CS 35L- Software Construction Laboratory

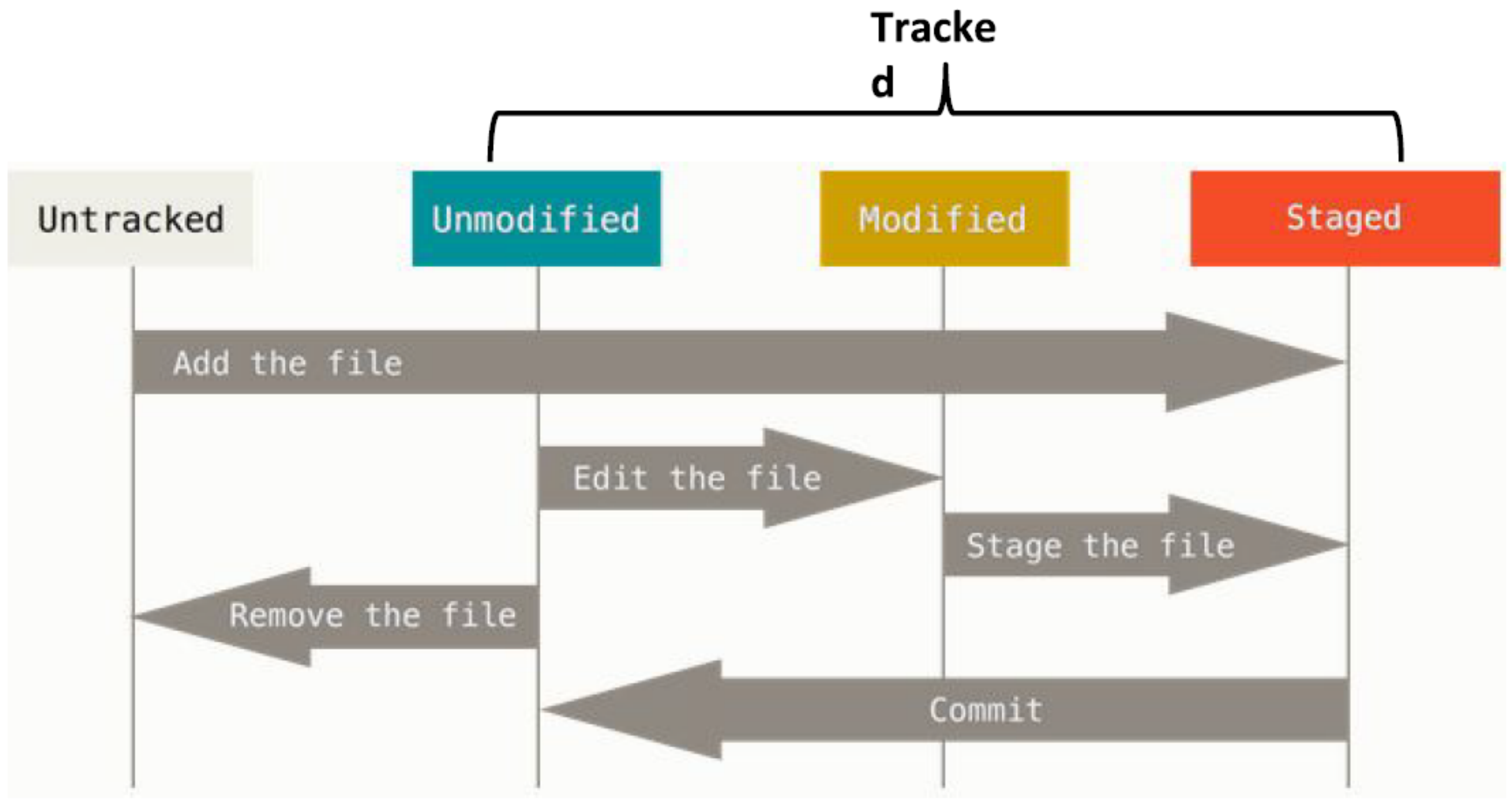
Fall 18

TA: Guangyu Zhou

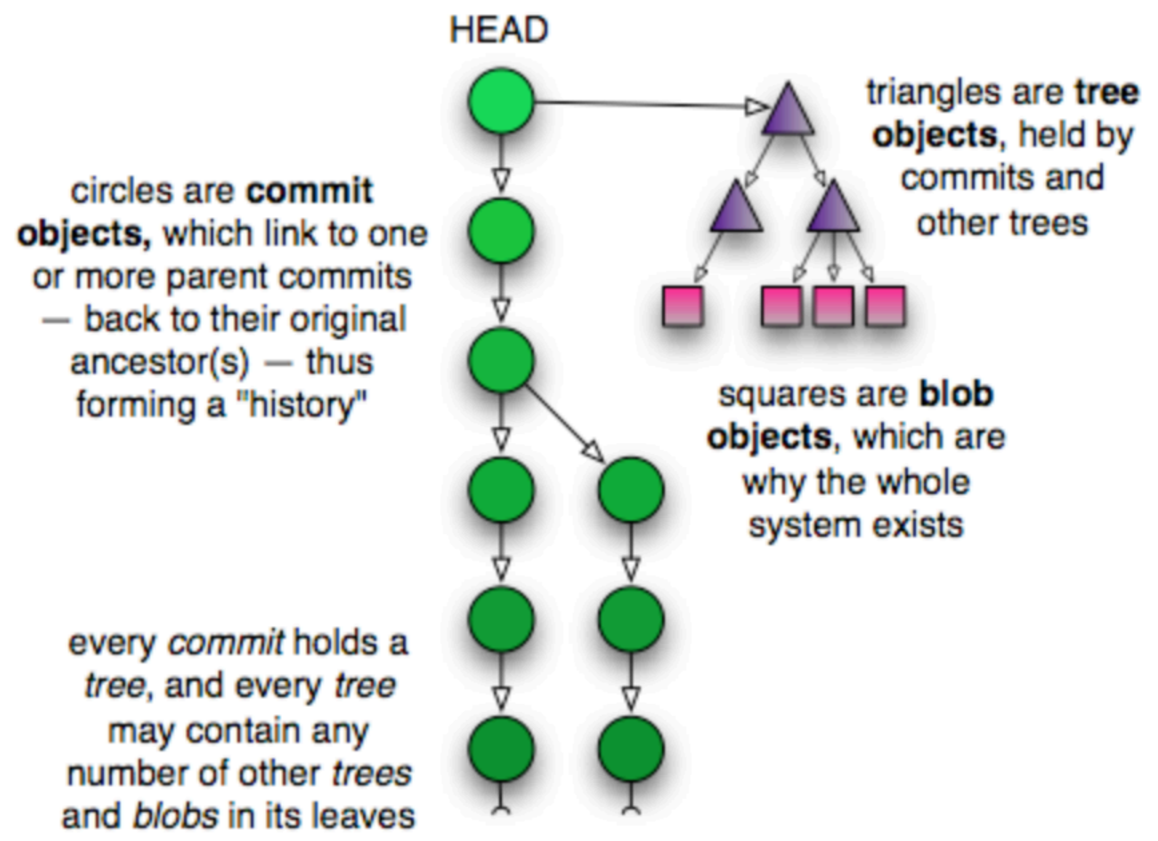
Change Management

Week 10

Git File Status



Git Repository Objects in one picture


HEAD
triangles are **tree objects**, held by commits and other trees
circles are **commit objects,** which link to one or more parent commits — back to their original ancestor(s) — thus forming a "history"
squares are **blob objects**, which are why the whole system exists
every *commit* holds a *tree*, and every *tree* may contain any number of other *trees* and *blobs* in its leaves

Undoing what is done

- **git checkout**
  - Used to checkout a specific version/branch of the tree
