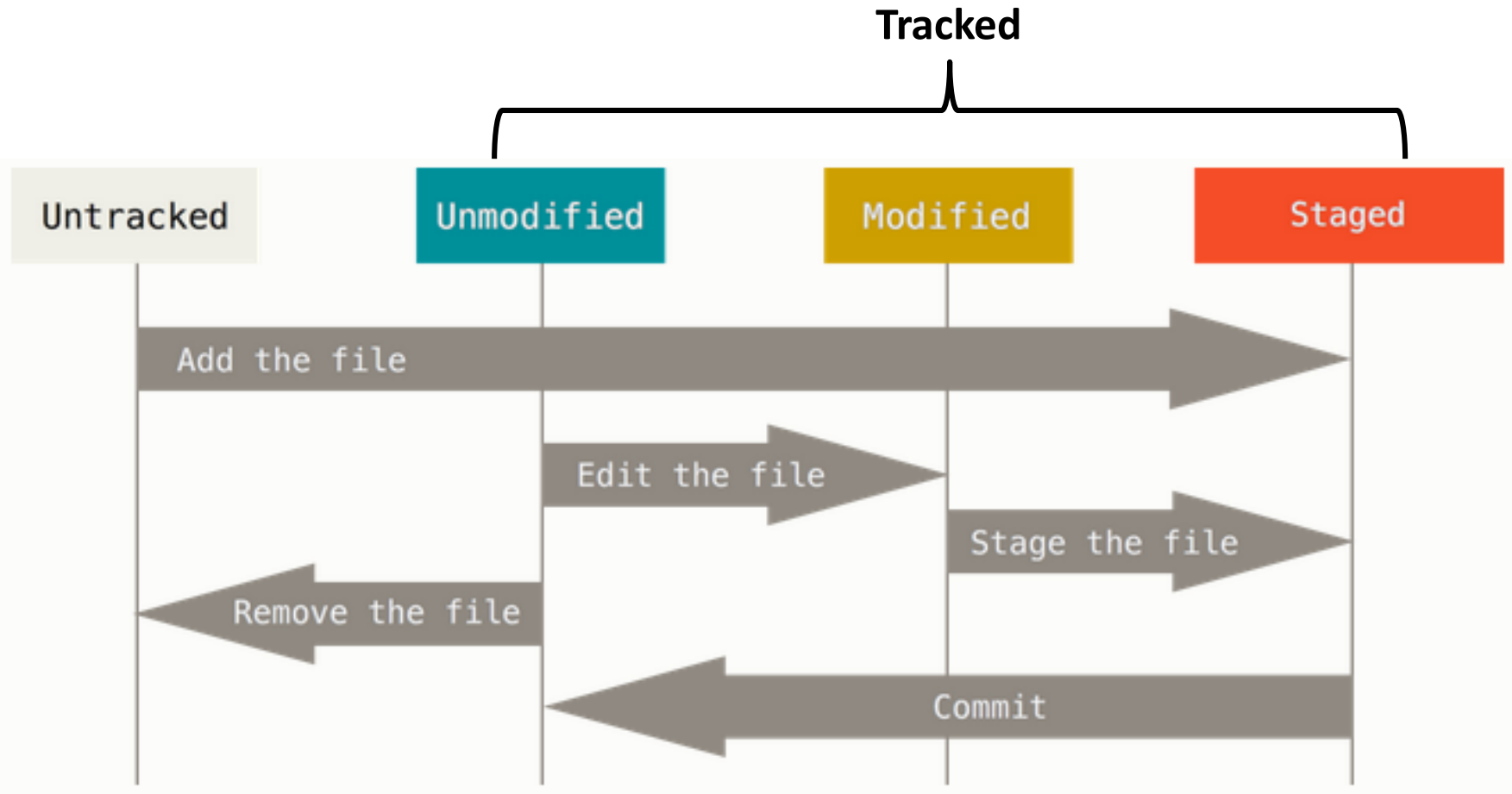
Verifying and Publishing a Backported Change

