CS 35L

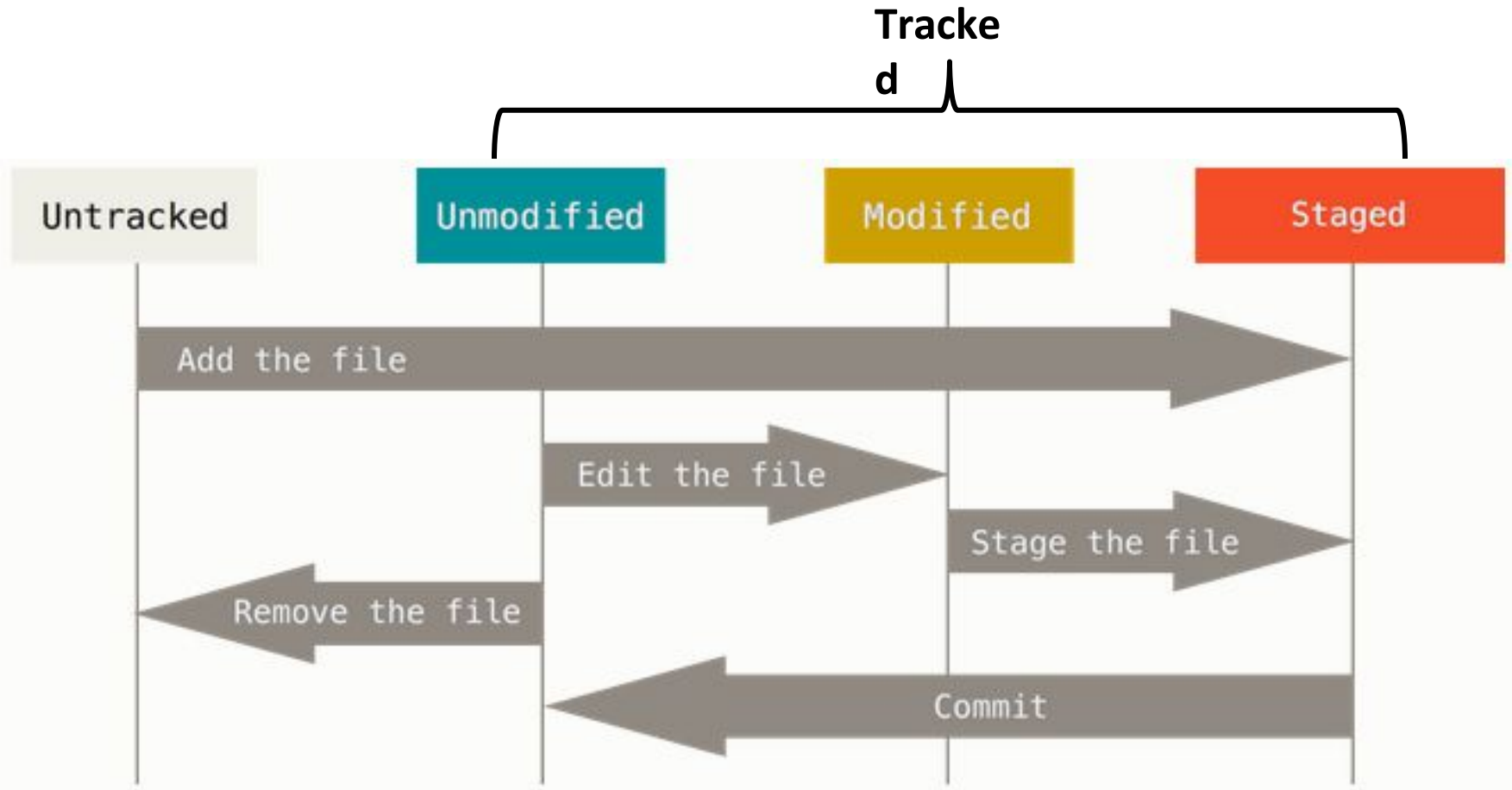
Week 4

TA: Tomer Weiss
Jan-28-2016

goo.gl/oLTghu

Slides

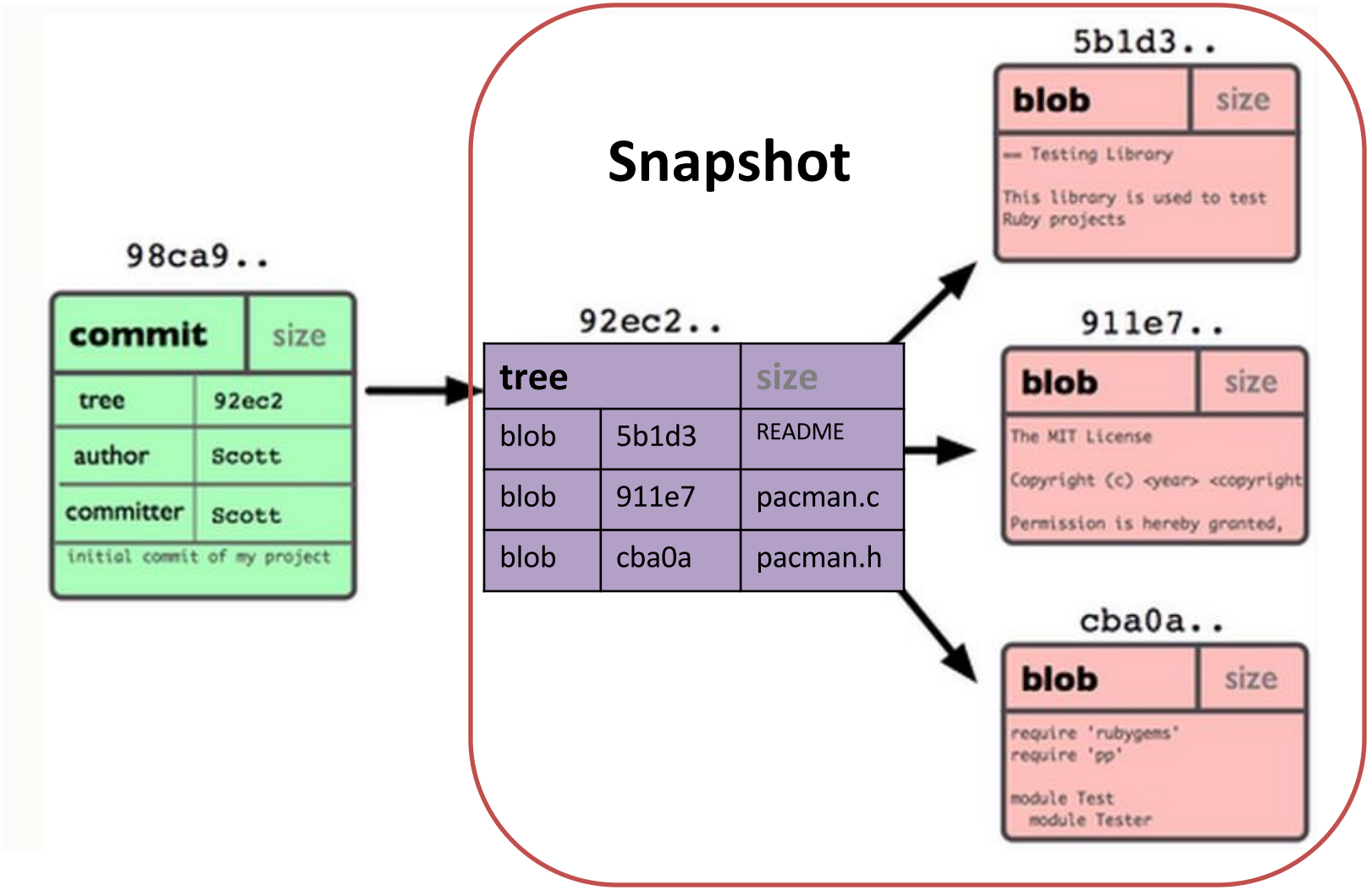
Git File Status Lifecycle



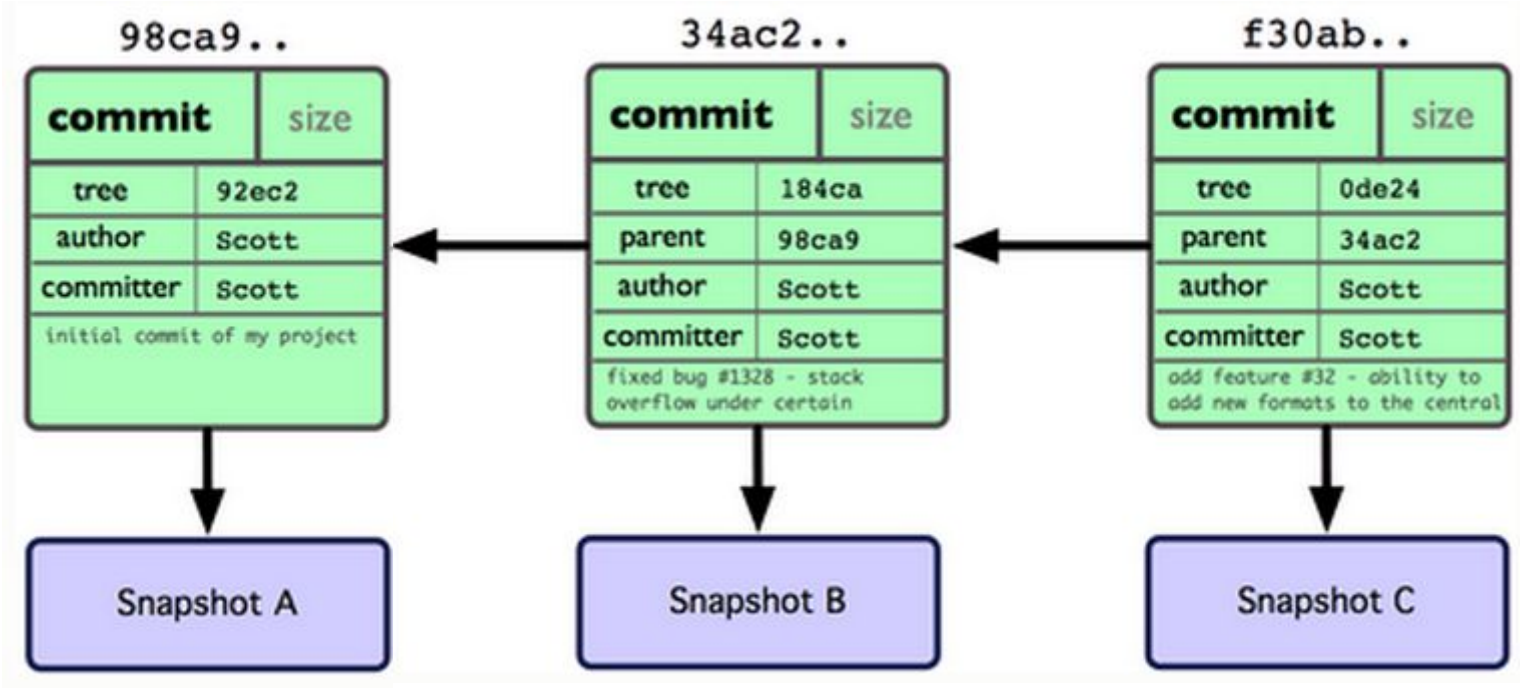
Git Example

- Project
  - games: pacman.c, pacman.h, README
- Create repository to track new project
