git

Outline

- Version Control System (VCS) basics
- Git terminology & concepts
- Basic Git commands
- Branches in Git
- Git over the network (time permitting)

Why do I need version control?

- How many lines of code was the biggest project you've worked on thus far?
- Were you collaborating with others on that project?
- How did you share code?
- How did you keep track of different changes?
- Were you ever in a place where the project once worked but it didn't when you needed it?

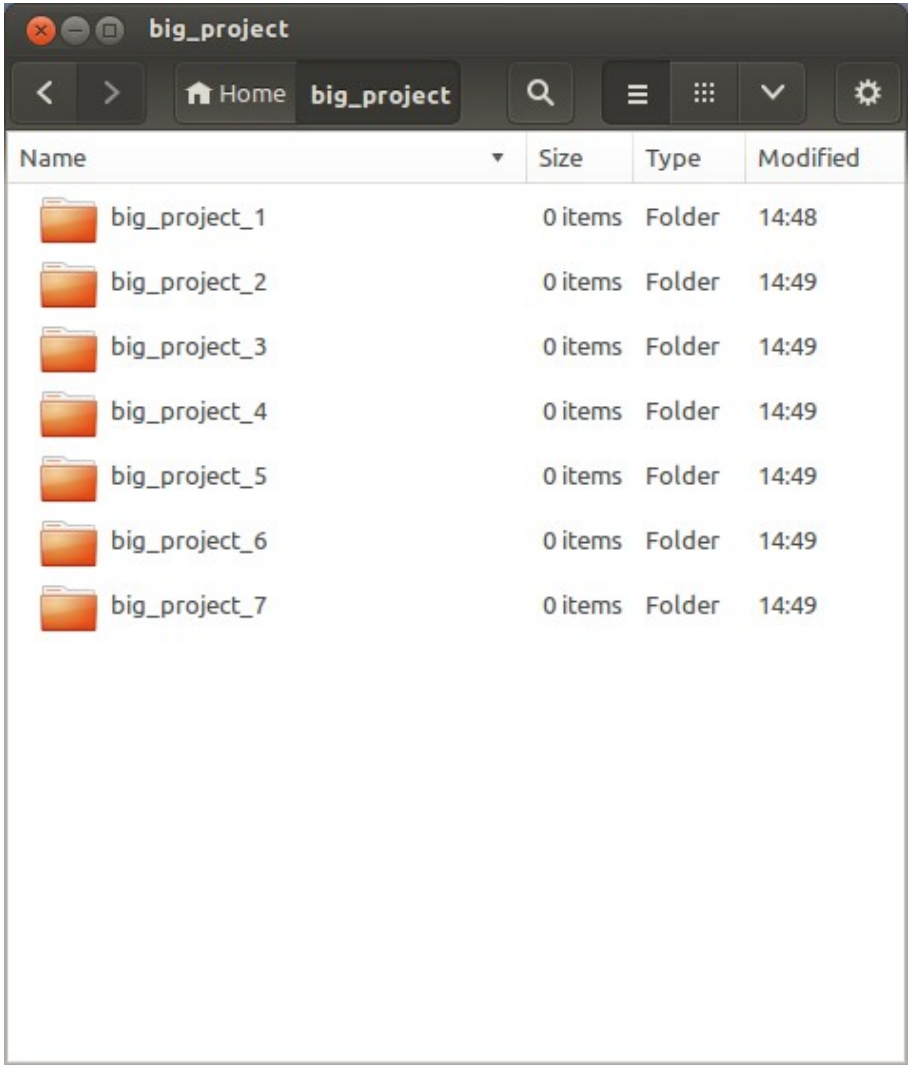
What does VCS do for me?

What we want from a VCS:

- Save a history of changes we make
  - Visibly show the history
  - Allow us to revert to an older state
- Aid in collaboration with others
  - Should include tools for combining code together
- Aid in creating backups of our work

The local approach

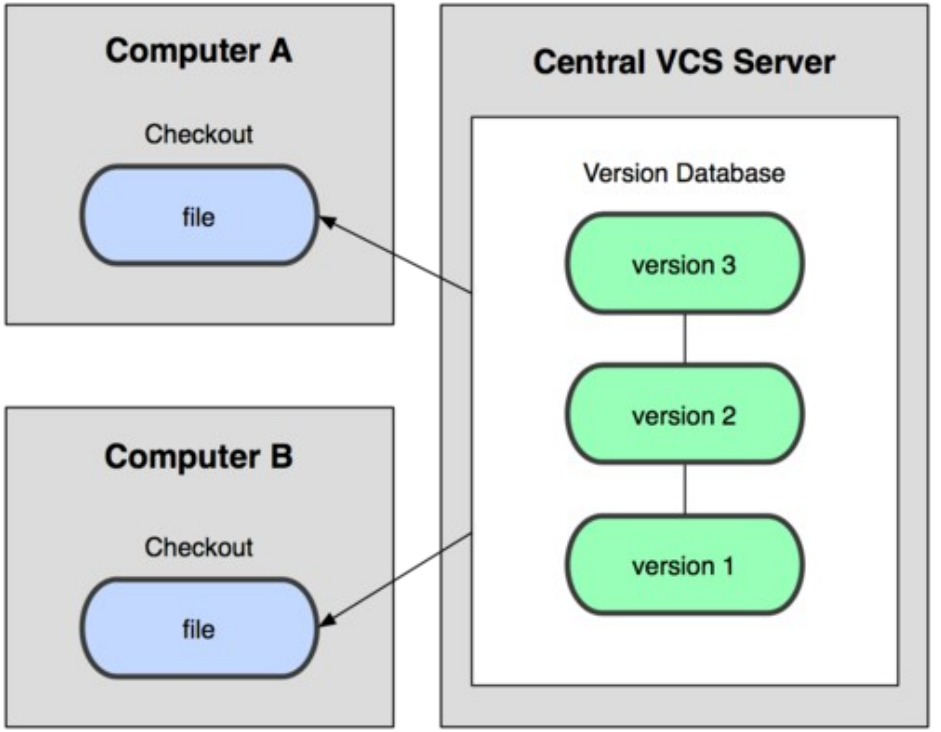
- No special program, just a collection of folders
- Versions identified by a number or date
- Does this look familiar?