Homework 4

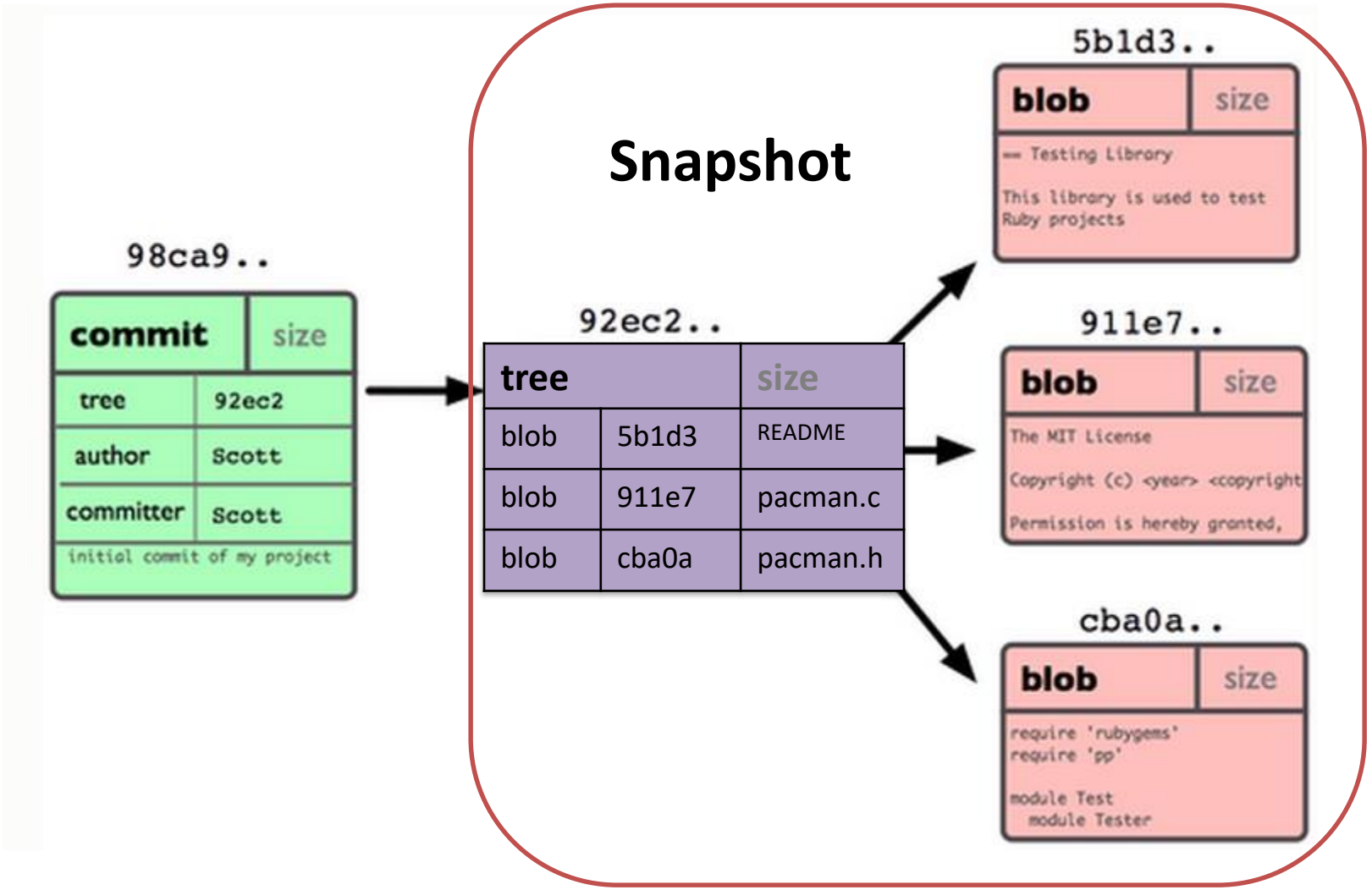
Git File Status Lifecycle



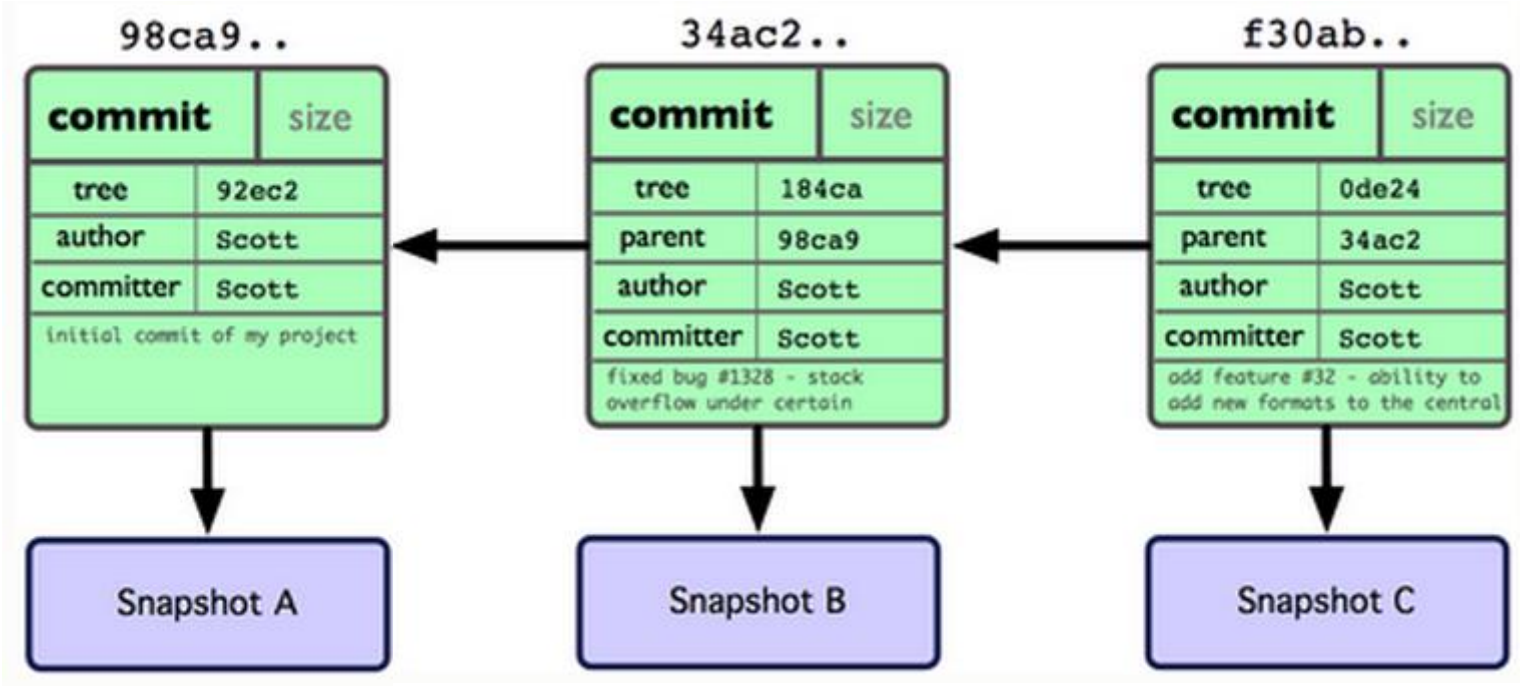
Git Example