  - `$ git init` (creates .git dir w/ all necessary repo files)
- Is the project tracked?
  - No, need to add files and do an initial commit
    - `$ git add` pacman.c pacman.h README
    - `$ git commit -m` "initial commit of my project"

Git Repo Structure



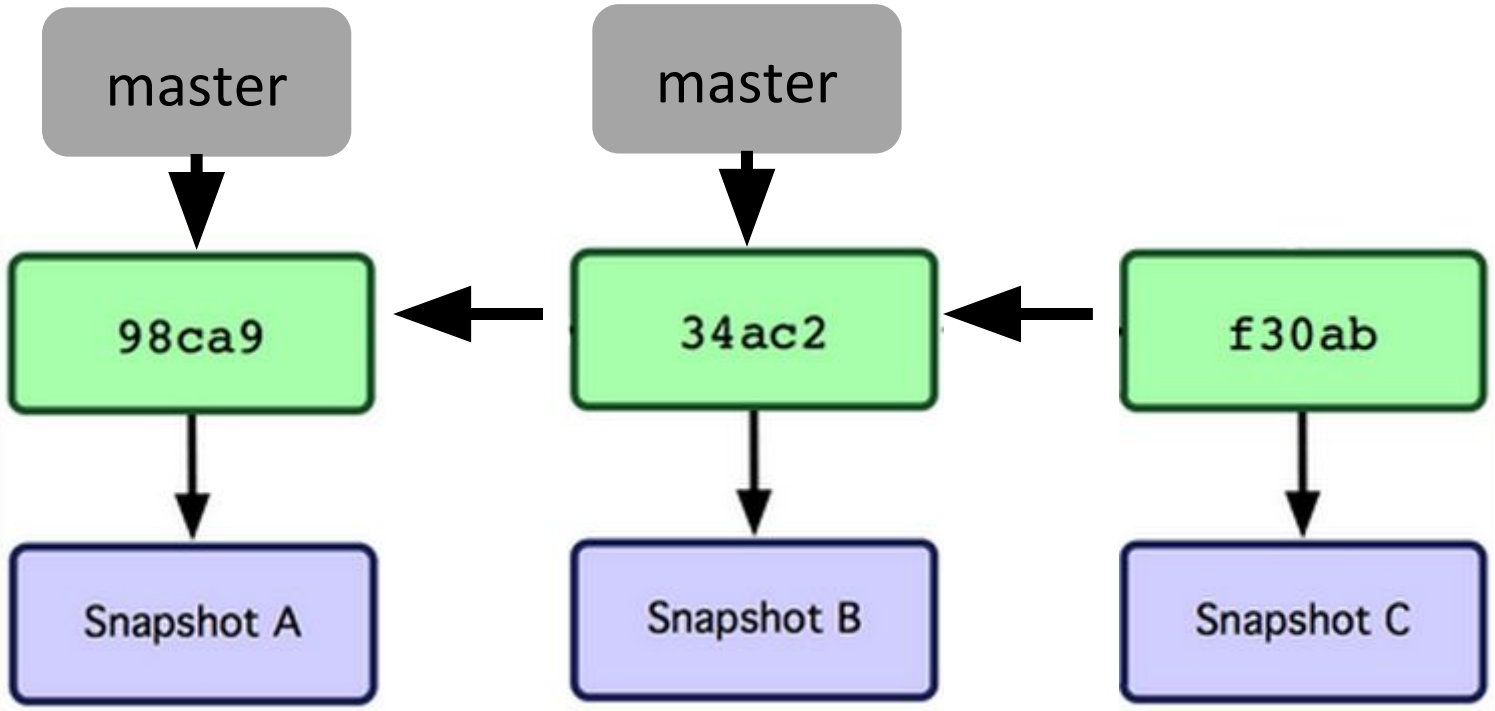
After 2 More Commits...