  - git rebase master (returns to current working version)
- **git revert**
  - Reverts a commit
  - Does not delete the commit object, just applies a patch
  - Reverts can themselves be reverted!
- **Git never deletes a commit object**
  - It is very hard to lose data

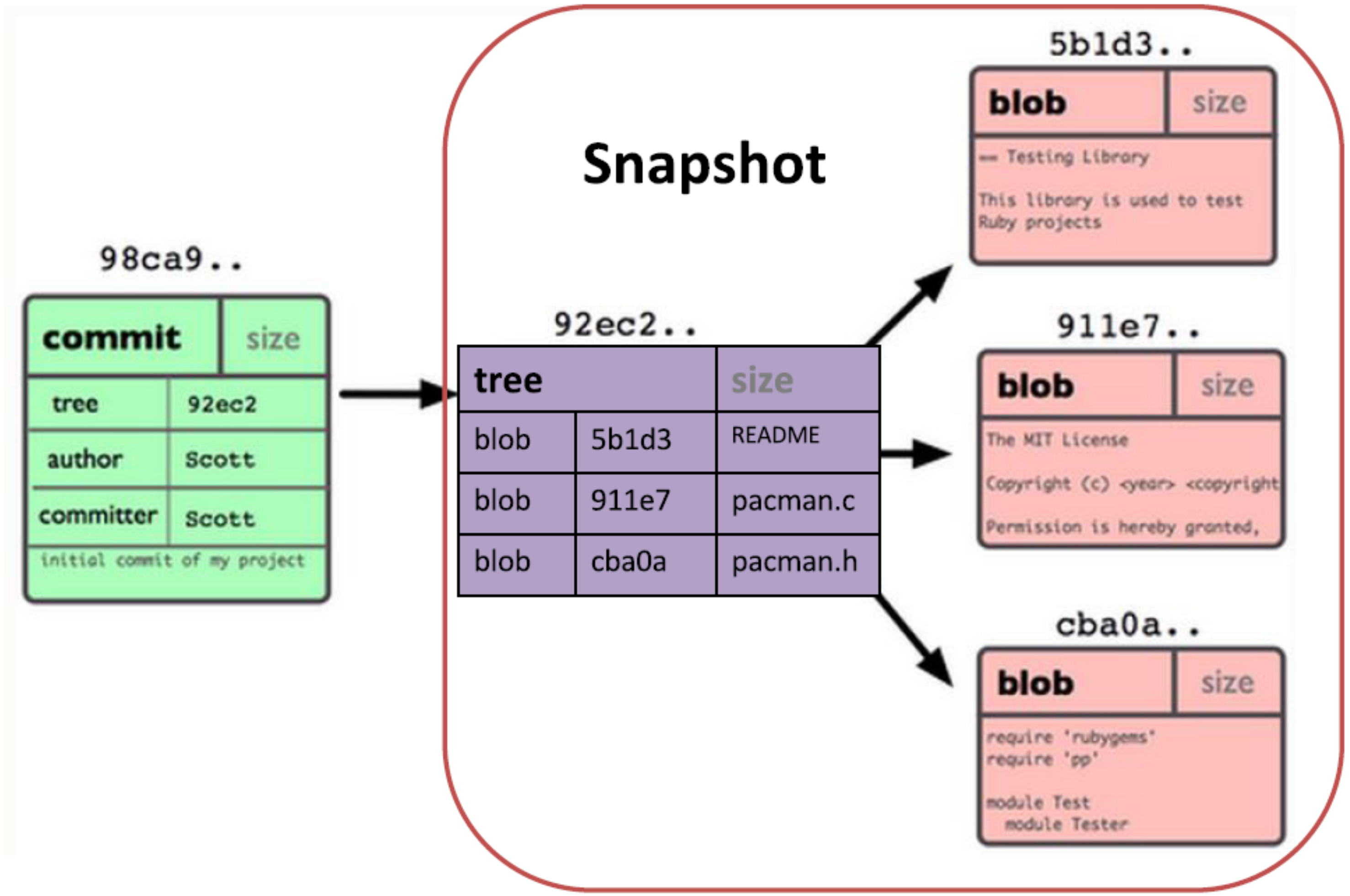
More git commands

- Reverting
  - **git checkout HEAD main.cpp**
    Gets the HEAD revision for the working copy
  - **git checkout -- main.cpp**
    Reverts changes in the working directory
  - **git revert**
    Reverts commits (this creates new commits)
- Cleaning up untracked files
  - **git clean**
- Tagging
  - Human readable pointers to specific commits
  - **git tag -a v1.0 -m ‘Version 1.0’**
    This will name the HEAD commit as v1.0