IF YOU COULD QUIT THAT
THAT'D BE GREAT
made on imgur

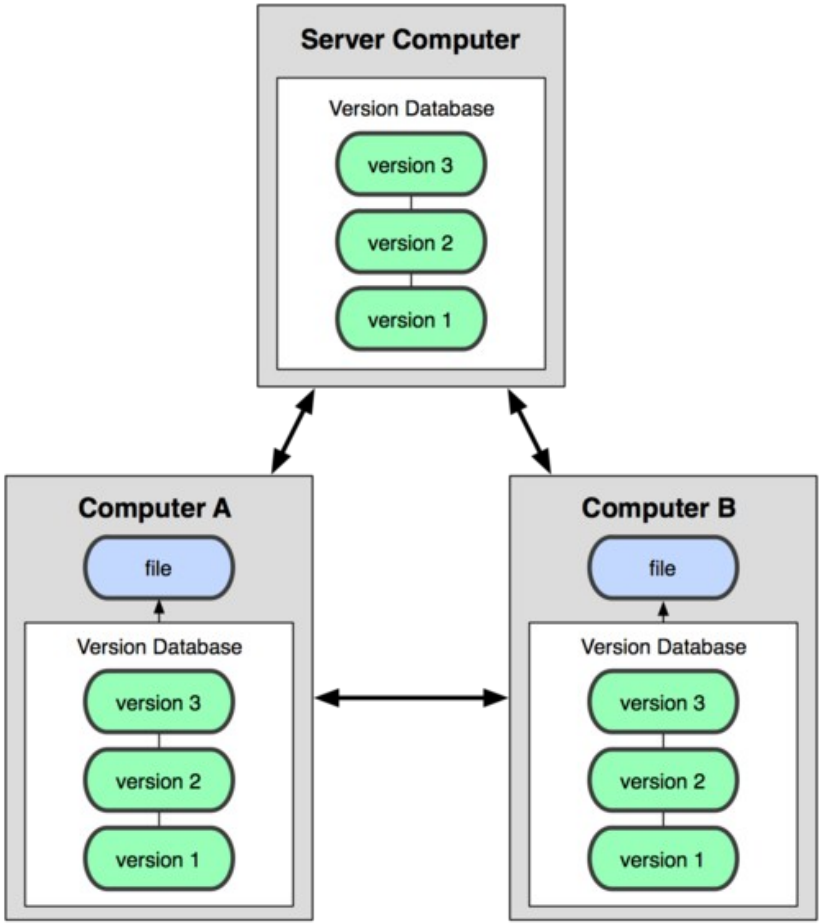
The centralized approach

- Server-Client model
- A single server stores all revisions
- Clients check-out the revision they want



The distributed approach

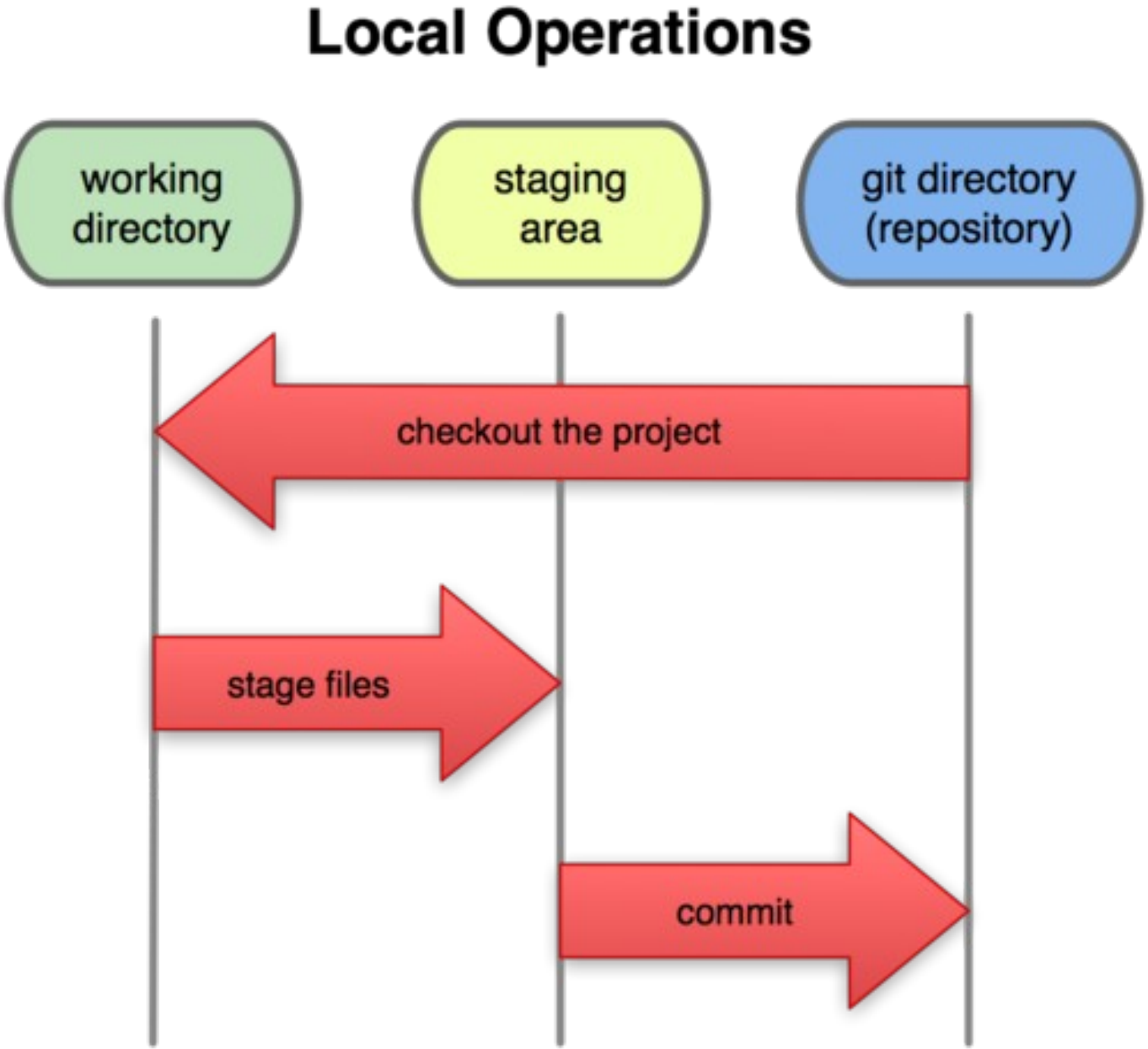
- Everyone has the full history of the project
- Possible to have more than one server
- Crashed server is only a minor speedbump
- Easy to recover from bad hard drive on server