What Is a Branch?

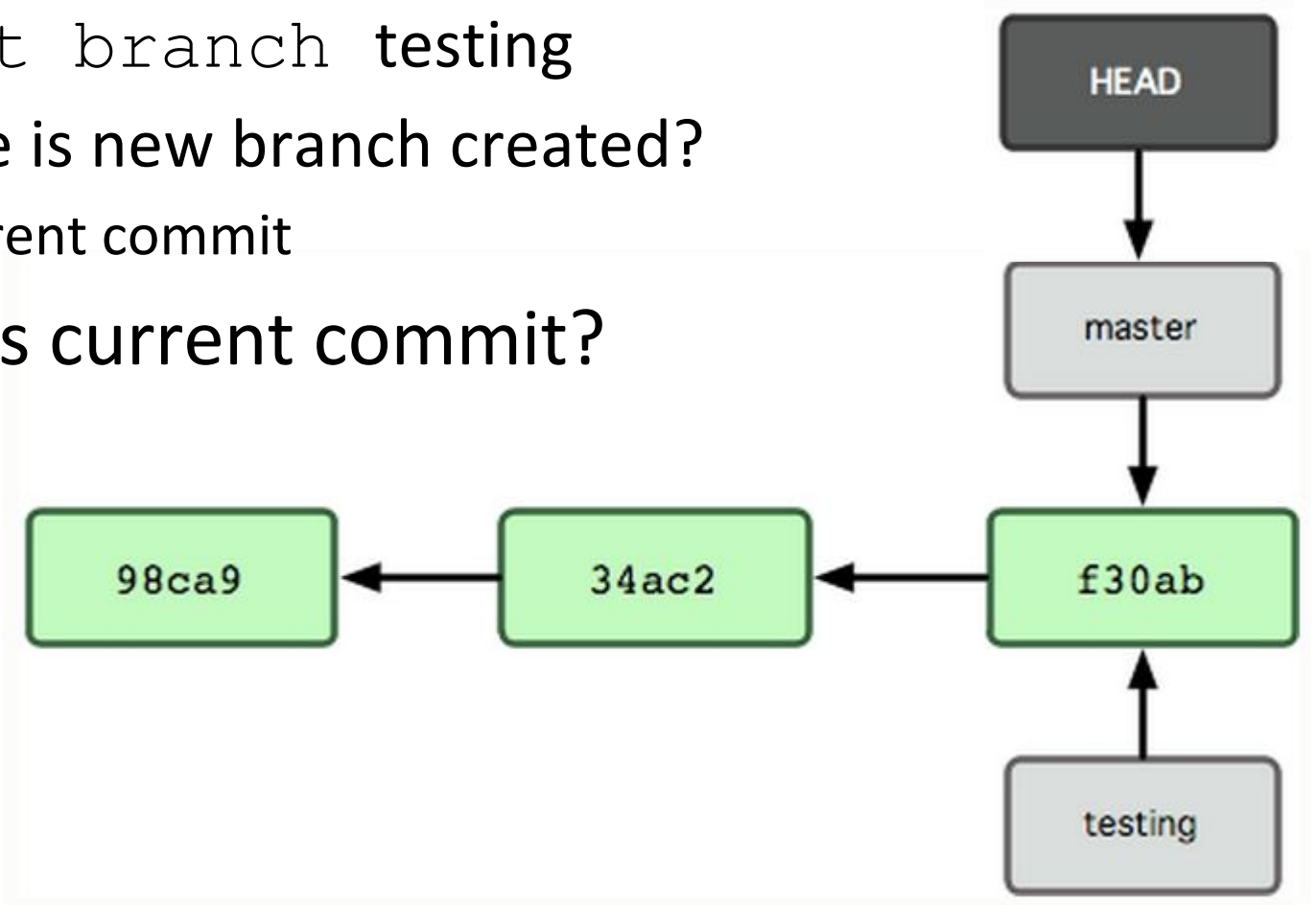
- A pointer to one of the commits in the repo (head) + all ancestor commits
- When you first create a repo, are there any branches?
  - Default branch named ‘master’
- The default master branch
  - points to last commit made
  - moves forward automatically, every time you commit

Where Is Master?