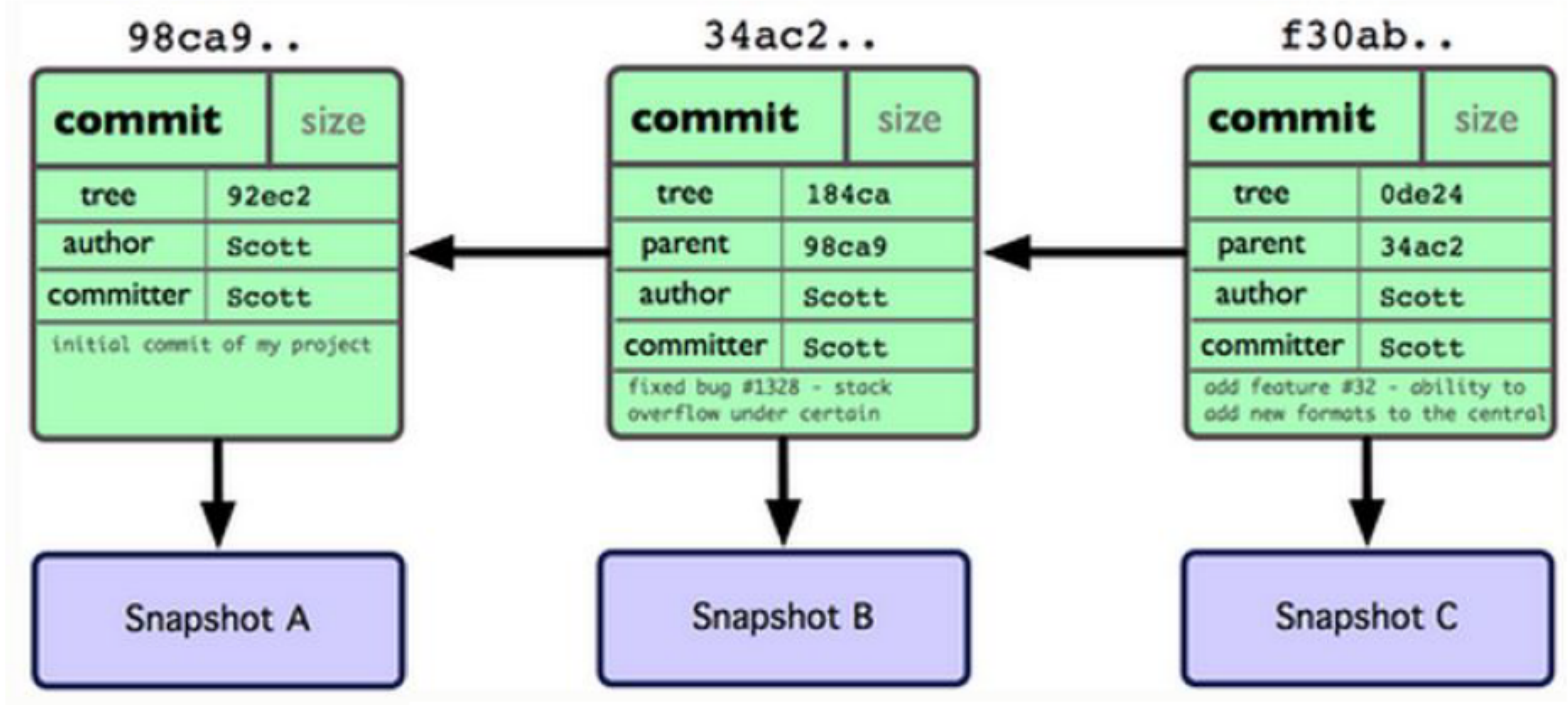
Different names of commit

- **branchname** — As has been said before, the name of any branch is simply an alias for the most recent commit on that “branch”. This is the same as using the word HEAD whenever that branch is checked out.
- **tagname** — A tag-name alias is identical to a branch alias in terms of naming a commit. The major difference between the two is that tag aliases never change, whereas branch aliases change each time a new commit is checked in to that branch.
- **HEAD** — The currently checked out commit is always called HEAD.