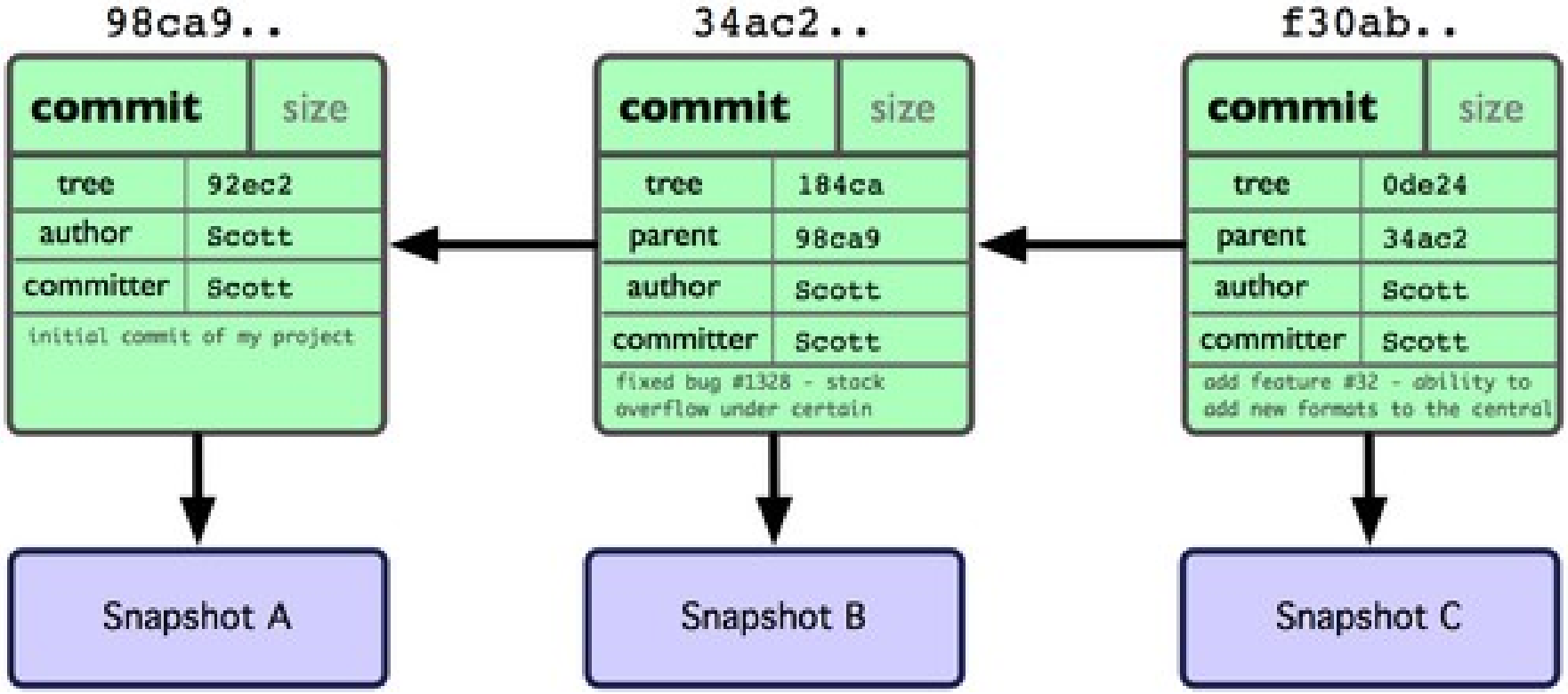
The three states of a tracked file

- Modified
  - The file has been modified after you checked it out
- Staged
  - You tell git when to stage a file(s)
  - Files must be staged before they are committed
- Committed
  - You tell git when to commit a file(s)
  - Committed files are recorded in a database

The three states of a file



Git creates a database of snapshots



- Each commit has data about the author, committer, time committed, notes, etc.
- Each commit also has references to the binary objects needed to recreate your project.

First steps

First thing you do in Git needs to be to tell Git who you are. This is needed for your work history when collaborating with others.

$ git config --global user.name “John Doe”

$ git config --global user.email “johndoe@aol.com”

Git Init

You have to tell git where you want to make a repository. This path will be the top directory of your project.

```
$ cd ~/school/ece473/lab/lab_1
$ git init
```

Git Status

The “git status” command tells you which branch you are on, which files are modified, and which are staged.

```
$ git status

# On branch master
#
# Initial commit
#
```

Git Status Example #2

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

#

#modified: hello.py

#

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

Git Status Example #2

(use "git checkout -- <file>..." to discard changes in working directory)

#

#modified: main.py

#

Untracked files:

(use "git add <file>..." to include in what will be committed)

#

#hello.pyc

Git Add

After adding new files or modifying existing ones we use the “git add” command to add files to the staging area.

```
$ vim main.c
$ vim fm_radio.c
$ git add *.c
```

Git Commit

The git commit command creates a new commit (aka snapshot) of your project with all the changes that are in the staging area (used git add to stage files).

```
$ git commit
```

Important flags:
-a: add all new/modified files and then commit
-m: write commit message in command

```
$ cp ~/math_lib/ .
$ git commit -a -m “Add math library to project.”
```

