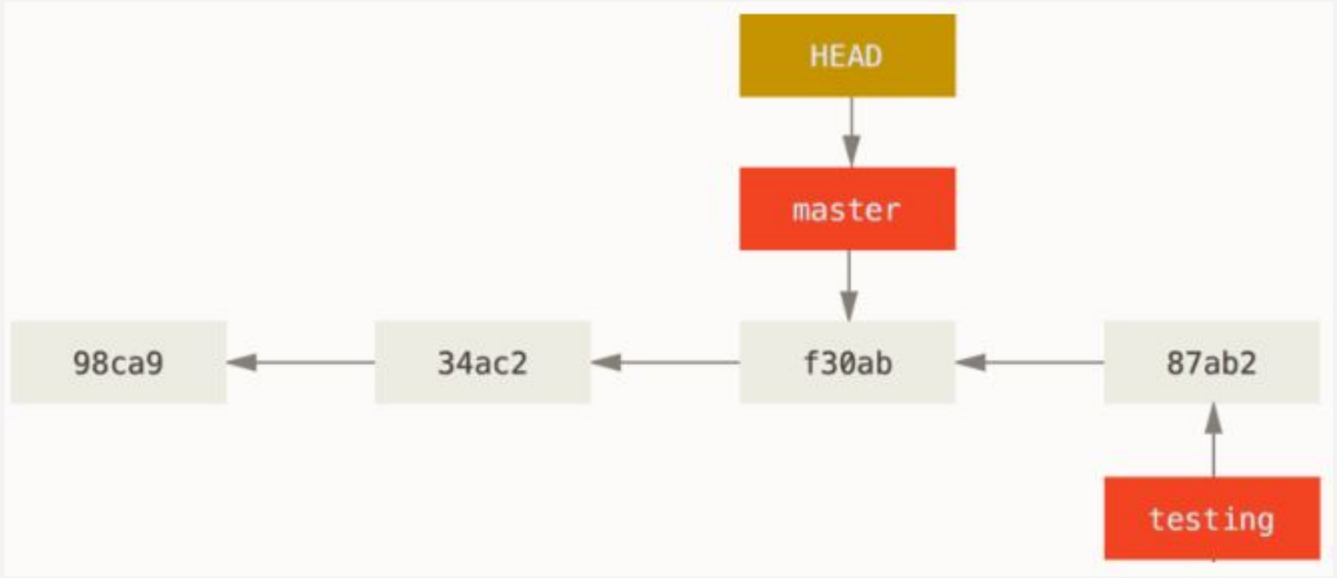
```
git commit
-a -m ‘made
a change’
```