Git Repo Structure



After add two more commits

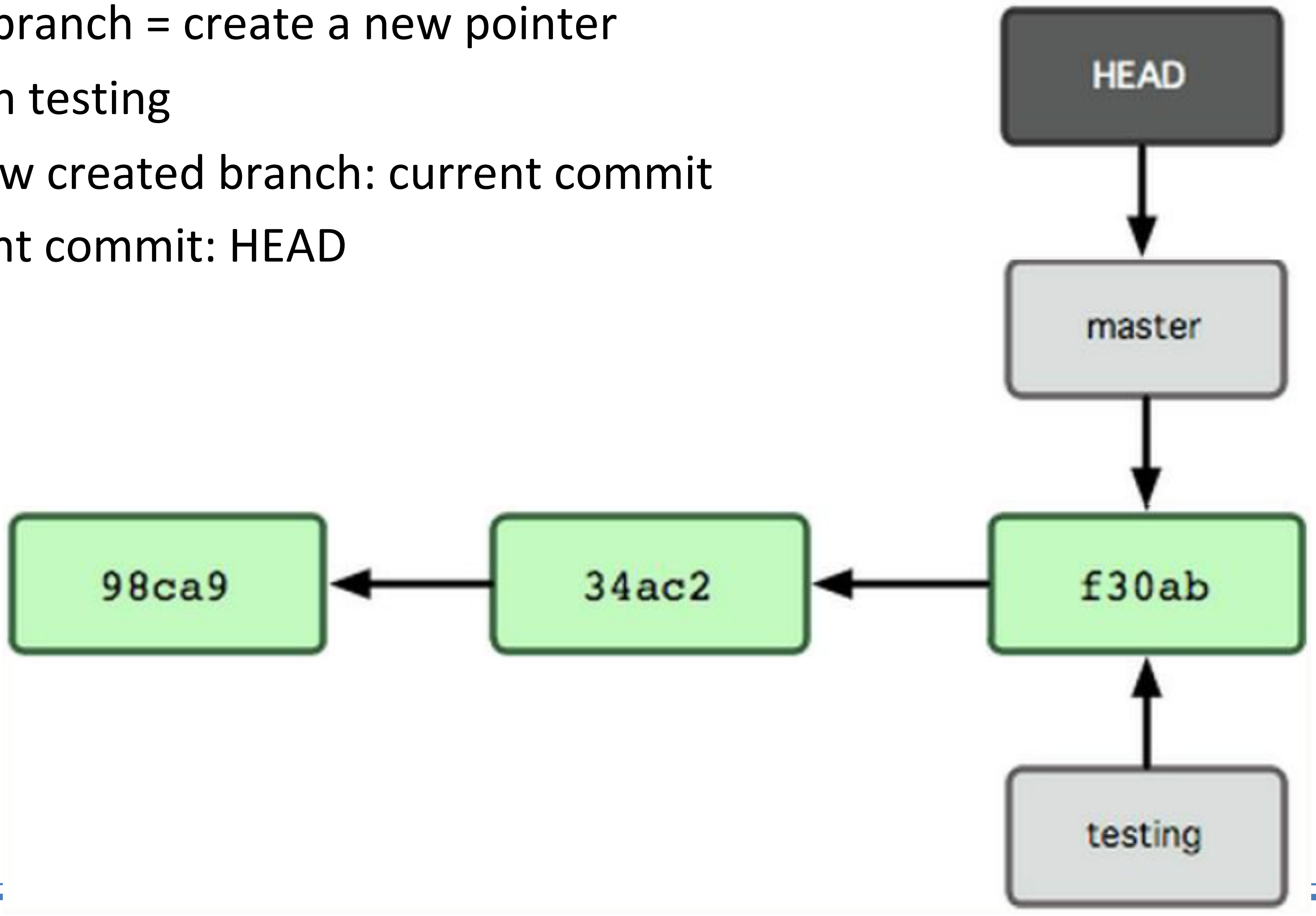


The concept of branch

- A pointer to one of the commits in the repo(head) and all ancestor commits
- When you create a new repo, the default branch is named master
- The default master branch
  - Points to last commit made
  - Move forward automatically, every time you commit

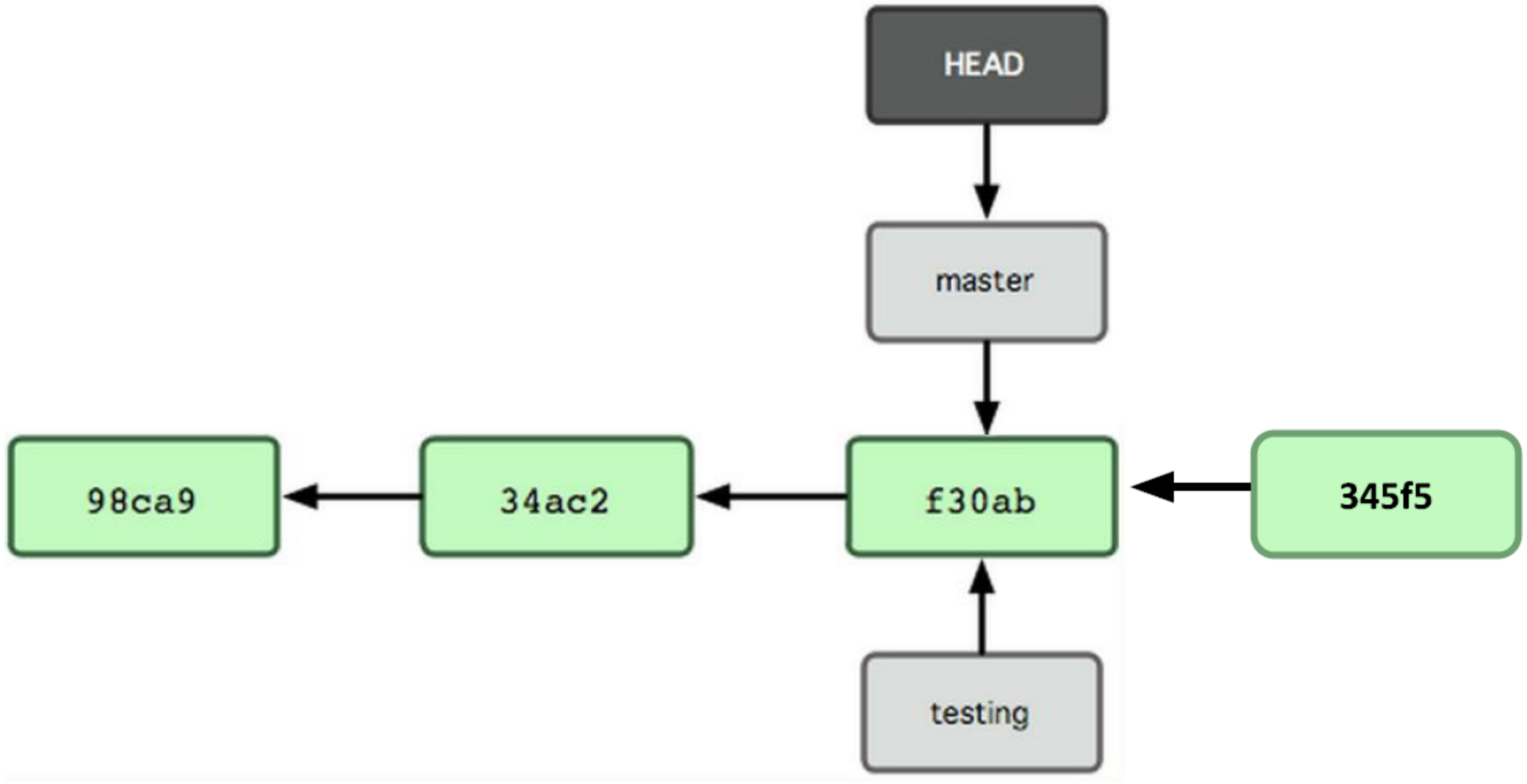
New Branch

- Create a new branch = create a new pointer
  - $ git branch testing
  - Place of new created branch: current commit
- Place of current commit: HEAD