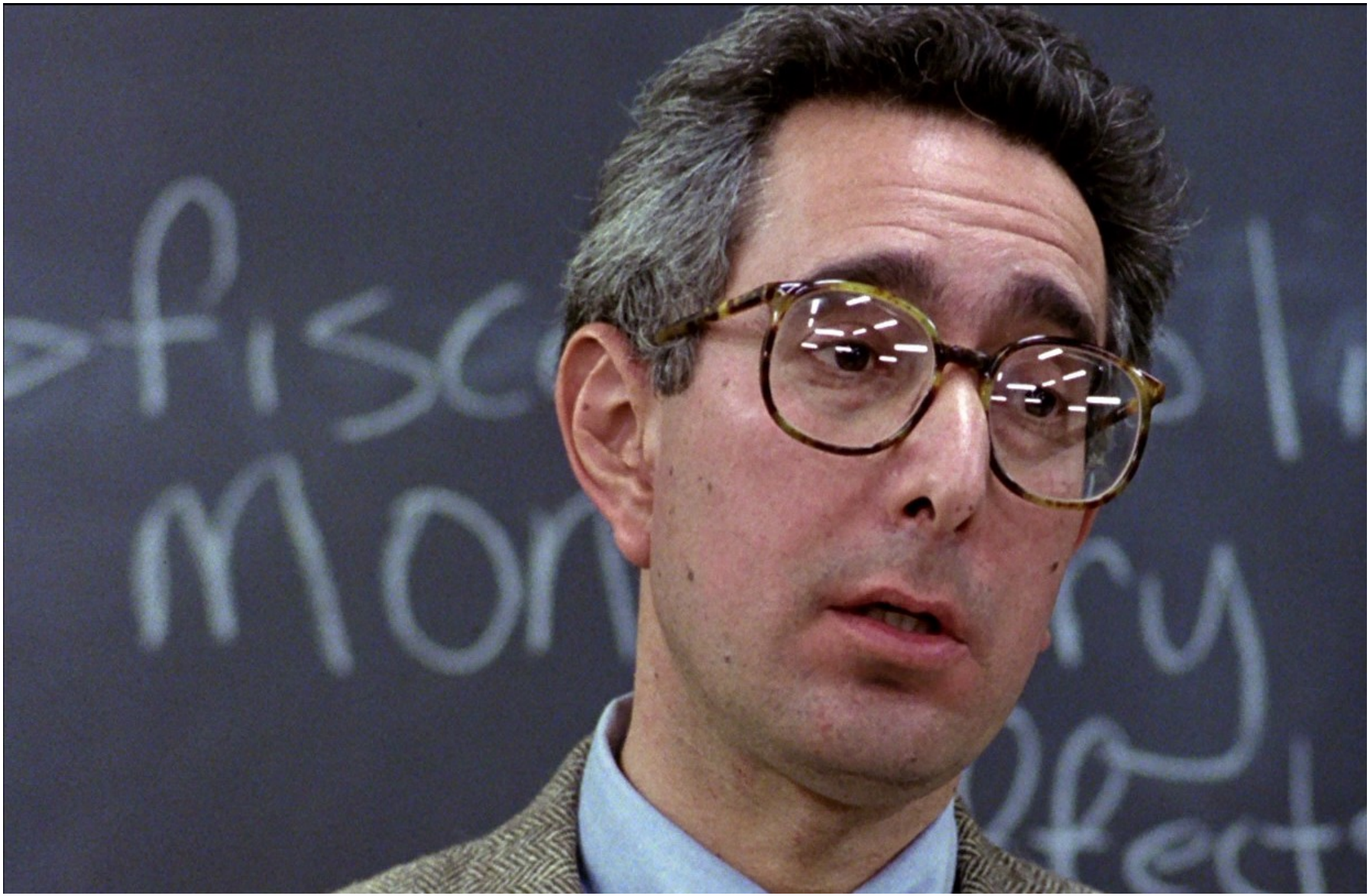
Git Log

The git log command prints to the terminal the history of your commit messages for the branch that you are currently on.

```
$ git log

Commit: e95513e4fe173ca6b0246d2a4c85057ee41b639c
Author:  Micah Losli <micah.losli@gmail.com>
Date:    (7 days ago) 2013-11-11 21:22:53 -0800
Subject: Add support for song options
```

Good so far?



In Class Exercise Part 1

Branch?

“Branching means you diverge from the main line of development and continue to do work without messing with the main line.”

-Scott Chacon, Pro Git

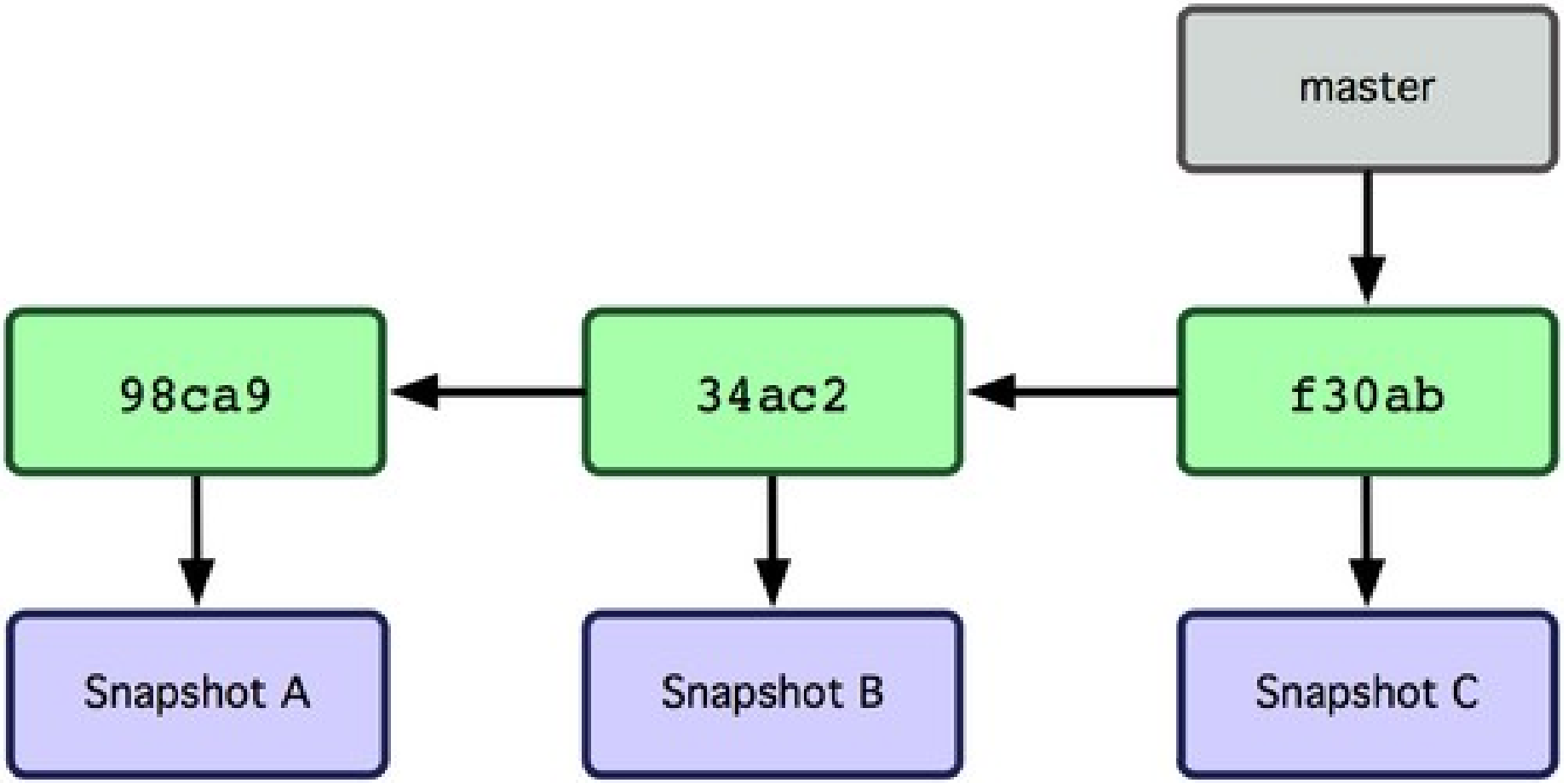
“Think of a branch as you would a copy of your poject folder: a safe place to experiment with new code.”

-Micah Losli

What is a branch?

- In most VCSs, a branch is actually a new copy of your project directory. With a large project, this is very expensive in terms of disk space.
- In Git, a branch is merely a pointer to a snapshot (aka commit). This makes working with branches in git very “cheap”.

Branches are pointers



A repository after 3 commits and no new branches.