Branching Example

Then we can go back to the master branch

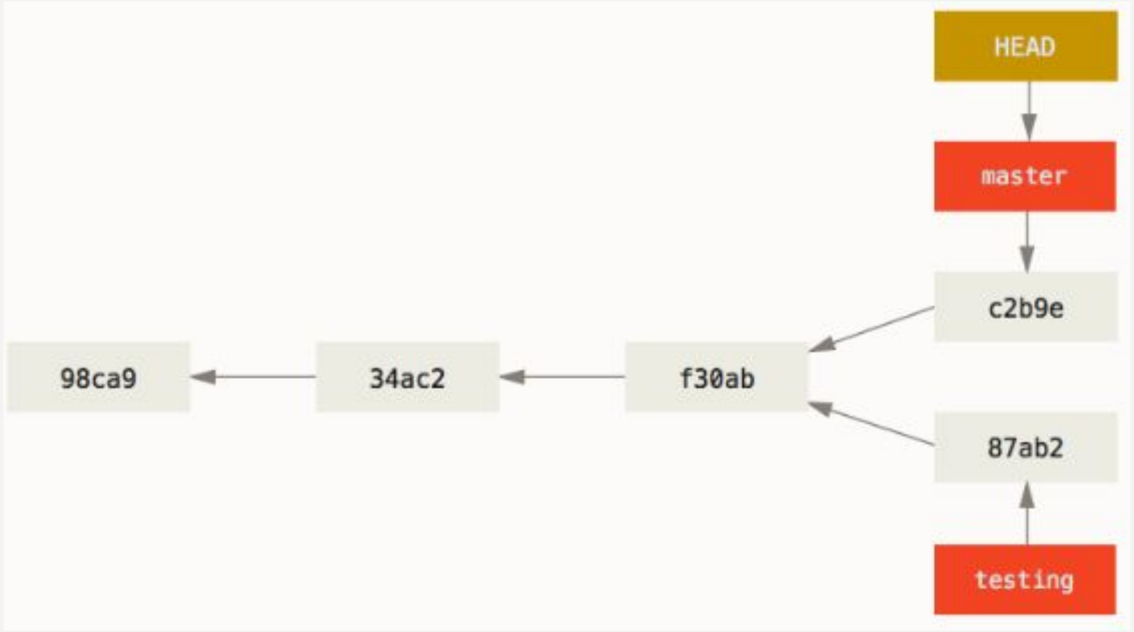
`git checkout master`



Branching Example

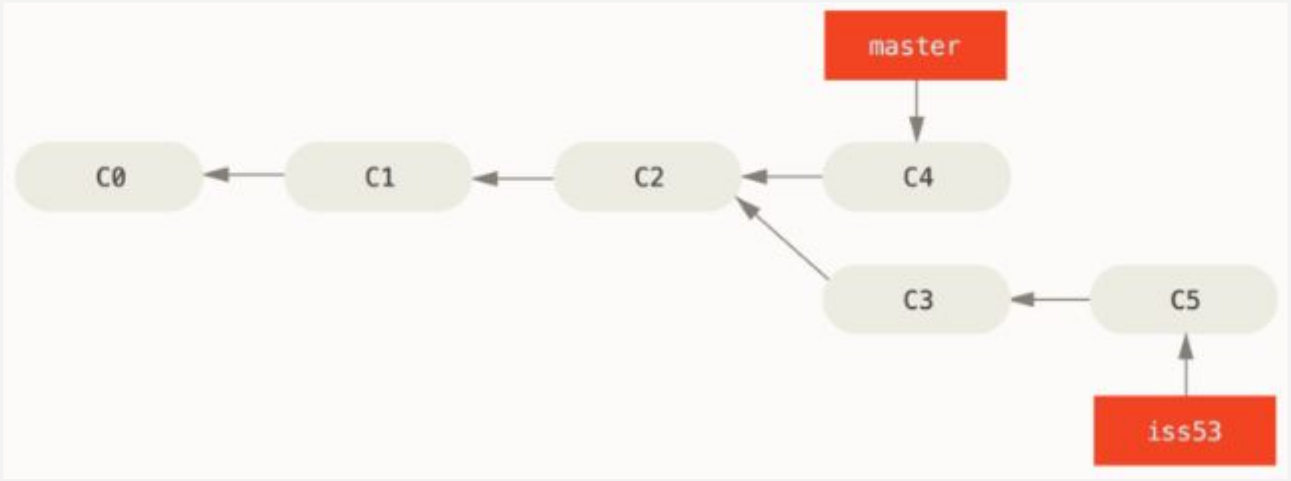
Make a new change, and the branches diverge

```
git commit
-a -m ‘other
changes’
```



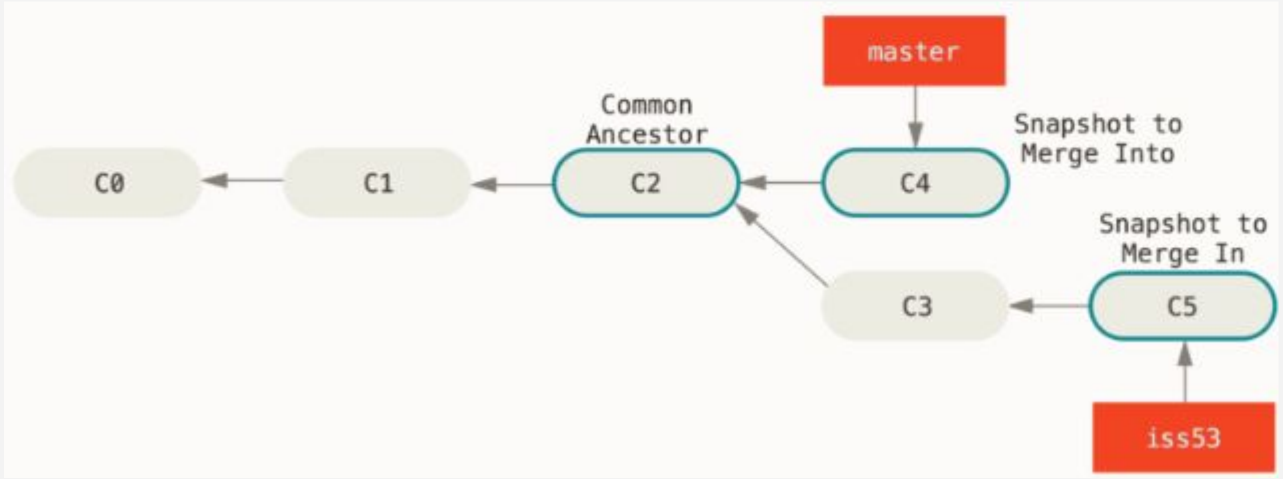
Merging

- If you branch from the master branch
  - You may need to merge back with it
  - This can be messy if master has changed since you branched