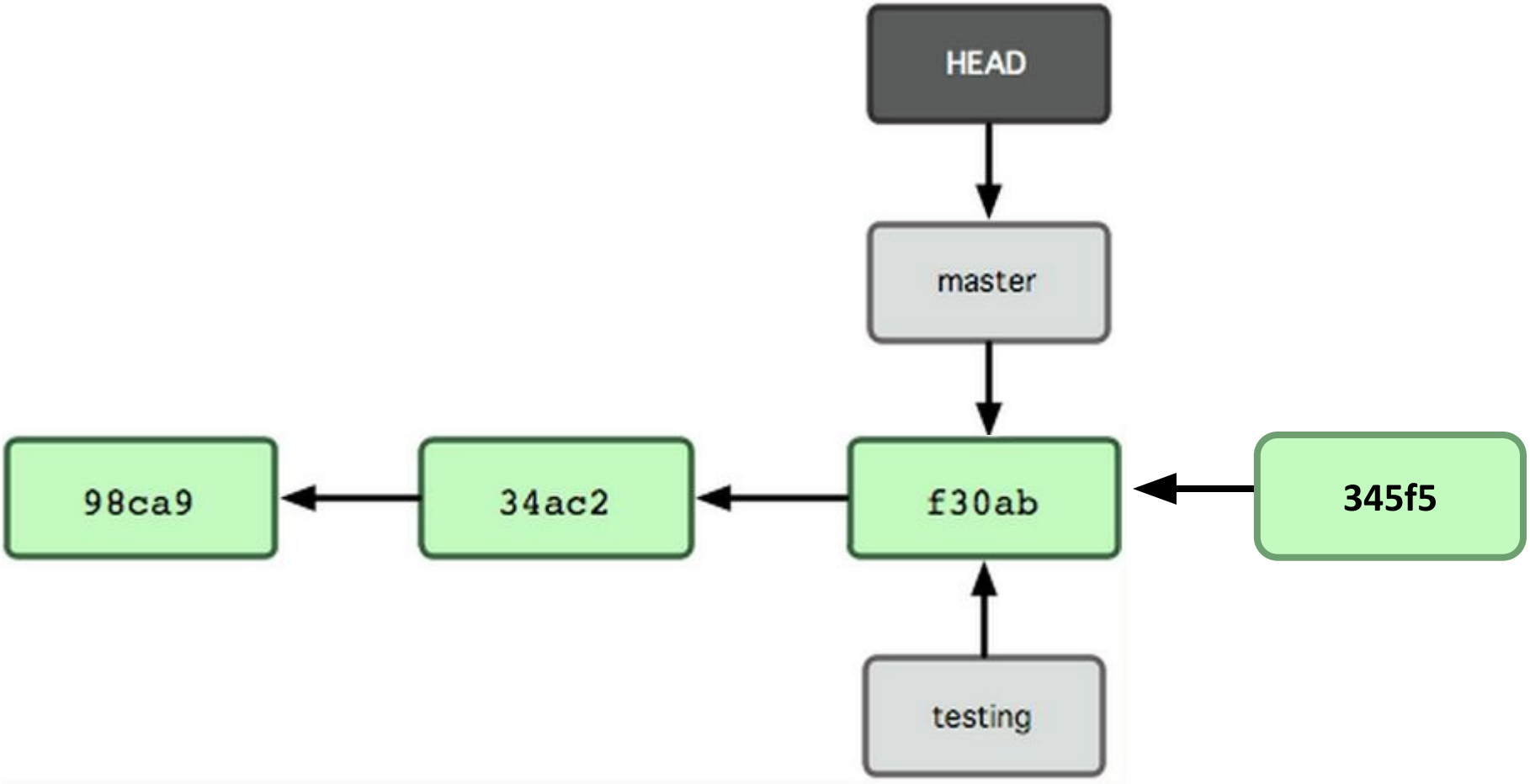
New Branch

- Creating a new branch = creating new pointer
  - `$ git branch` testing
  - Where is new branch created?
    - Current commit
- Where is current commit?
  - HEAD



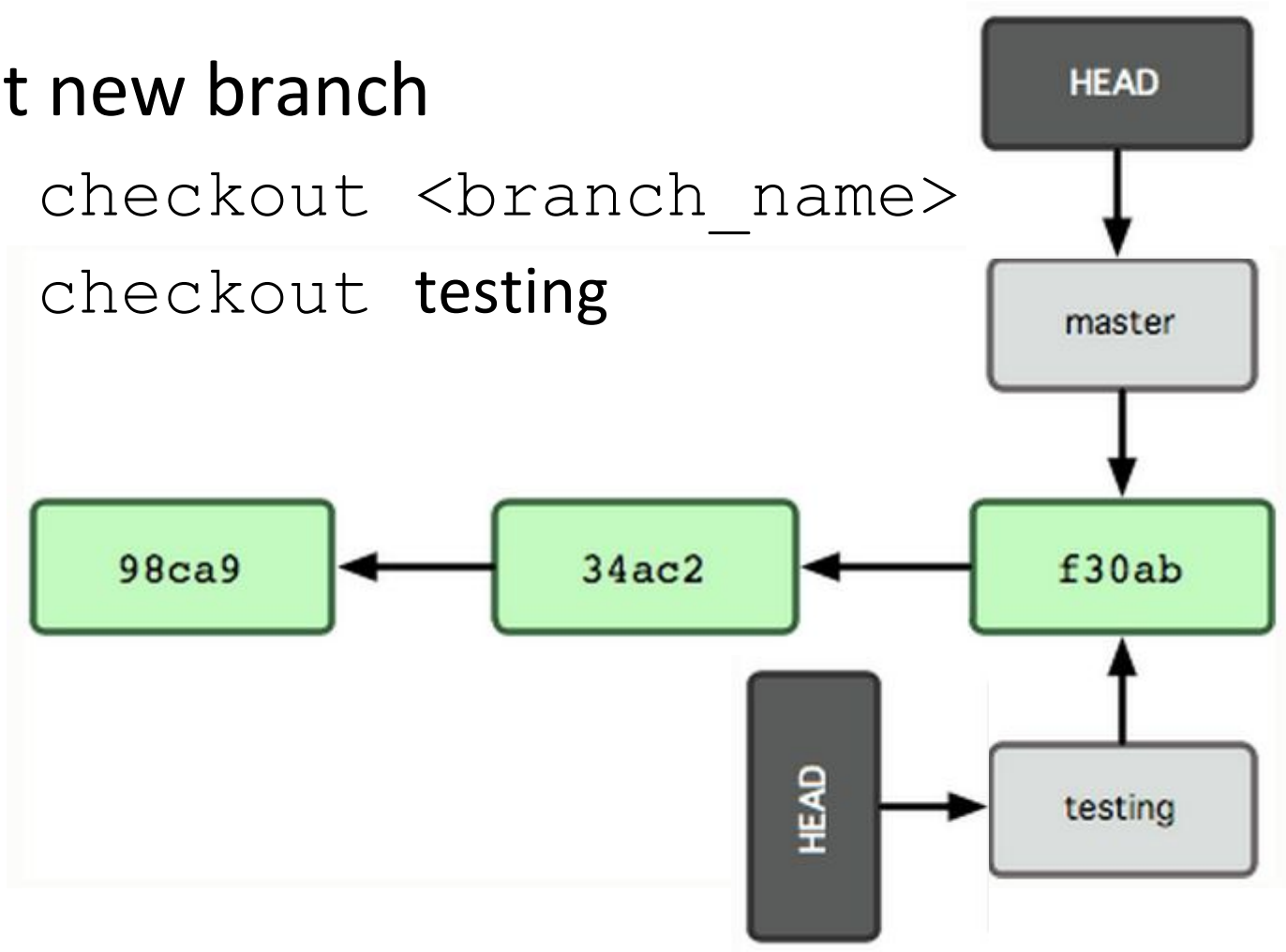
New Commit

- What happens if we make another commit?