Git Branch

$ git branch

- List local branches

$ git branch <branch_name>

- Make a new branch with name branch_name

$ git branch -d <branch_name>

- Delete the <branch_name> branch

Git Checkout

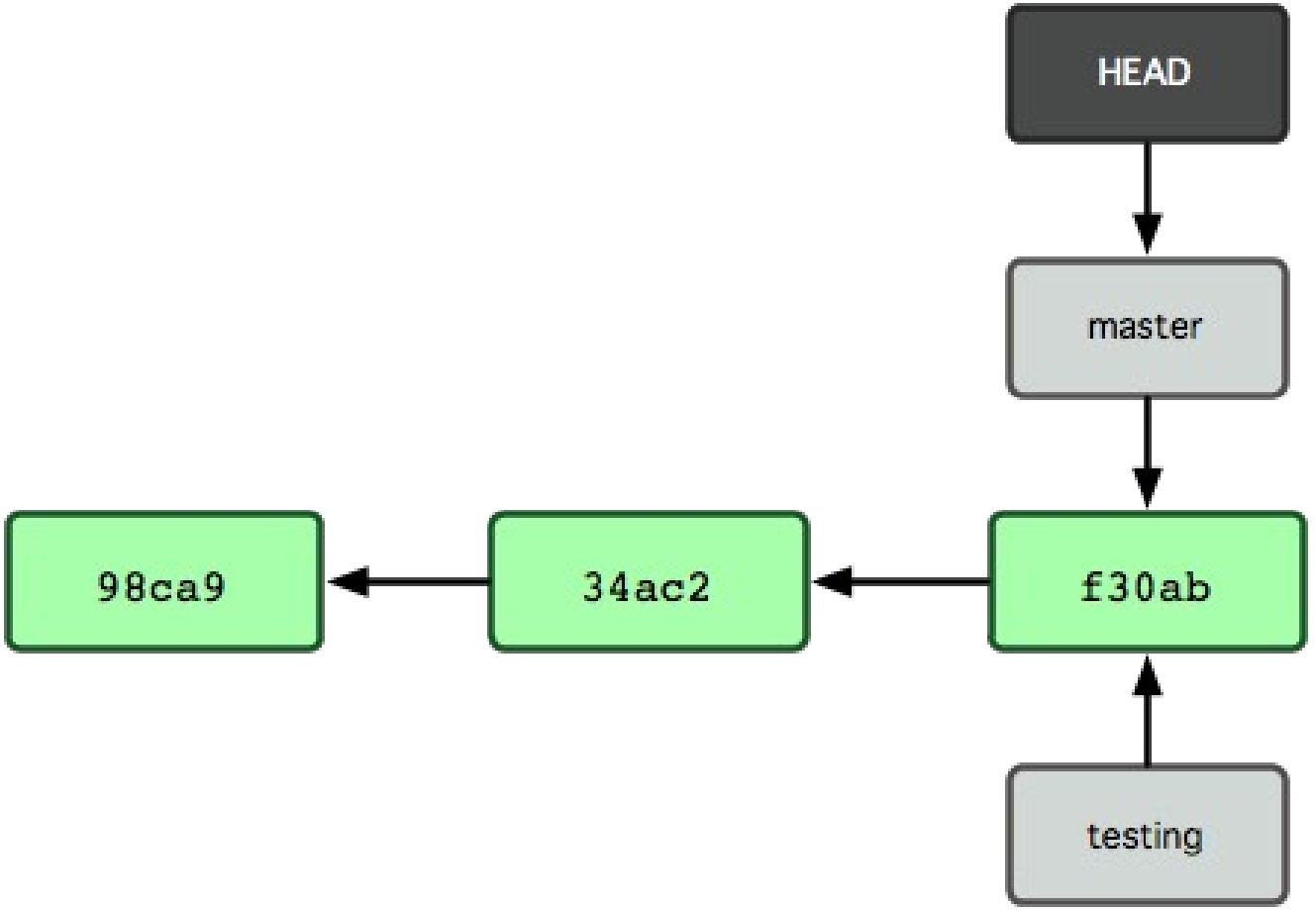
The git checkout command changes our current branch. When the branch is changed, the project directory is updated to the contents of that branch.

```
$ git branch testing
$ git checkout testing
```

Or you can use an equivalent shortcut:

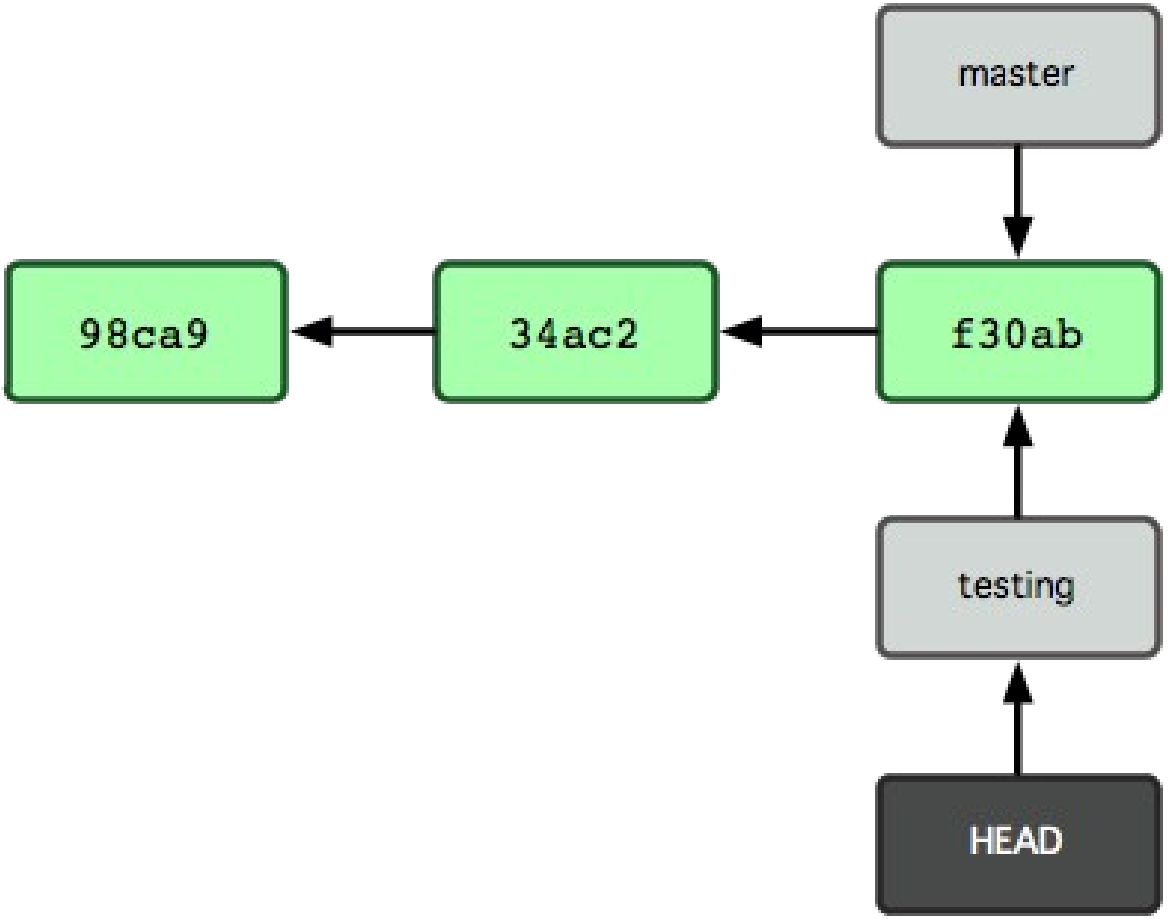
```
$ git checkout -b testing
```

Branches are pointers



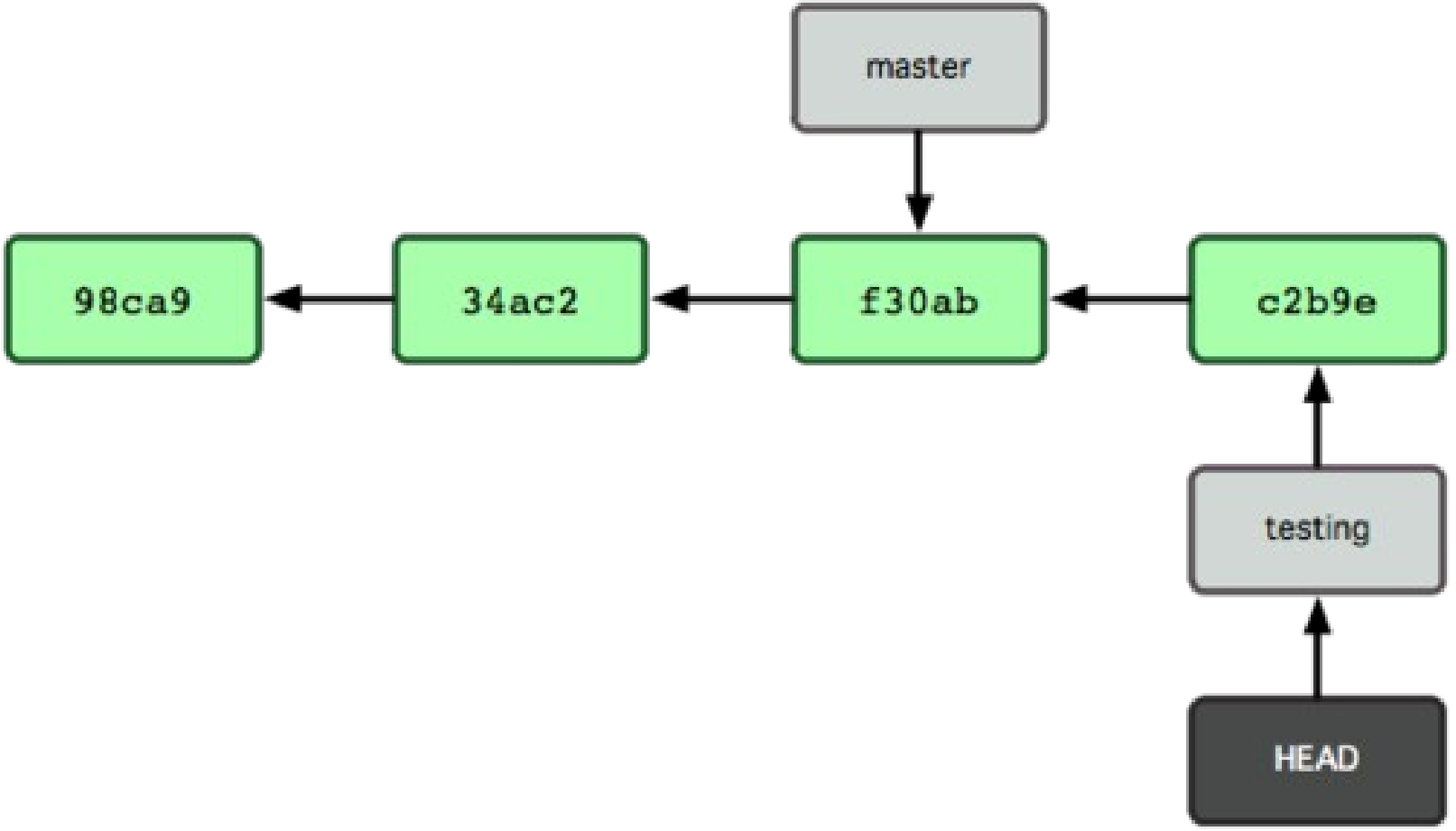
$ git branch testing

Branches are pointers