- 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



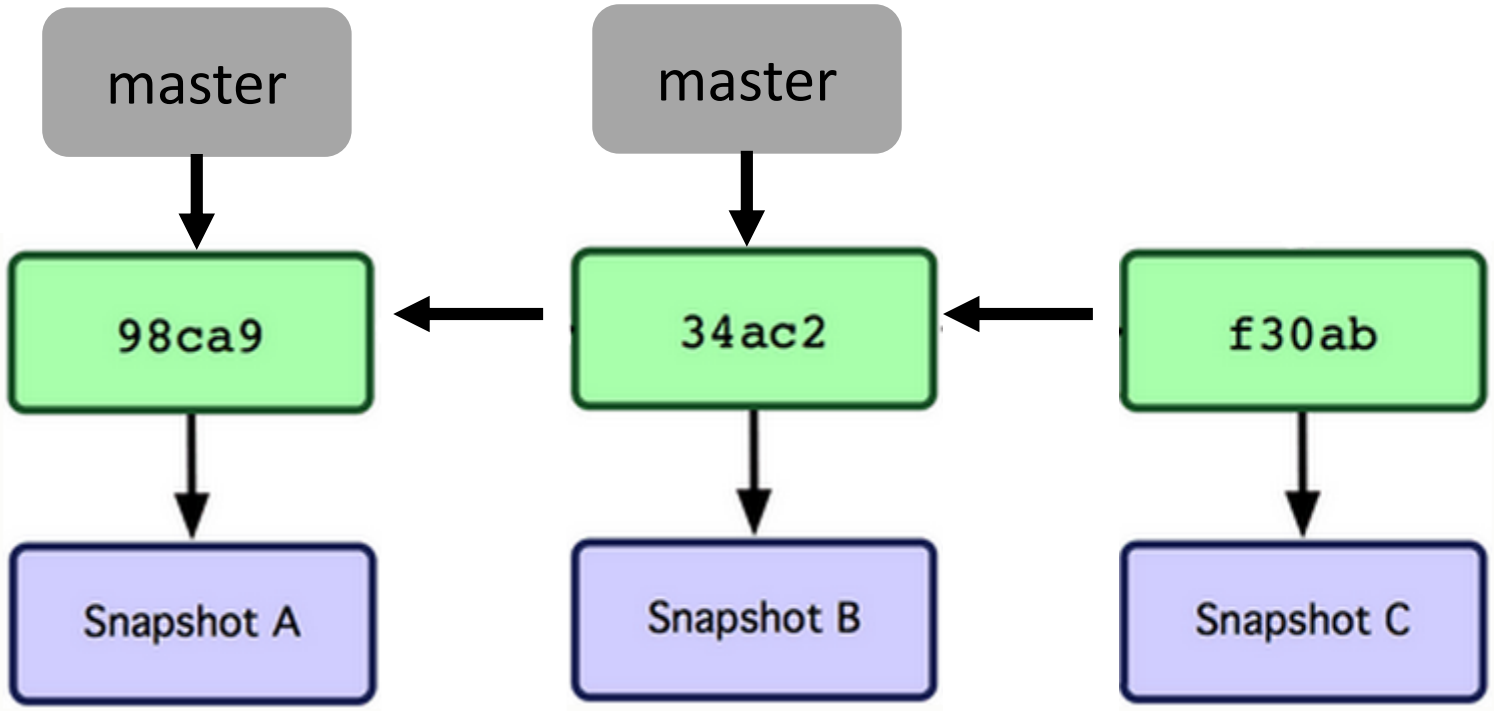
After 2 More Commits…



What Is a Branch?

