Human-Computer Interaction Design

COGS120/CSE170 - "Intro. HCI"

Instructor: Philip Guo

Lab 1 - Version control and HTML (2018-10-03)

by Michael Bernstein, Scott Klemmer, Philip Guo, and Sean Kross

[Announce on Piazza] Before coming to lab 1:

1. Create a GitHub account (if you don't have one)
2. Download the Atom text editor for Windows/Mac https://atom.io/
3. Download Git for Windows/Mac https://git-scm.com/downloads (for Windows, choose to preserve Unix line endings as an option, and also make sure you can run "Git Bash" after you install)

Wi-fi may be slow in the classroom, so if you want to follow along please download and try to install everything beforehand. It's OK if you want to use your own text editor and not Atom.

Why does this class have a lab?

This is not a "learn to code" class. For this class, coding is a means to an end (i.e., implementing your ***design ideas***). Thus, we won't go deep into the computer science concepts behind any technologies. These labs are just a starting point. You'll likely need to learn more on your own.

Since this is not a coding class, you will *not* turn in labs for credit.

However, concepts from labs will be tested on Exam 1 and Exam 2.

These labs will help you make a personal portfolio web site

- **Never lose code**: version control
- **Get content on a page**: static HTML and CSS
- **Update that content dynamically**: JavaScript
- Later: analyze its use, make it mobile, make it look good

Approximate format for labs:

1. I'll give a mini-lecture on the slides for 30 minutes. You can follow along if you want or jump ahead at your own pace. [room will be mostly silent so that everyone can hear clearly]

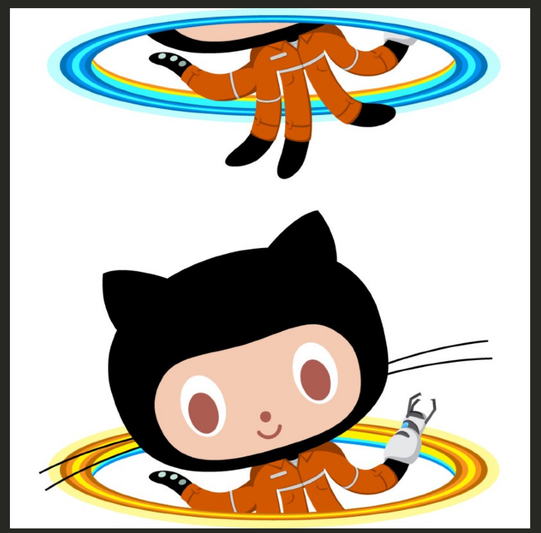
2. Then we will open it up for free-form lab work with TAs walking around to help. You can also come up to the podium to ask me for help too. [room will be louder]

Lab 1: Why use version control

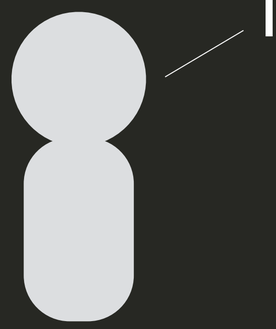
- Make checkpoints on your computer: git
- Back up your code safely in the cloud: GitHub
- Work together with your teammates without overwriting each other's code edits

Lab 1: Version control

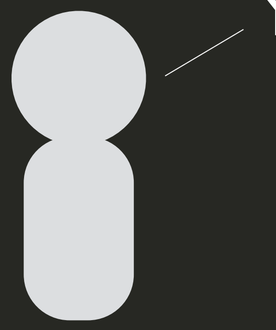
- We will be learning to use the Git version control system.



Why learn it? Because it's by far the most popular system in use today, especially in free and open source code. The GitHub website is a convenient, free, and popular place to host your Git code repositories.



- Feel free to run ahead and aim for the lab stretch goal:
  - Turn the portfolio page scaffold into a fully functional web site with content from your previous projects. Must include images, descriptions, and multiple .html files.
- Even better: volunteer to help your classmates. Please be patient, though, since people come from all sorts of prior programming backgrounds.



- All speeds are welcome here. We really want everyone to get up to speed. Lab time (and office hours) are for you!

- Try first: ask your friends, help your friends
- Try second: raise your hand to summon a TA in lab
- Try third: come to our office hours for more help