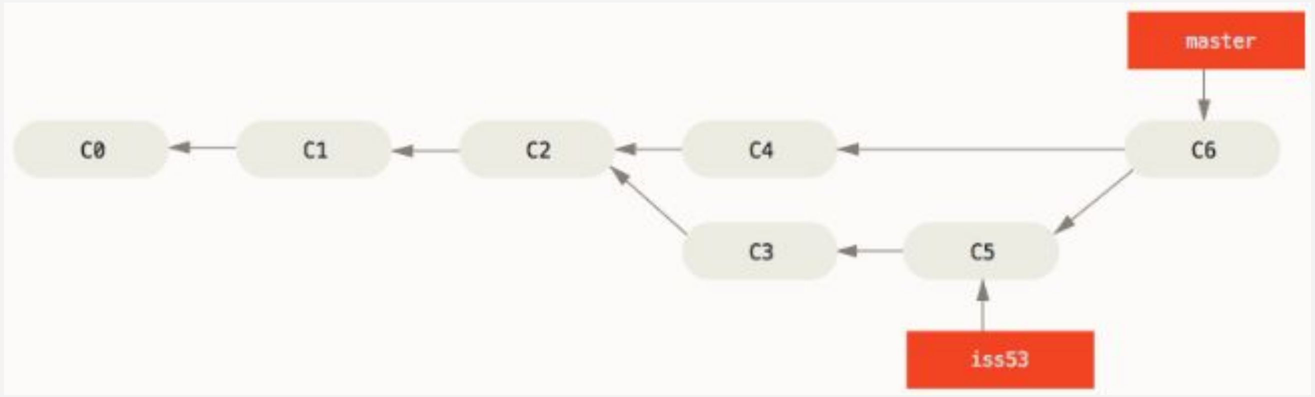
Merging

- To merge the snapshot from iss53:
  - `git checkout master`
  - `git merge iss53`



Merging

- If the merge is successful, then there will be a new snapshot that will be from a 3-way merge between master, iss53, and their common ancestor



Brief Git Introduction: First Repository

- `mkdir gitroot`
- `cd gitroot`
- `git init`
  - Creates empty git repo, with default branch “master”
- `echo “Hello World” > hello.txt`
- `git add .`
  - Stage changes to the index
- `git commit -m ‘Check in number one’`

Brief Git Introduction: Working with Git

- ➔ `echo “I love Git” >> hello.txt`
- ➔ `git status`
  - ◆ Shows list of modified files
- ➔ `git diff`
  - ◆ Shows changes we made compared to the index
- ➔ `git add hello.txt`
- ➔ `git diff`
  - ◆ After staging, no changes will be shown
- ➔ `git diff HEAD`
  - ◆ This will show differences by in the working version
- ➔ `git commit -m “Second Commit`

Git Commands

- Repository Creation
  - git init
    - Start a new repository
  - git clone
    - Create a copy of an existing repository
- Branching
  - git checkout <tag/commit> -b <new-branch-name>
    - Create a new branch at the commit you are checking out

Git Commands

- Commits
  - git add
    - Stage modified/new files
  - git commit -m “<commit_message>”
    - Check-in the changes to the repository
    - -a option combines add and commit
- Getting Help
  - git help

Git Commands

- Getting Info
  - git status
    - Shows modified files between index and current HEAD
    - Shows modified files between working tree and index
    - Shows new files
  - git diff
    - Compare working copy with staged files
  - git log
    - Show history of commits
  - git show <object>
    - Show a certain object in the repository

Assignment 9

- In the lab you will be applying a patch to an old version of diffutils
  - You will get this old version by checking out version 3
- In the homework you will be creating a new branch from version 3 that utilizes the patch that you wrote
  - You will once again need a partner
  - Partners will test each others patches
- The homework asks you to use gitk
  - This visualizes the git tree
  - In order to use it on the linux server you will have to enable X forwarding

Questions?