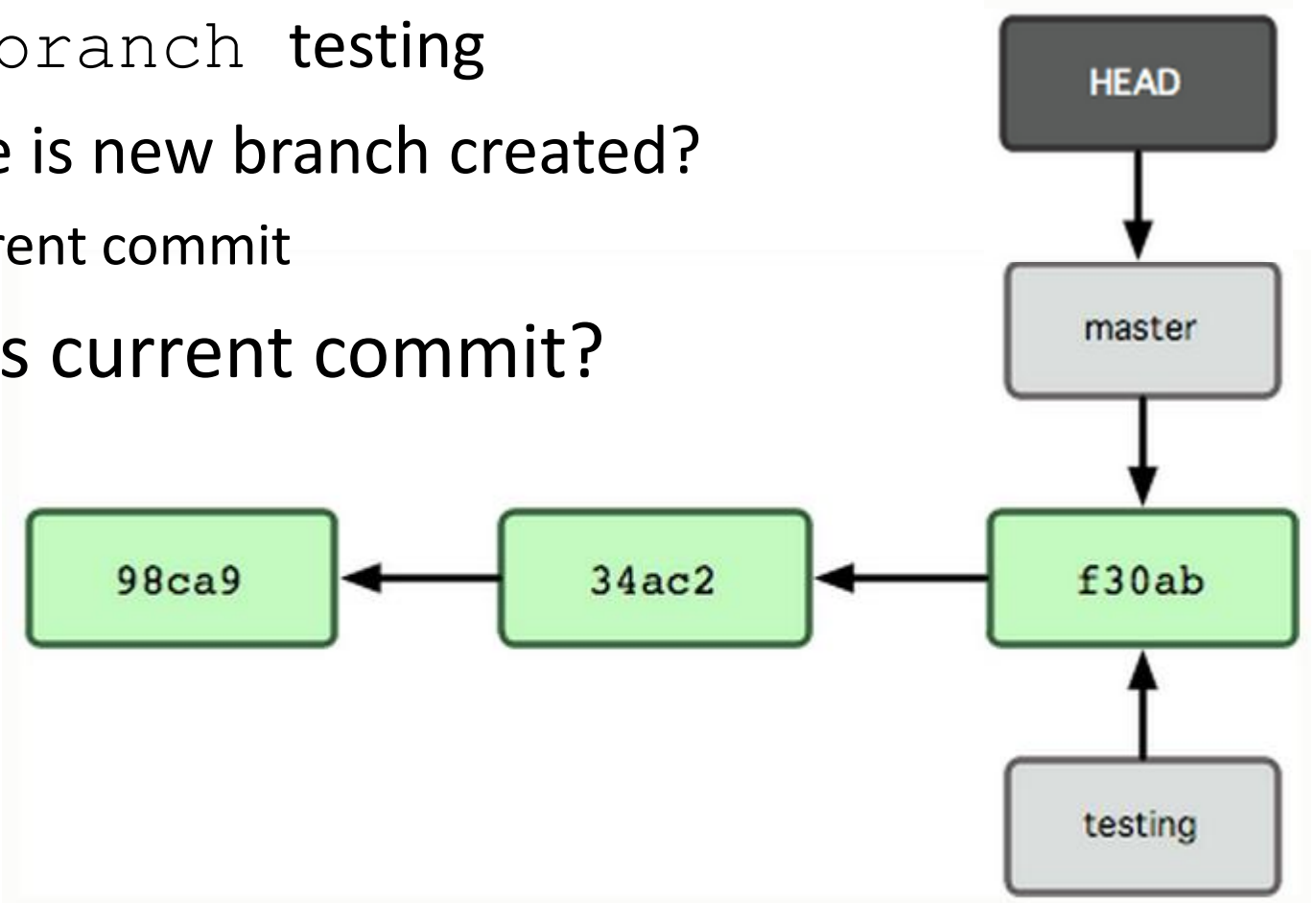
- A pointer to one of the commits in the repo (head) + all ancestor commits
- When you first create a repo, are there any branches?
  - Default branch named ‘master’
- The default master branch
  - points to last commit made
  - moves forward automatically, every time you commit

Where Is Master?