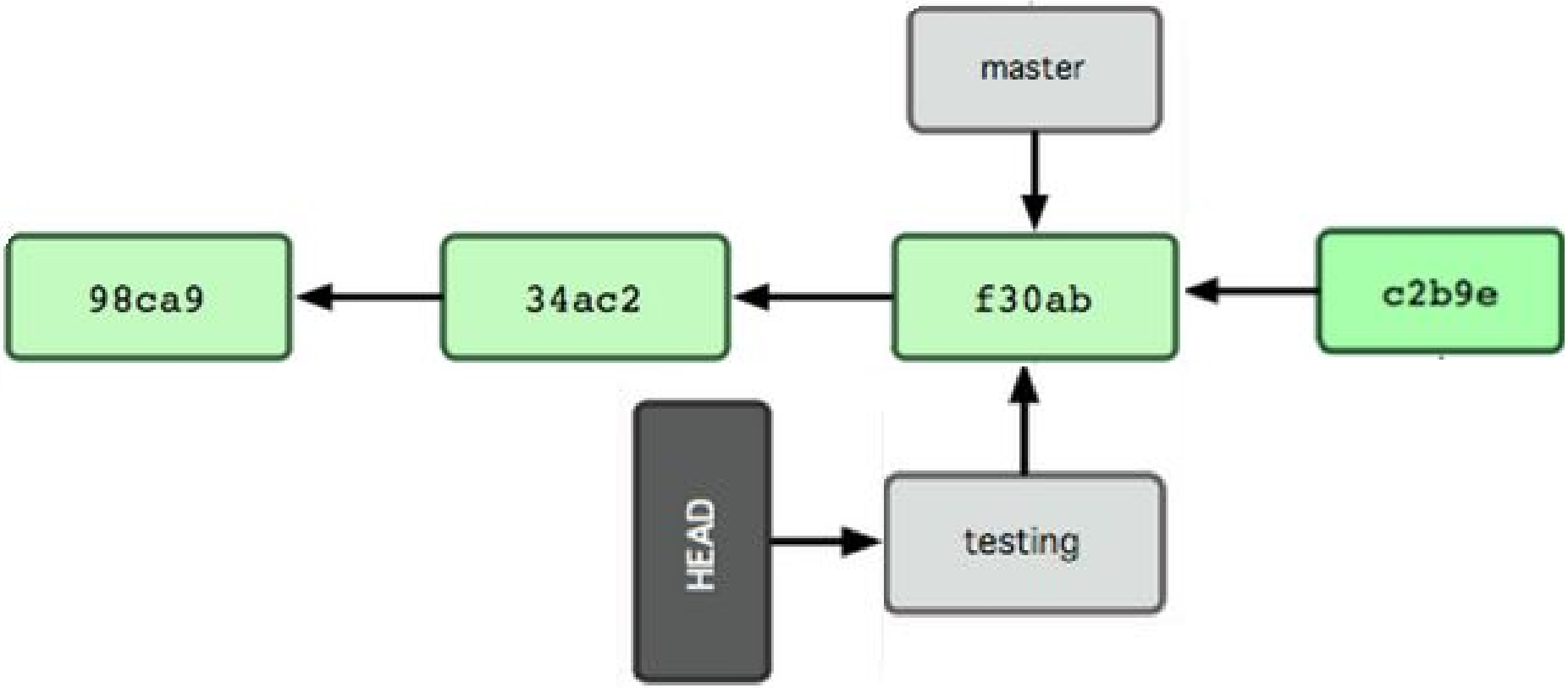


Switching to New Branch

- Check out new branch
  - `$ git checkout <branch_name>`
  - `$ git checkout` **testing**



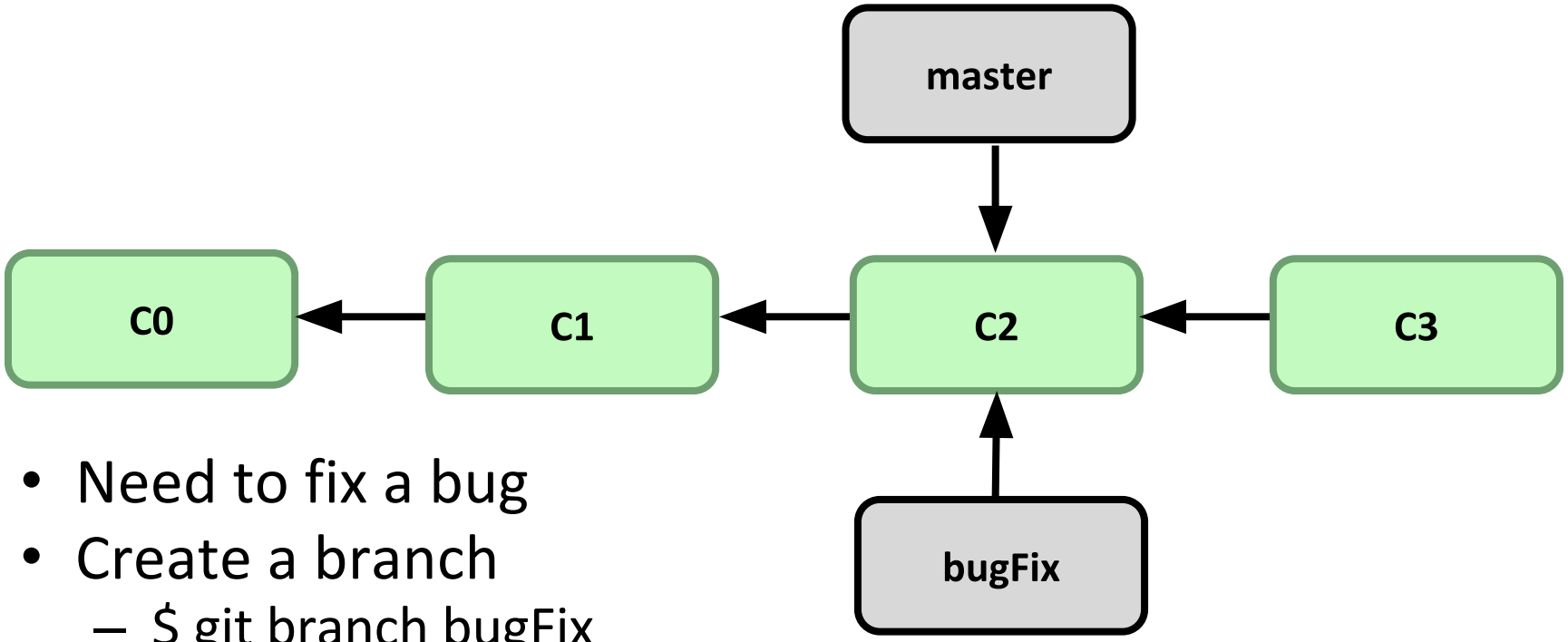
Commit After Switch



Why Branching?