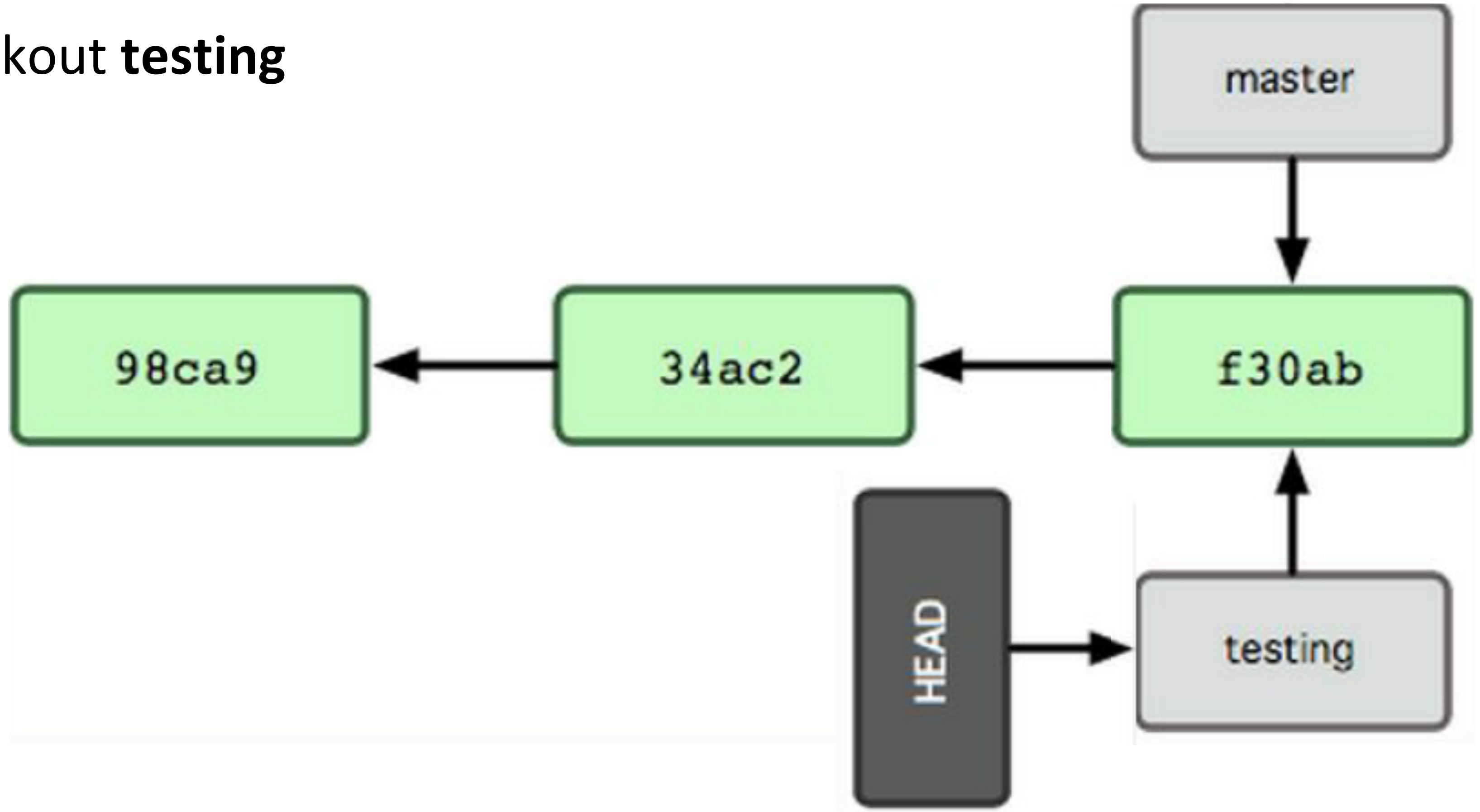
New Commit

- When make a new commit

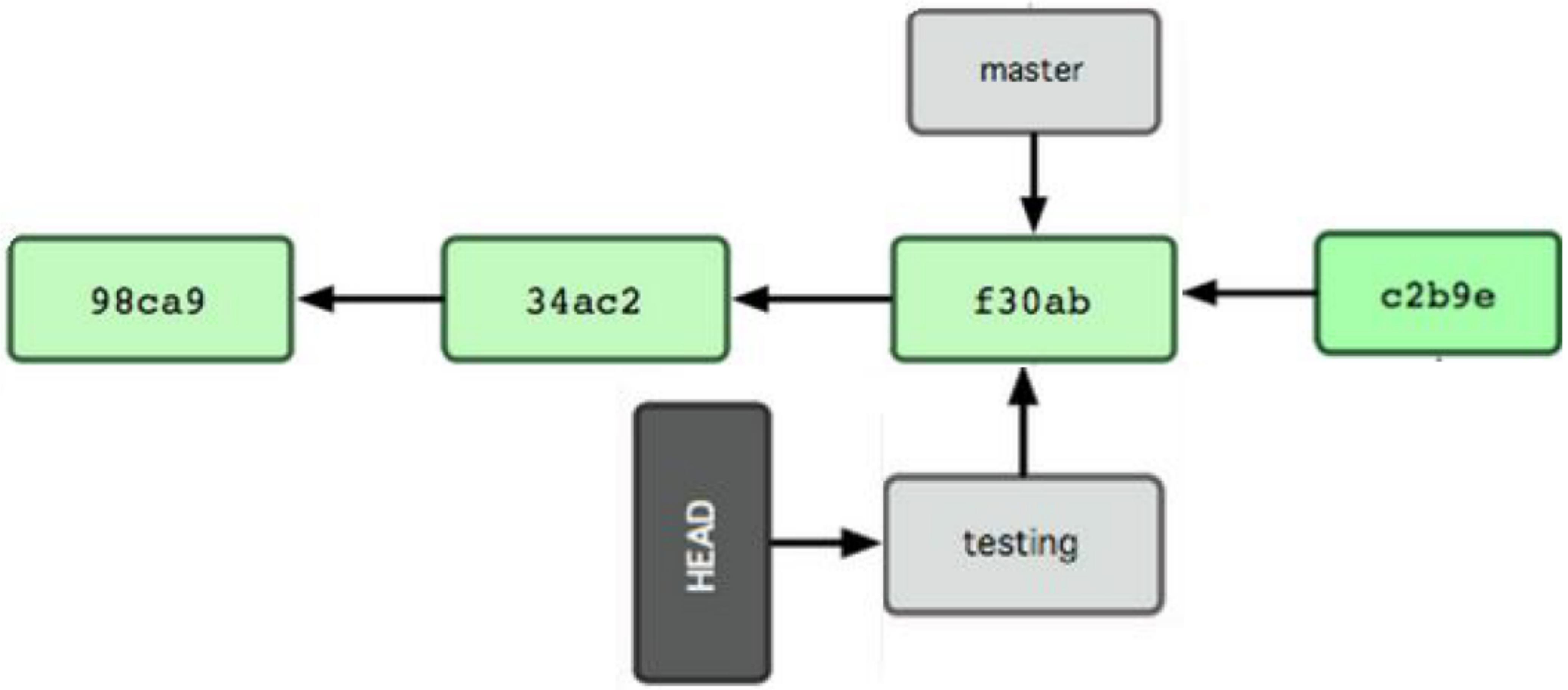


Switch to new branch