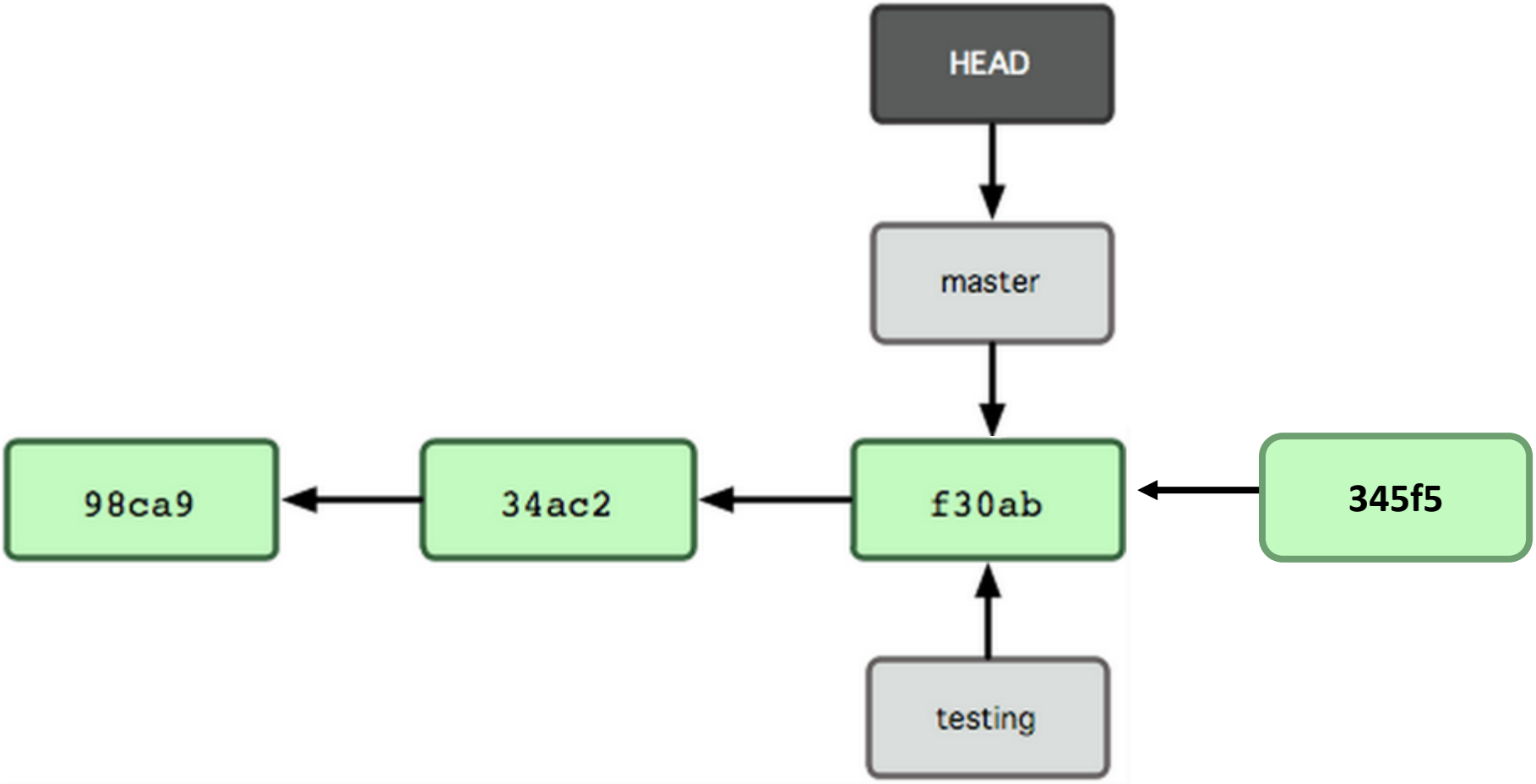
New Branch

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



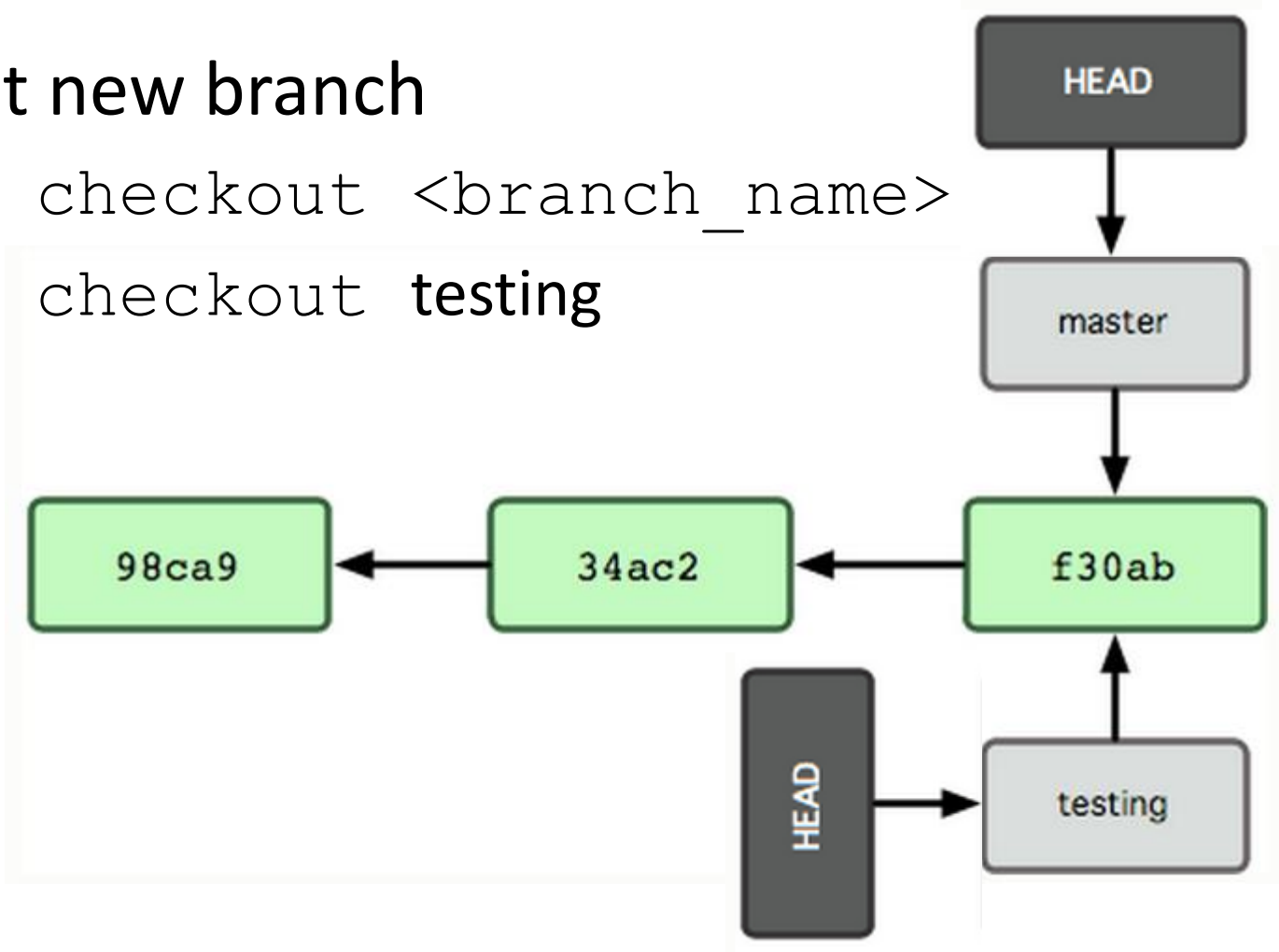
New Commit

- What happens if we make another commit?