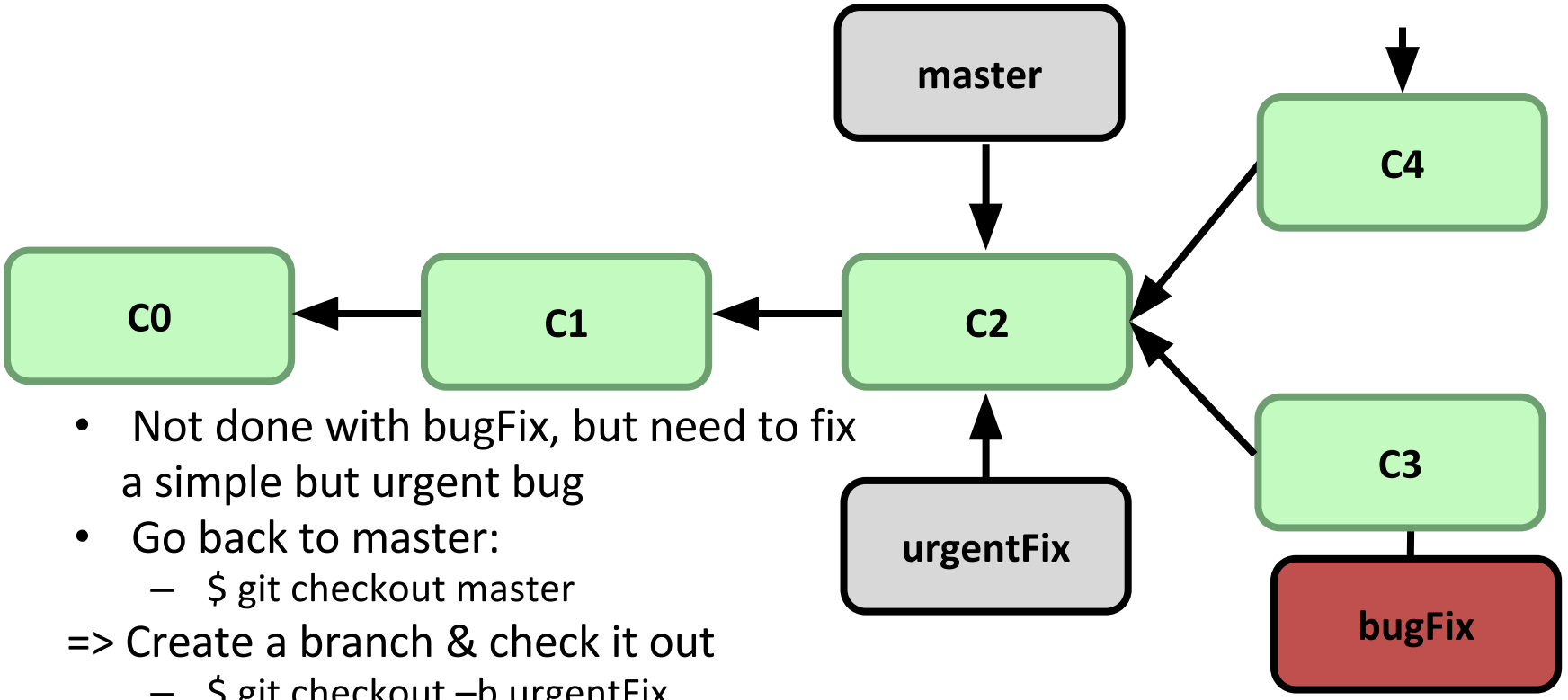
- Experiment with code without affecting main branch
- Separate projects that once had a common code base
- 2 versions of the project

Merging I



- Need to fix a bug
- Create a branch
  - $ git branch bugFix
  - $ git checkout bugFix
- Make some progress
  - Make a commit

Merging II



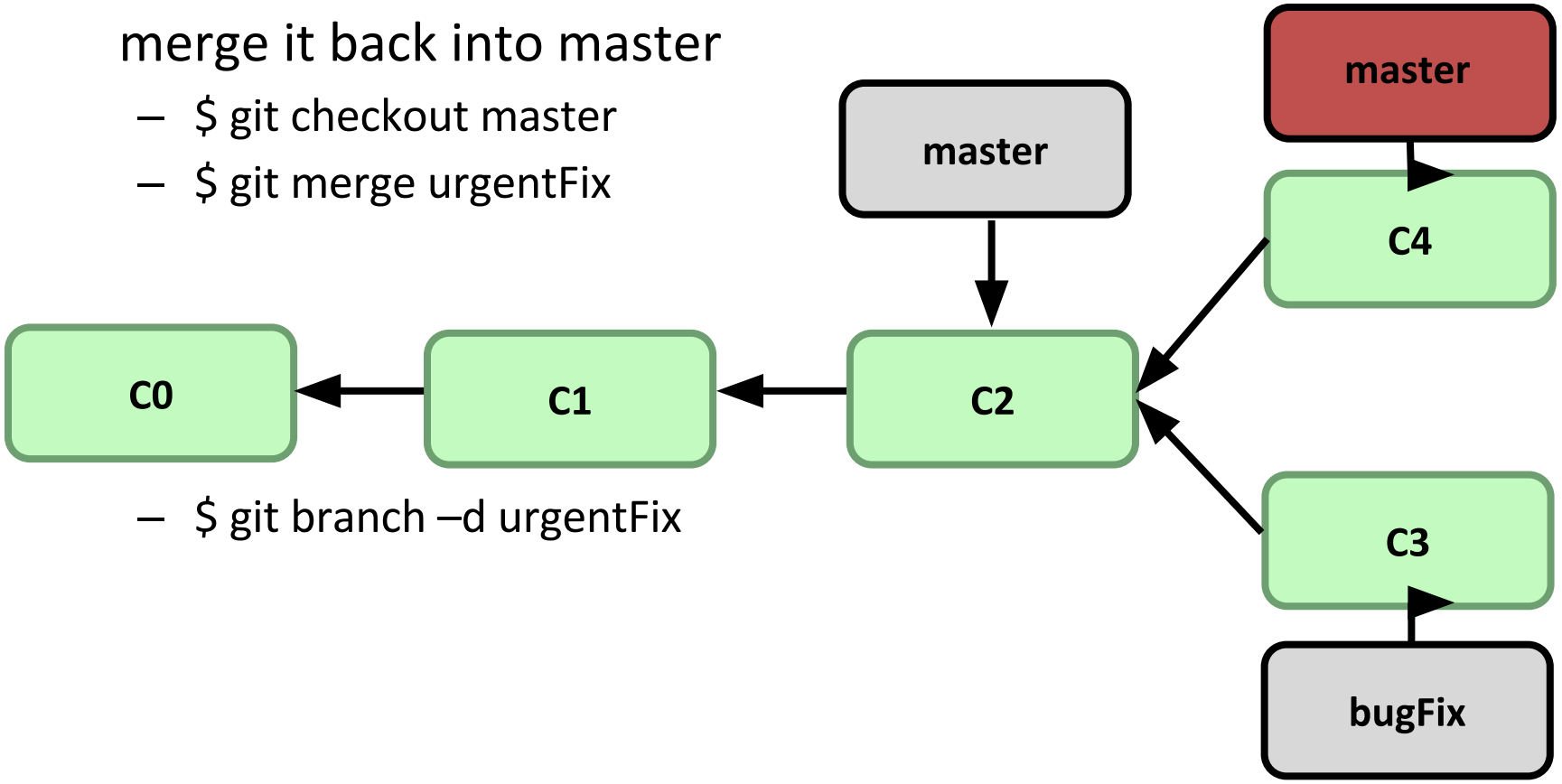
- Not done with bugFix, but need to fix a simple but urgent bug
- Go back to master:
  - $ git checkout master