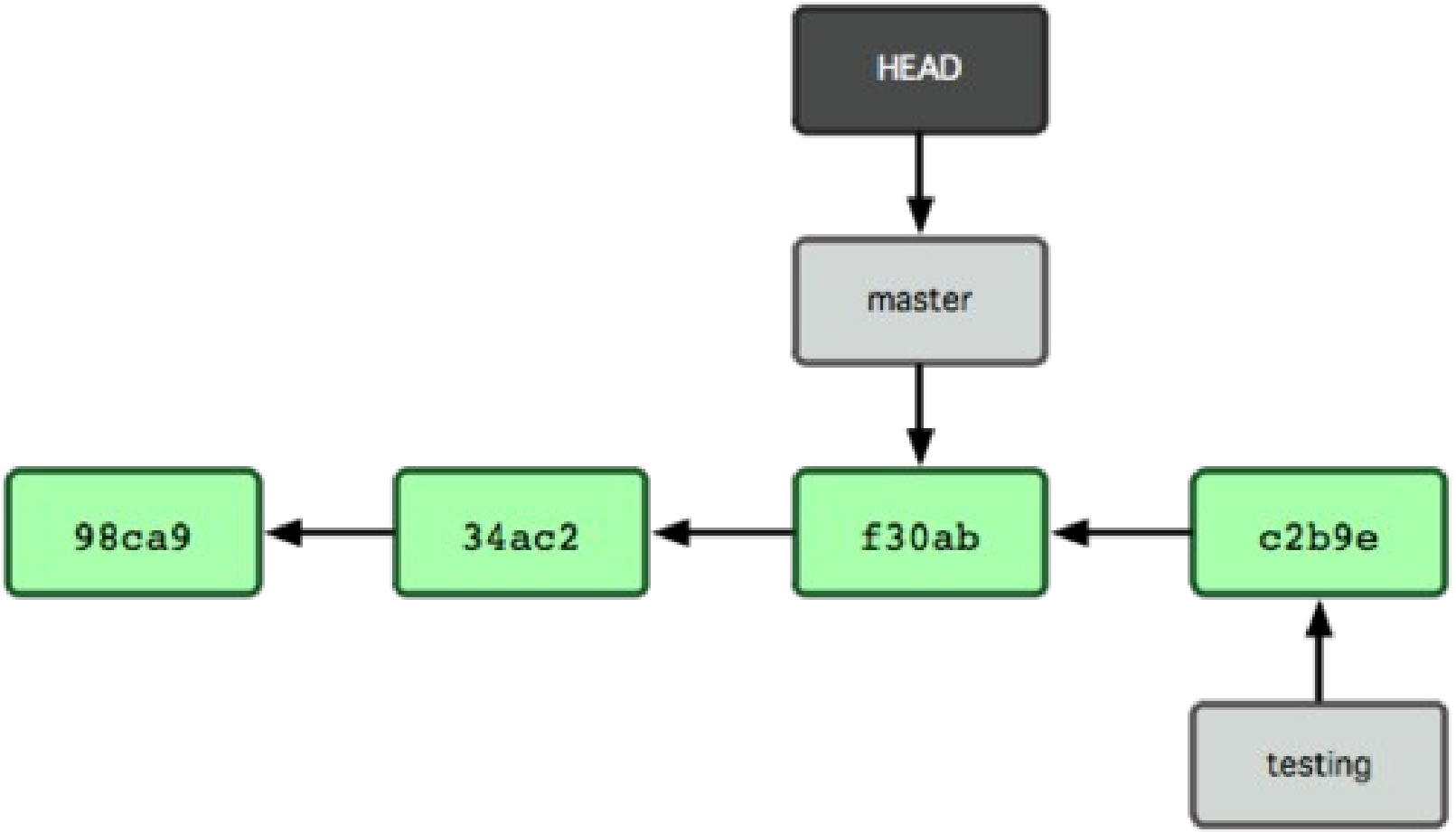
$ git checkout testing

Branches are pointers



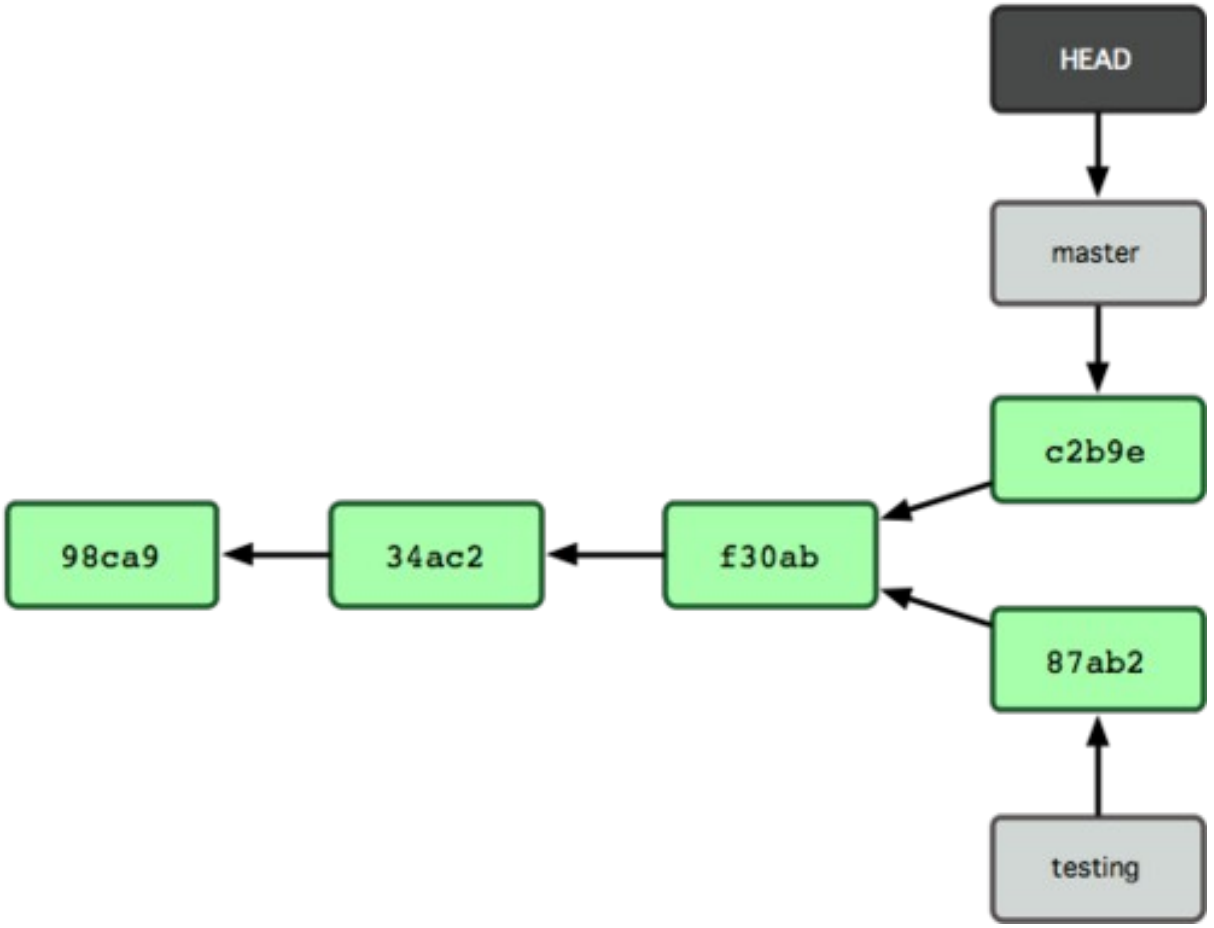
A new commit on the testing branch.