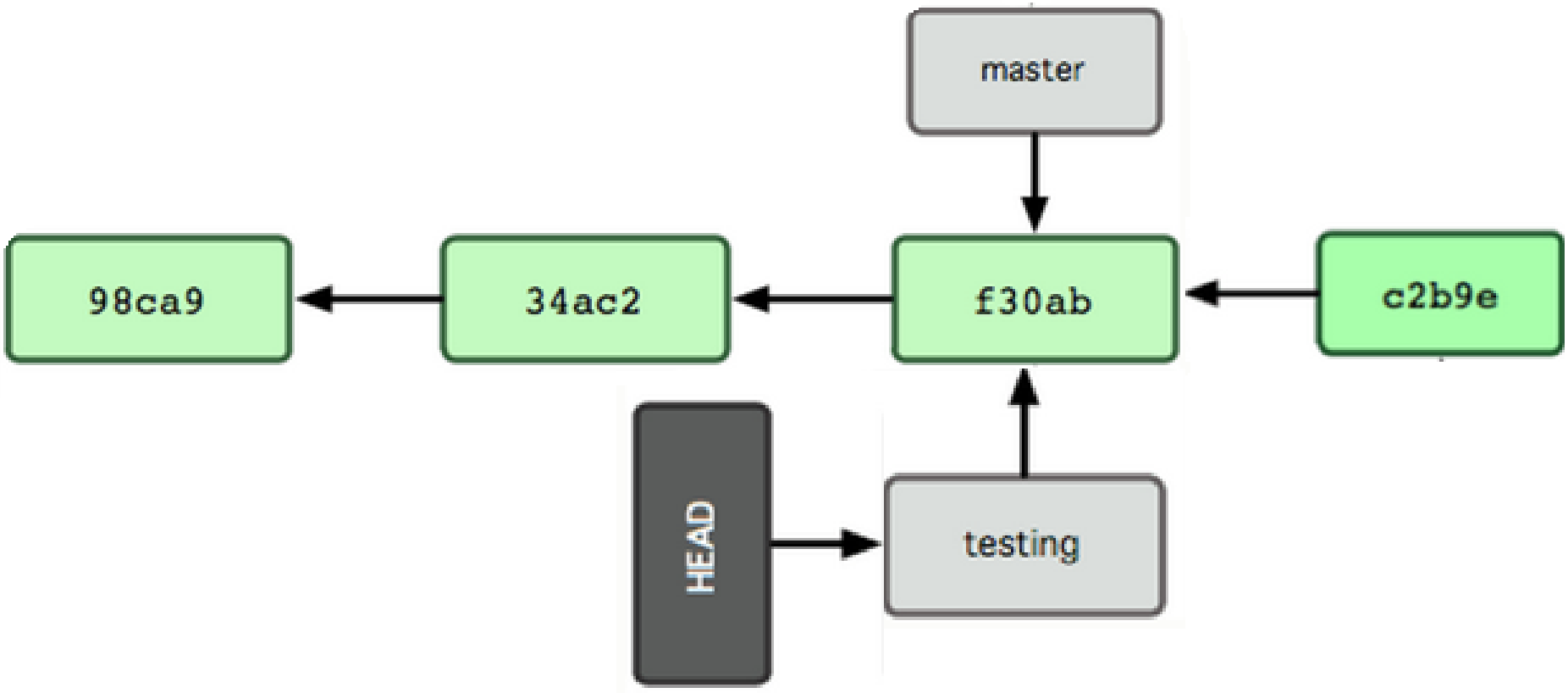


Switching to New Branch

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



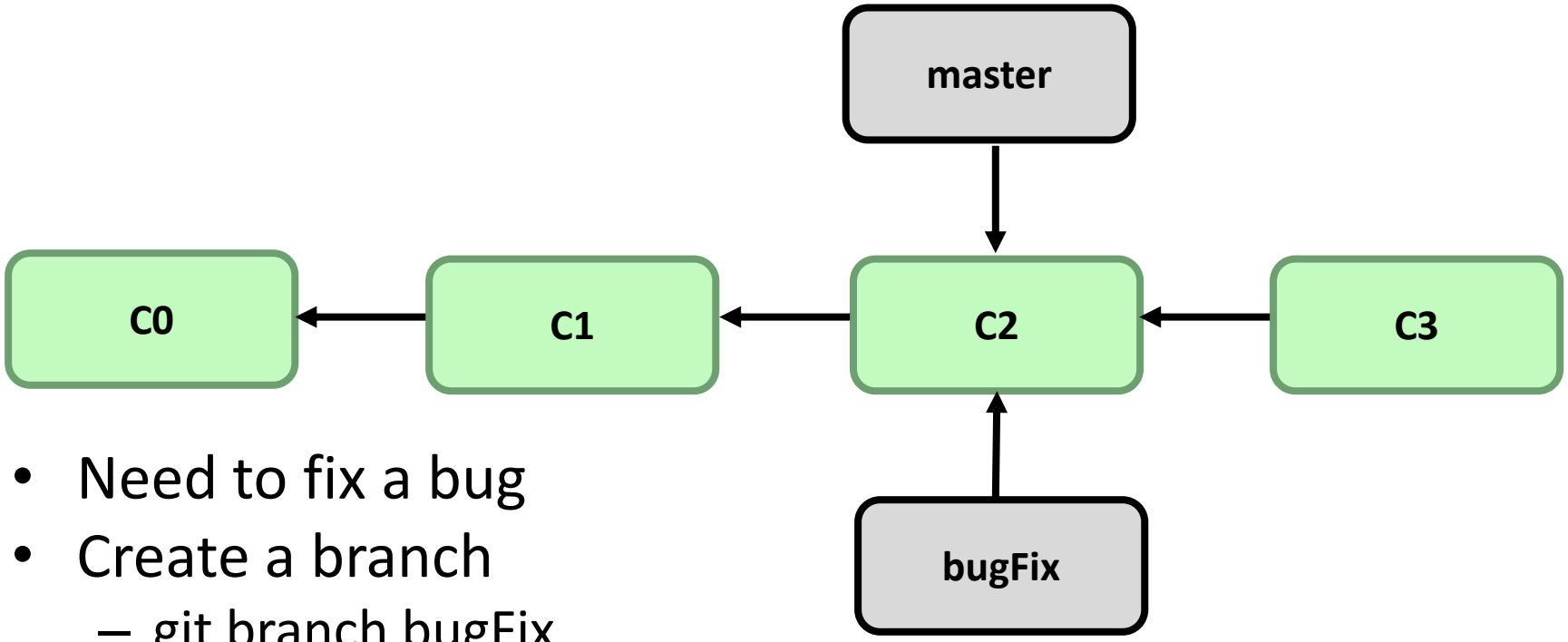
Commit After Switch