=> Create a branch & check it out

  - $ git checkout –b urgentFix
- Make some progress
  - Make a commit

Merging III

- When confident about fix, we can merge it back into master
  - $ git checkout master
  - $ git merge urgentFix



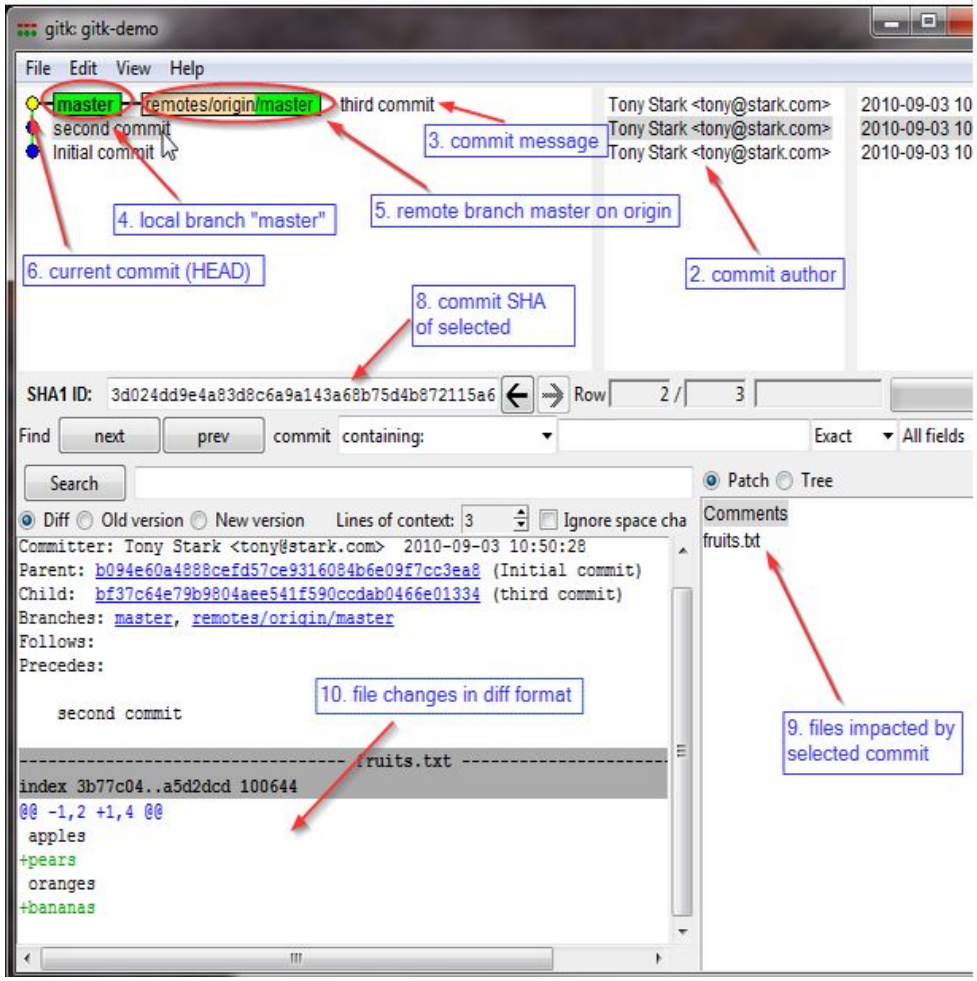
  - $ git branch –d urgentFix

Homework 4

- Publish patch you made in lab 4
  - Create a new branch “quote” off of version 3.0
    - Branch command + checkout command **(git branch quote v3.0; git checkout quote)**
    - `$ git checkout v3.0 -b quote`
  - Use patch from lab 4 to modify this branch
    - Patch command
    - `$ patch –pnum <` quote-3.0-patch.txt
  - Modify ChangeLog-2008 file in diffutils directory
    - Add entry for your changes similar to entries in ChangeLog
  - Commit changes to the new branch
    - `$ git add .  $ git commit –F <Changelog file>`
  - Generate a patch that other people can use to get your changes
    - `$ git format-patch -[num] --stdout > formatted-patch.txt`
  - Test your partner’s patch
    - Check out version 3.0 into a tmp branch
    - Apply patch with `git am` command: $ `git am <` formatted-patch.txt
    - Build and test with $ `make check`
    - Make sure partner’s name is in HW4.txt for #8

Gitk

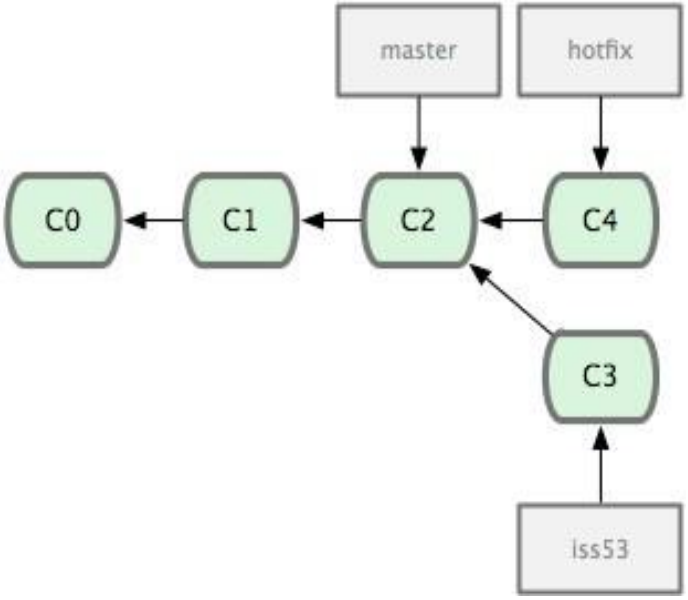
- A repository browser
  - Visualizes commit graphs
  - Used to understand the structure of the repo
  - Tutorial: http://lostechies.com/joshuaflanagan/2010/09/03/use-gitk-to-understand-git/