- 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
- Two versions of the project

Basic Merging

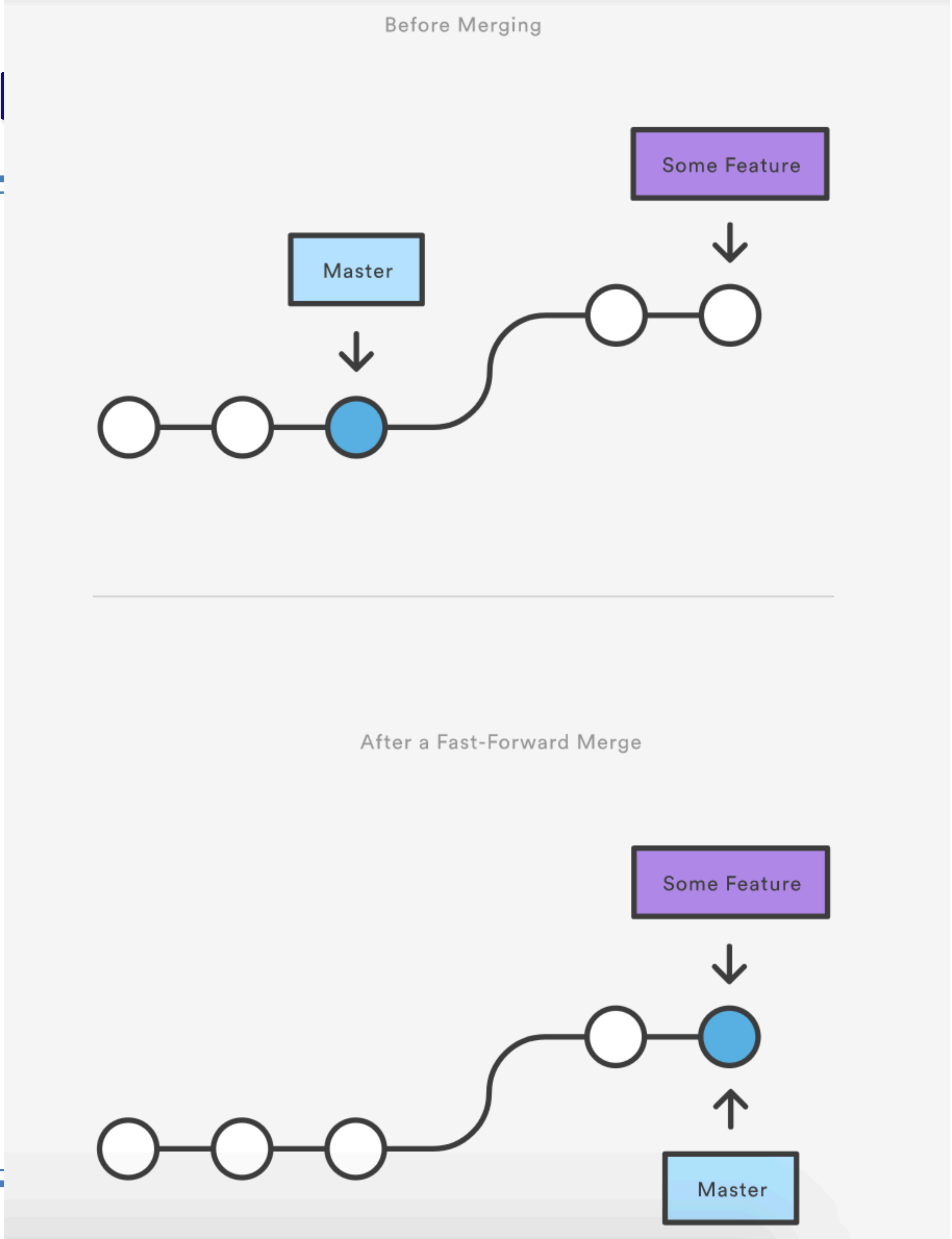
- To merge another branch into current branch