Why Branching?

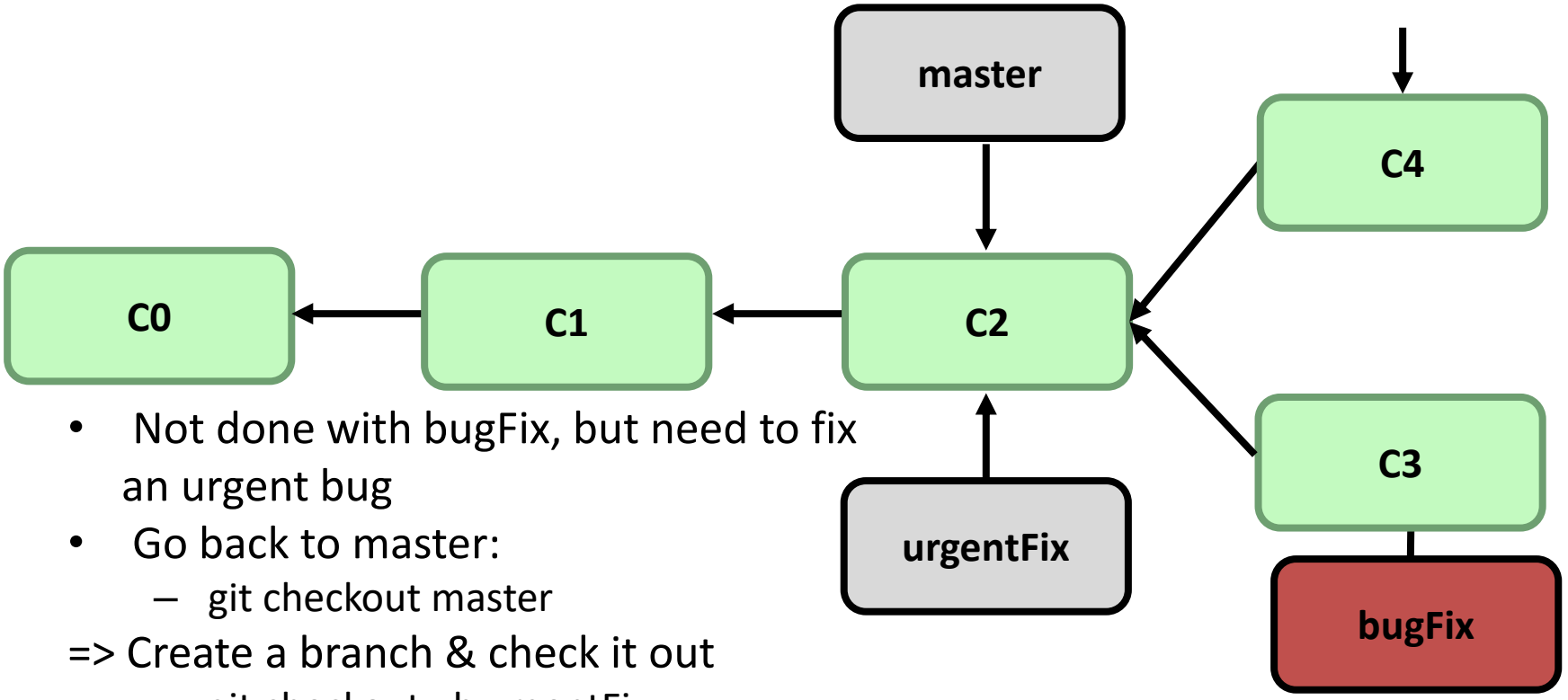
- Experiment with code without affecting main branch
- Separate projects that once had a common code base
- 2 versions of the project