Gitk

- SSH into the server with X11 enabled
  - ssh -X for OS with terminal (OS X, Linux)
  - Select “X11” option if using putty (Windows)
- Run gitk in the ~eggert/src/gnu/emacs directory
  - Need to first update your PATH
    - $ export PATH=/usr/local/cs/bin:$PATH
  - Run X locally before running gitk
    - Xming on Windows

Merging



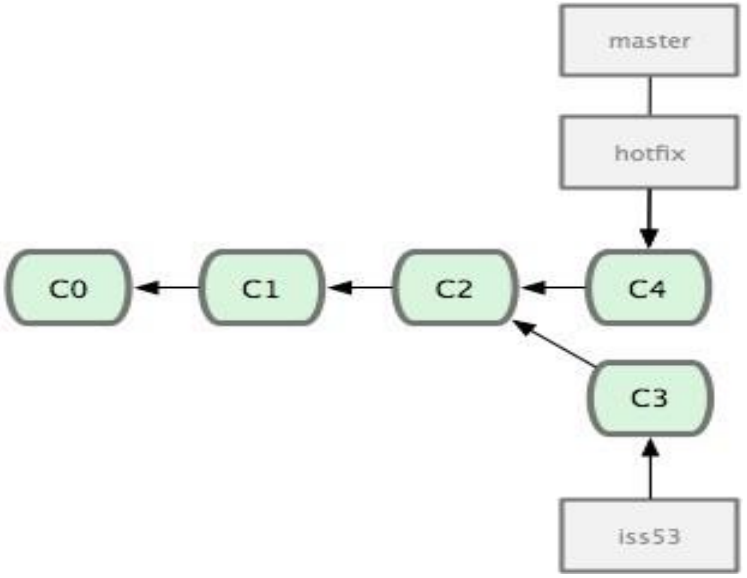


Image Source: git-scm.com

- Merging hotfix branch into master
  - git checkout master
  - git merge hotfix
- Git tries to merge automatically
  - Simple if it is a forward merge
  - Otherwise, you have to manually resolve conflicts

Merging

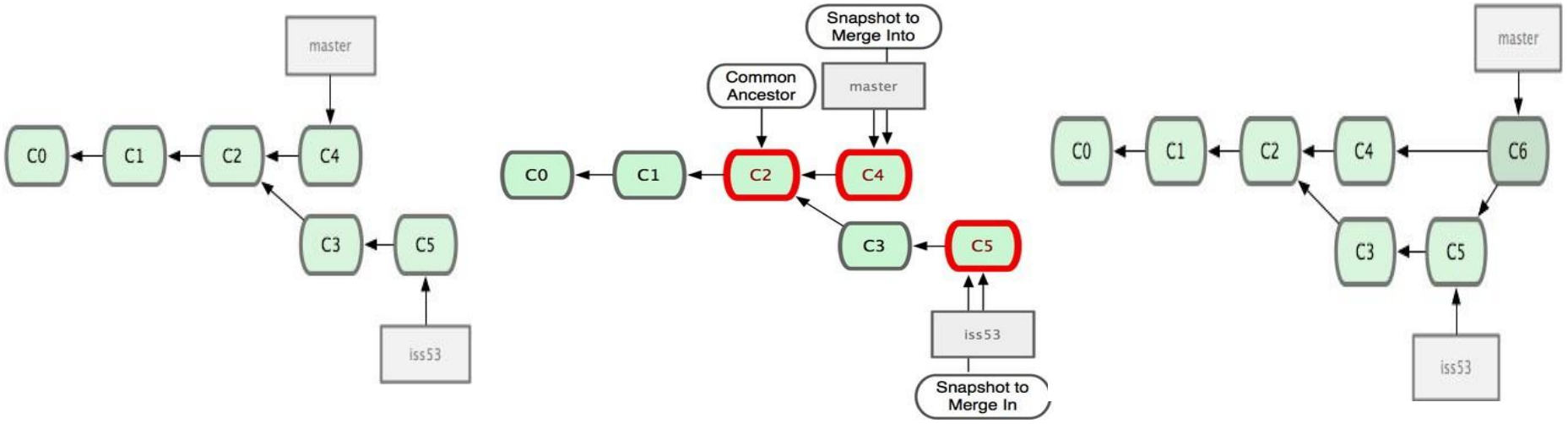


Image Source: git-scm.com

- Merge iss53 into master
- Git tries to merge automatically by looking at the changes since the common ancestor commit
- Manually merge using 3-way merge or 2-way merge
  - Merge conflicts - Same part of the file was changed differently

Merging

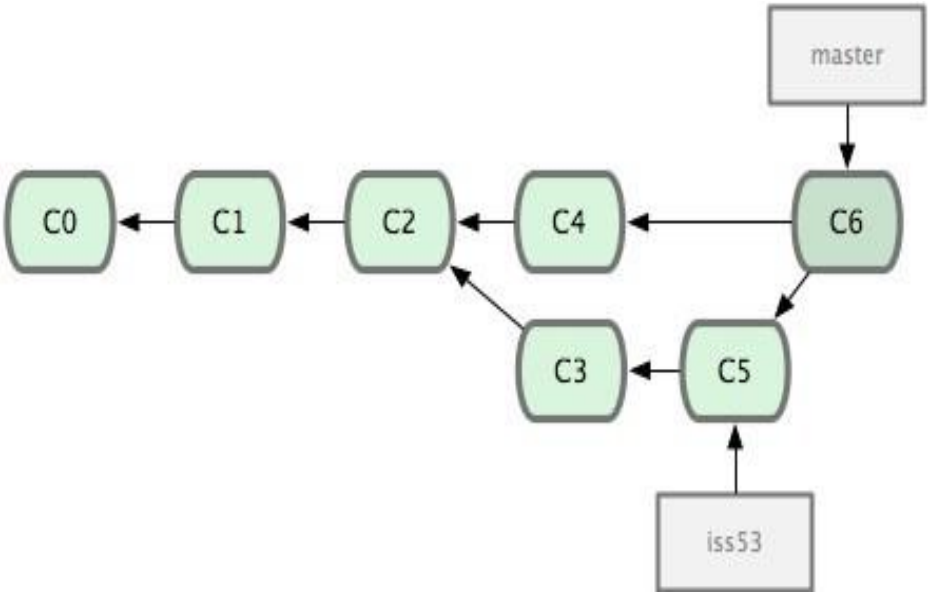


Image Source: git-scm.com

- Refer to multiple parents
  - git show hash
  - git show hash^2 (shows second parent)
- HEAD^^ == HEAD~2

Lab

Still confused about git? rogerdudler.github.io/git-guide

web.cs.ucla.edu/classes/winter16/cs35L/assign/assign4.html