  git merge <branch_name>

- Two types
  - Fast-forward
  - Three-way

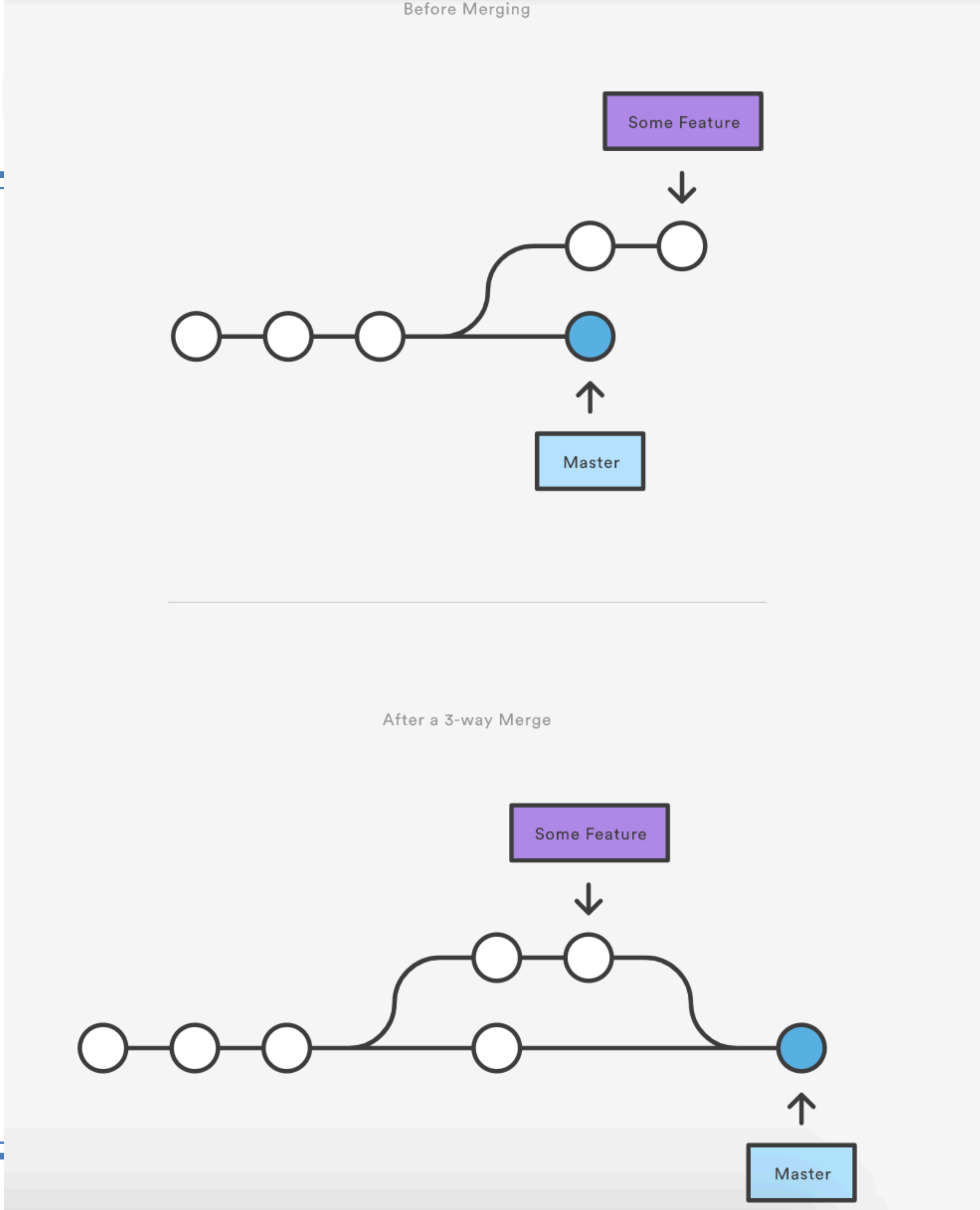
Fast forward

- merge one commit with a commit that can be reached by following the first commit's history
- Git merge will simply move the pointer



Three way

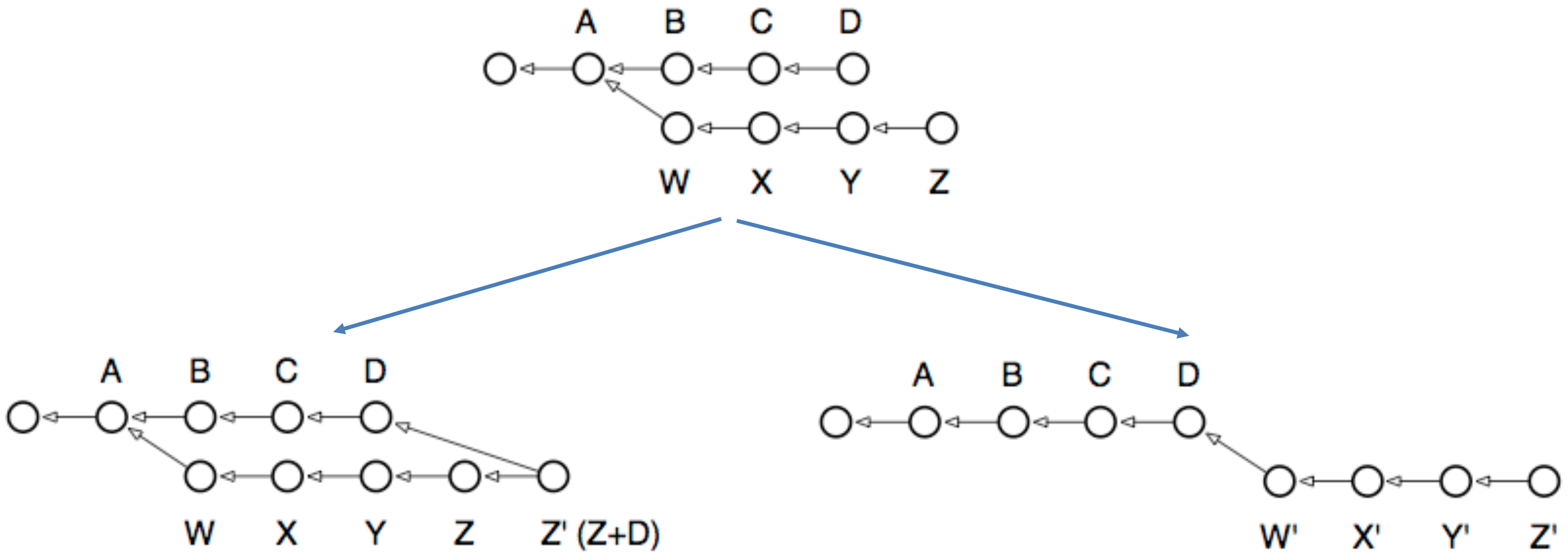
- Three parties: two snapshots pointed to by the branch tips and the common ancestor of the two
- Git create a new snapshot from the merge and automatically create a new commit pointing to it
- Git will find the appropriate common ancestor automatically



Basic merge conflict

- Usually git will do merge automatically
- Conflict arises when you changed the same part of the same file differently in the two branches you’re merging together
- The new commit object will not be created
- You need to resolve conflicts manually

Branching vs Rebase



Homework 9

- Publish the patch made in lab 9
- Create a new branch "quote" of version 3.0
  - *$ git checkout v3.0 –b quote*
- Use patch from lab 9 to modify this branch
  - *$ patch –pnum < quote-3.0-patch.txt*
- Modify the change log in *diffutils* directory
  - Add entry for your changes into those in the change log