Merging I



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

Merging II



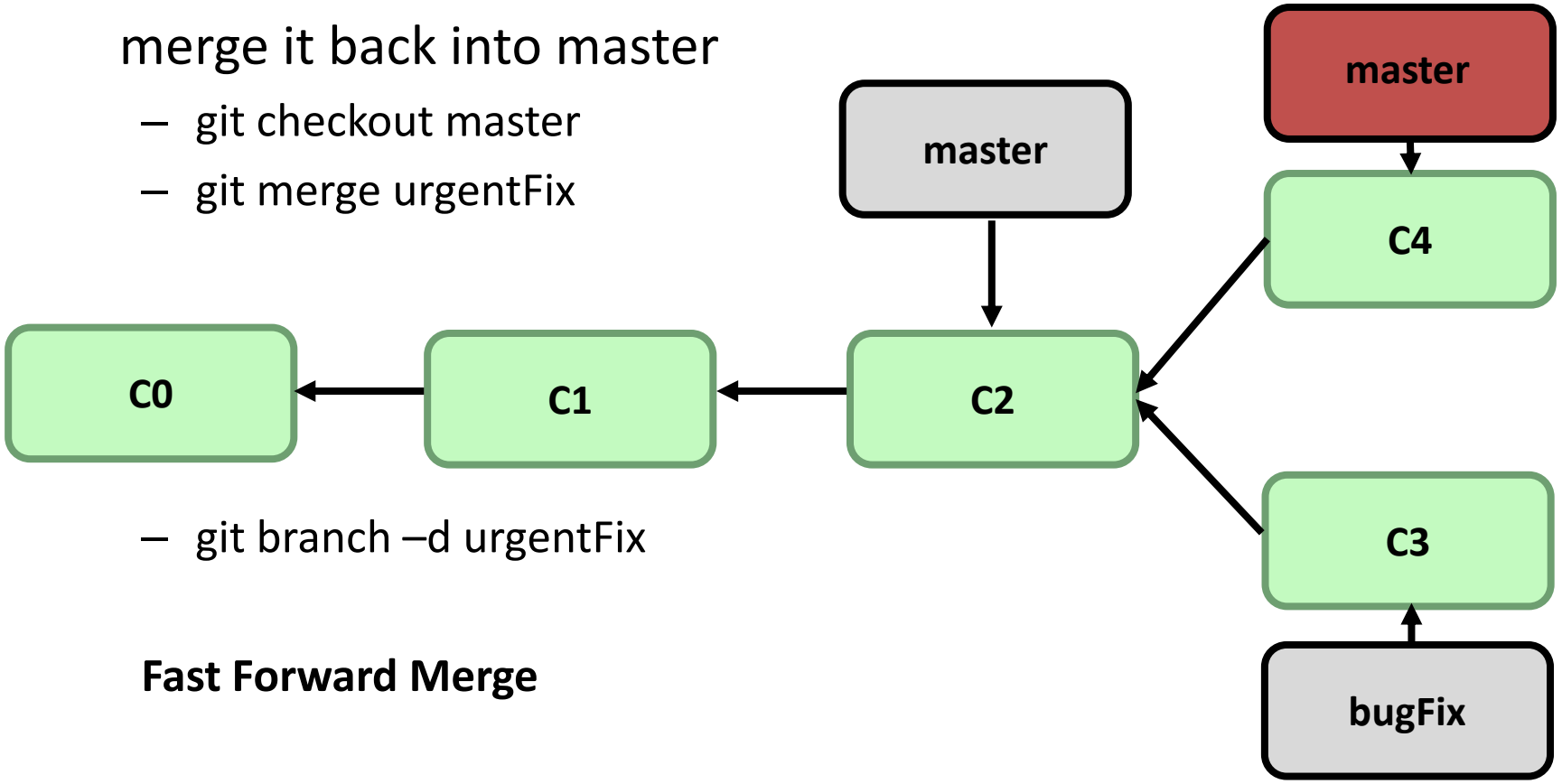
- Not done with bugFix, but need to fix an urgent bug
- Go back to master:
  - git checkout master

=> 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

Fast Forward Merge

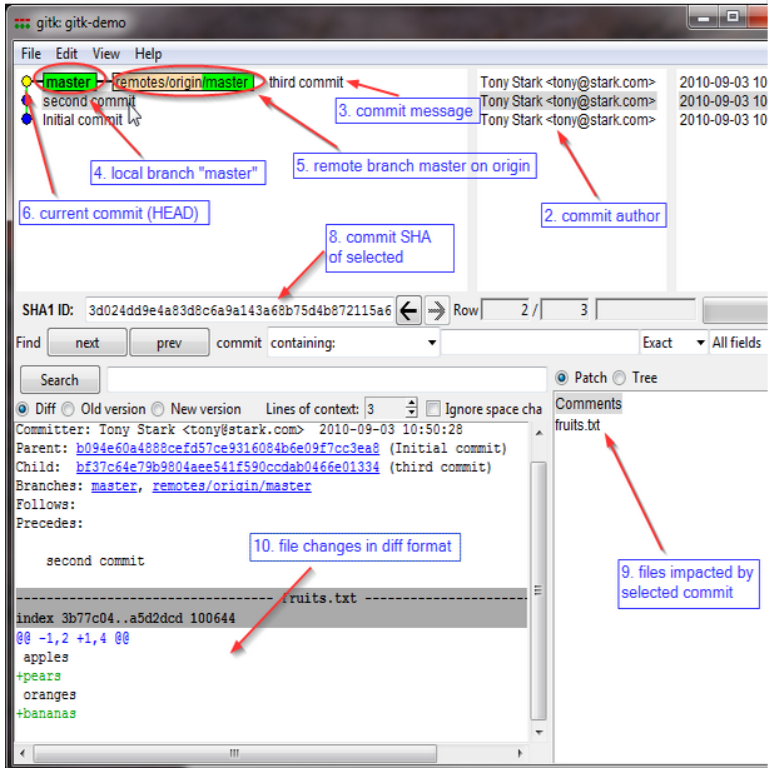
Homework 4

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
- Create Changelog entry (C-x 4 a)
- 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 temporary branch (“partner”)
  - Apply patch with `git am` command: `git am <` formatted-patch.txt
  - Build and test with `make check`

Gitk

- A repository browser
  - Visualizes commit graphs
  - Used to understand the structure of the repo



Need X11 forwarding

Tutorial here