Branches are pointers



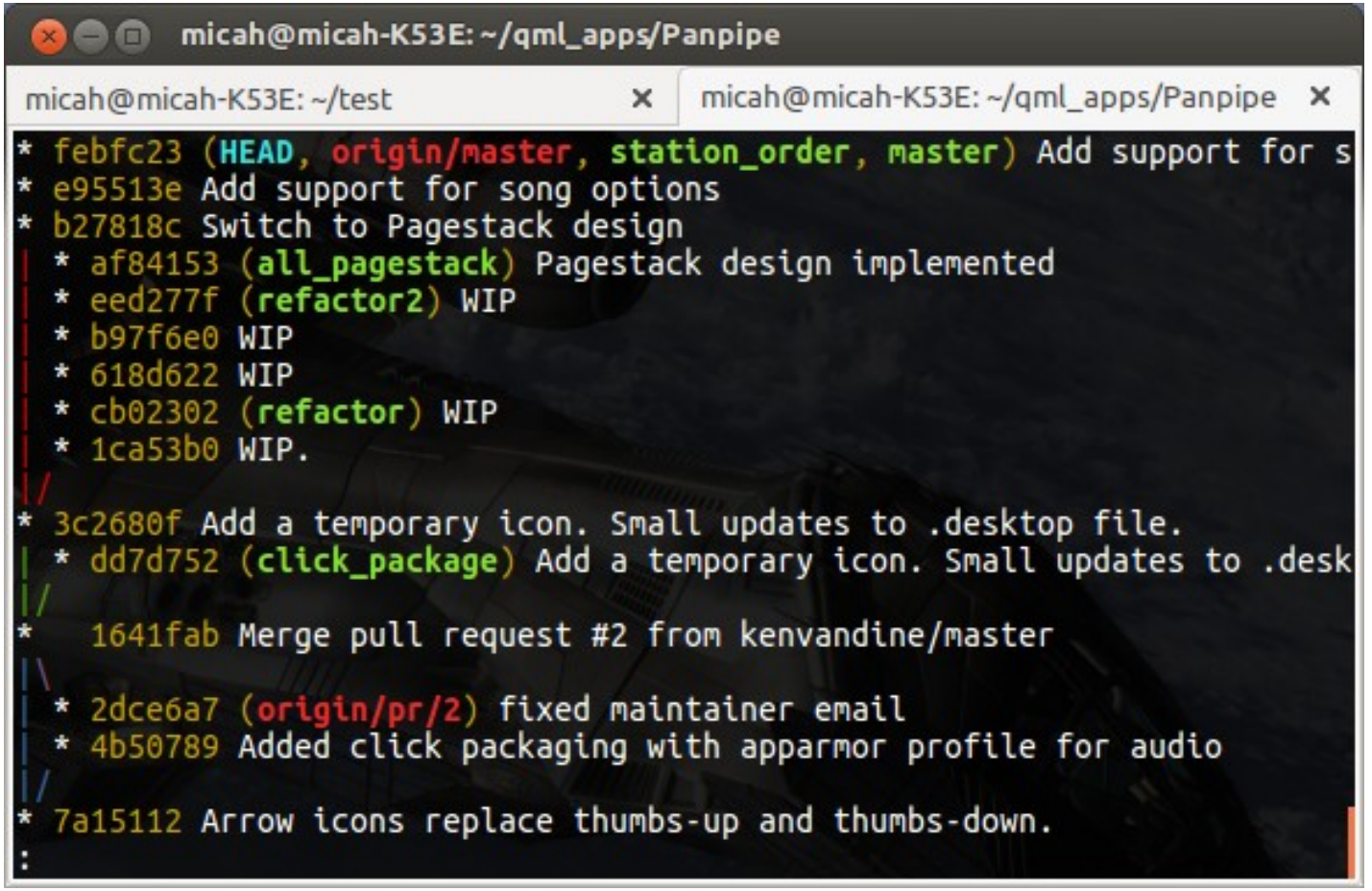
$ git checkout master

Branches are pointers



A new commit on the master branch.

Git Log EXTREME



The git log command has some tricks up its' sleeves.

Regular and Canadian LOLs

...just copy the following into ~/.gitconfig for your full color git lola action:

```
[alias]
	lol = log --graph --decorate --pretty=oneline --abbrev-commit
	lola = log --graph --decorate --pretty=oneline --abbrev-commit --all
[color]
	branch = auto
	diff = auto
	interactive = auto
	status = auto
```

From Conrad Parker's blog <http://blog.kfish.org/2010/04/git-lola.html>

Git Diff

Git can show you the difference between two commits or two branches.

```
$ git diff 3df73768  f3c3bfc
```

- Show the differences between the two commits

```
$ git diff master testing
```

- Show the differences between the two branches. Remember, branches are pointers to commits.

- The --stat flag shows a list of changed files rather than all the changes.

In Class Exercise Part 2

Git Merge

The git merge command merges in the named branch into the branch that you are currently on.

```
$ git merge <other_branch_name>
```

- Creates a new commit with the two branches combined.

```
$ git merge --squash <branch_name>
```

- Doesn't make a new commit with two ancestors. Instead all the merge differences are added to the index, and you can commit them when ready.

GIT MERGE
NO CONFLICTS
quickmeme.com

In Class Exercise Part 3

Remote Repositories

“Remote repositories are versions of your project that are hosted on the Internet or network somewhere.”

- Scott Chacon, Pro Git

Git Remote

$ git remote -v

- List all remote servers (-v means show remote url)

$ git remote add <local_name> <url>

- Add the repo at the given <url> as a remote server and call it by <local_name> in this repo.

Git Fetch

The git fetch command updates your repository's information about the remote repository. It will be easiest to observe the remote branches with the 'git lola' command.

$ git remote add <remote_name> <url>

$ git fetch <remote_name>

Git Push

The git push command puts/updates your local branch to a branch on the remote repository.

$ git push <remote_name> <local_branch_name>

- Add/update your <local_branch_name> on the remote server.

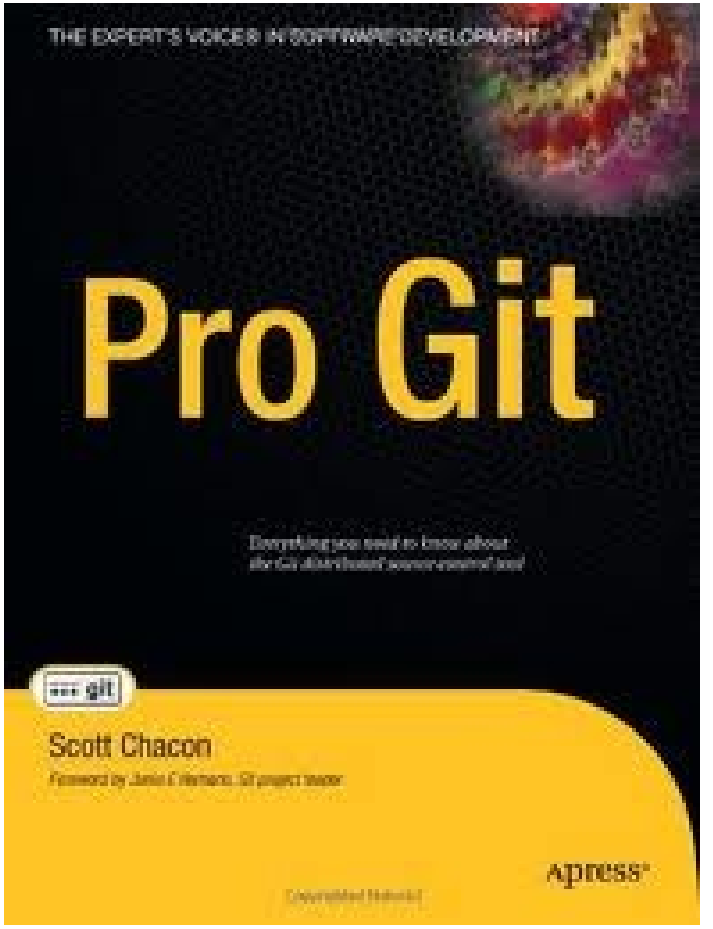
$ git push <remote_name> <local_branch_name>:<remote_branch_name>

- Same as above, but rename branch on remote

In Class Exercise Part 4

What You Need Now

- More practice
- Pro Git by Scott Chacon
  - $23 on Amazon
  - PDFs online for free



Git Questions?

Git Out Of Here

- I can linger for a while if you have further questions.
- Thank you for having me, it's a pleasure and honor to be here.
- Go into the world and use version control!

Git Out Of Here

- I can linger for a while if you have further questions.
- Thank you for having me, it's a pleasure and honor to be here.
- Go into the world and use version control!