Homework 9

- Commit changes to the new branch

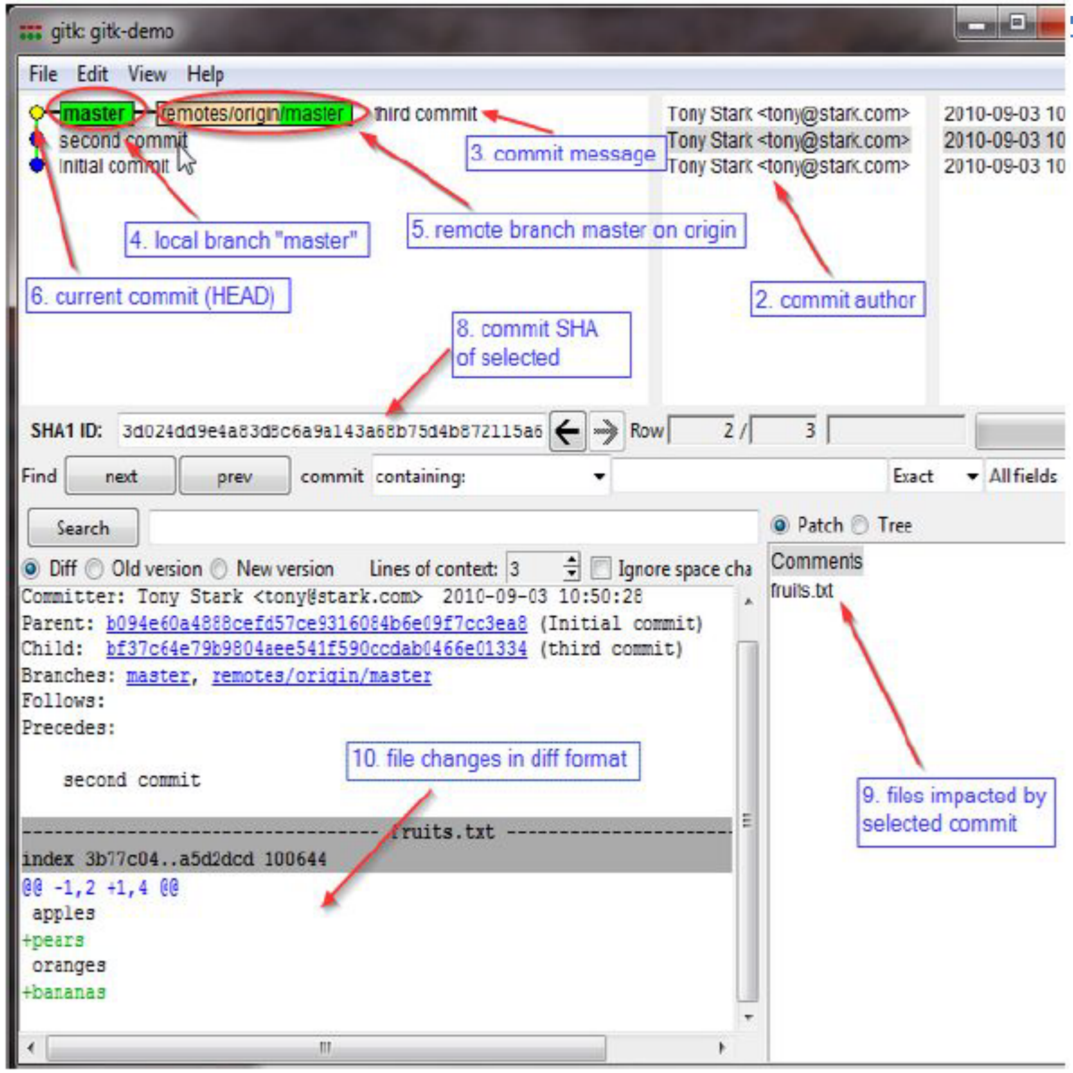
  $ git add . *$ git commit –F [change log file]*
- Generate a patch that other people can use to get your changes

  $ git format-patch -[num] –stdout > [patch file]
- Test your partners patch
  - Choose a partner from this class, include his/her name and UID in your report
  - Apply patch with command *git am*
  - Build and test with command *make check*

Homework 9 -- Gitk

- A repository browser
  - Visualize commit graphs
  - Understand the structure of repo
  - Tutorial: [Use gitk to understand git]

  See supplement materials


gitk: gitk-demo
File Edit View Help
master
remotes/origin/master
third commit
second commit
initial commit
Tony Stark <tony@stark.com>
Tony Stark <tony@stark.com>
Tony Stark <tony@stark.com>
2010-09-03 10
2010-09-03 10
2010-09-03 10
3. commit message
4. local branch "master"
5. remote branch master on origin
6. current commit (HEAD)
2. commit author
8. commit SHA of selected
SHA1 ID: 3d024dd9e4a83d8c6a9a143a68b75d4b872115a6
Row 2 / 3
Find next prev commit containing: Exact All fields
Search
Patch Tree
Diff Old version New version Lines of context: 3 Ignore space cha
Comments
fruits.txt
10. file changes in diff format
9. files impacted by selected commit

```
Committer: Tony Stark <tony@stark.com>  2010-09-03 10:50:28
Parent: b094e60a4888cefd57ce9316084b6e09f7cc3ea8 (Initial commit)
Child:  bf37c64e79b9804aee541f590ccdab0466e01334 (third commit)
Branches: master, remotes/origin/master
Follows:
Precedes:

    second commit

------------------------------- fruits.txt -------------------------------
index 3b77c04..a5d2dcd 100644
@@ -1,2 +1,4 @@
 apples
+pears
 oranges
+bananas
```

Homework 9 -- Gitk

- Usage
  - ssh –X for linux and MacOS
  - Select “X11” option if using putty (Windows)
  - See supplement materials [Putty X11 forwarding]
- Run gitk in the ~/eggert/src/gnu/emacs directory
  - Need first update your path

    export PATH=/usr/local/cs/bin:$PATH
  - Run X locally before running gitk

    Xming on Windows, **Xquartz** on Mac