Installing Git on your own computer

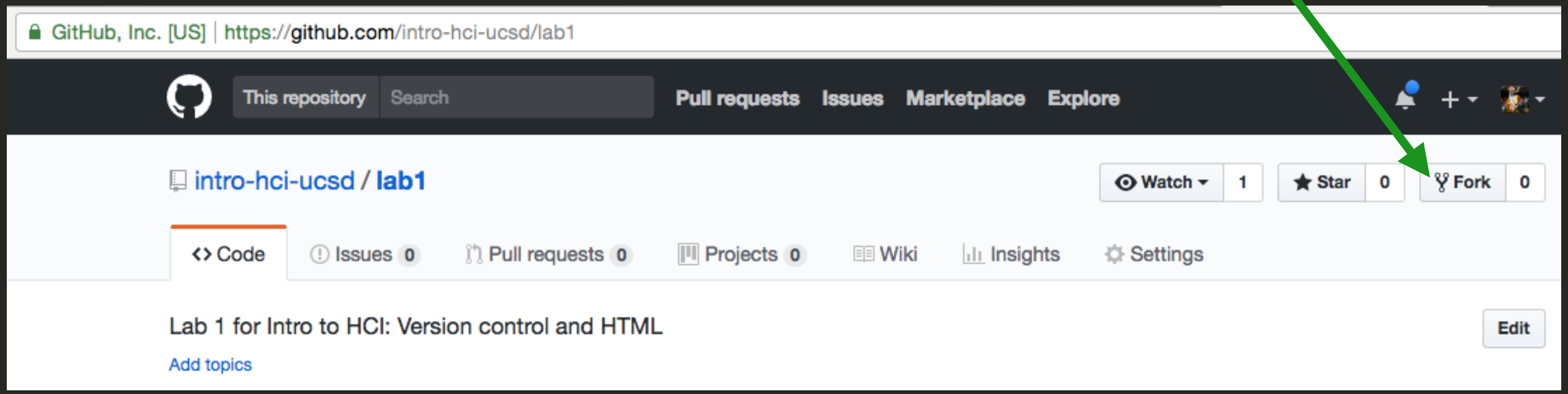
- On Windows, make sure to install Git so that it preserves Unix style line endings. Also make sure you can run "Git Bash" after you install it. Find it in Start Menu or desktop.
- On Mac OS, you may be prompted to install the Xcode development environment first, which can be annoying. Bypass that if you don't want to download gigabytes of Xcode package to use Git. TAs can help. Google online for "git mac os x without xcode" to see resources such as:

- http://blog.bobbyallen.me/2014/03/07/how-to-install-git-

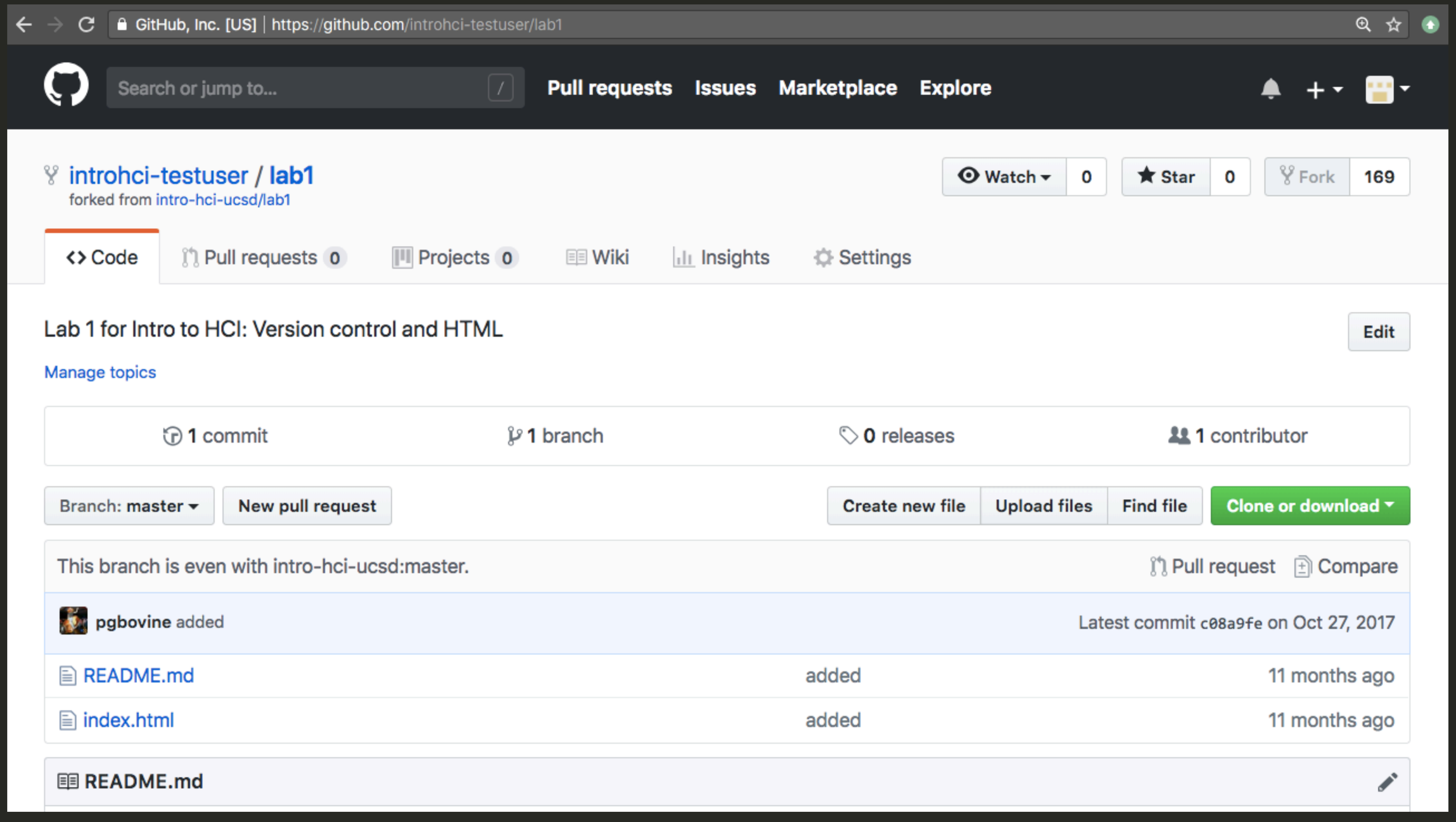
Demo: Git & GitHub

Fork your first repository

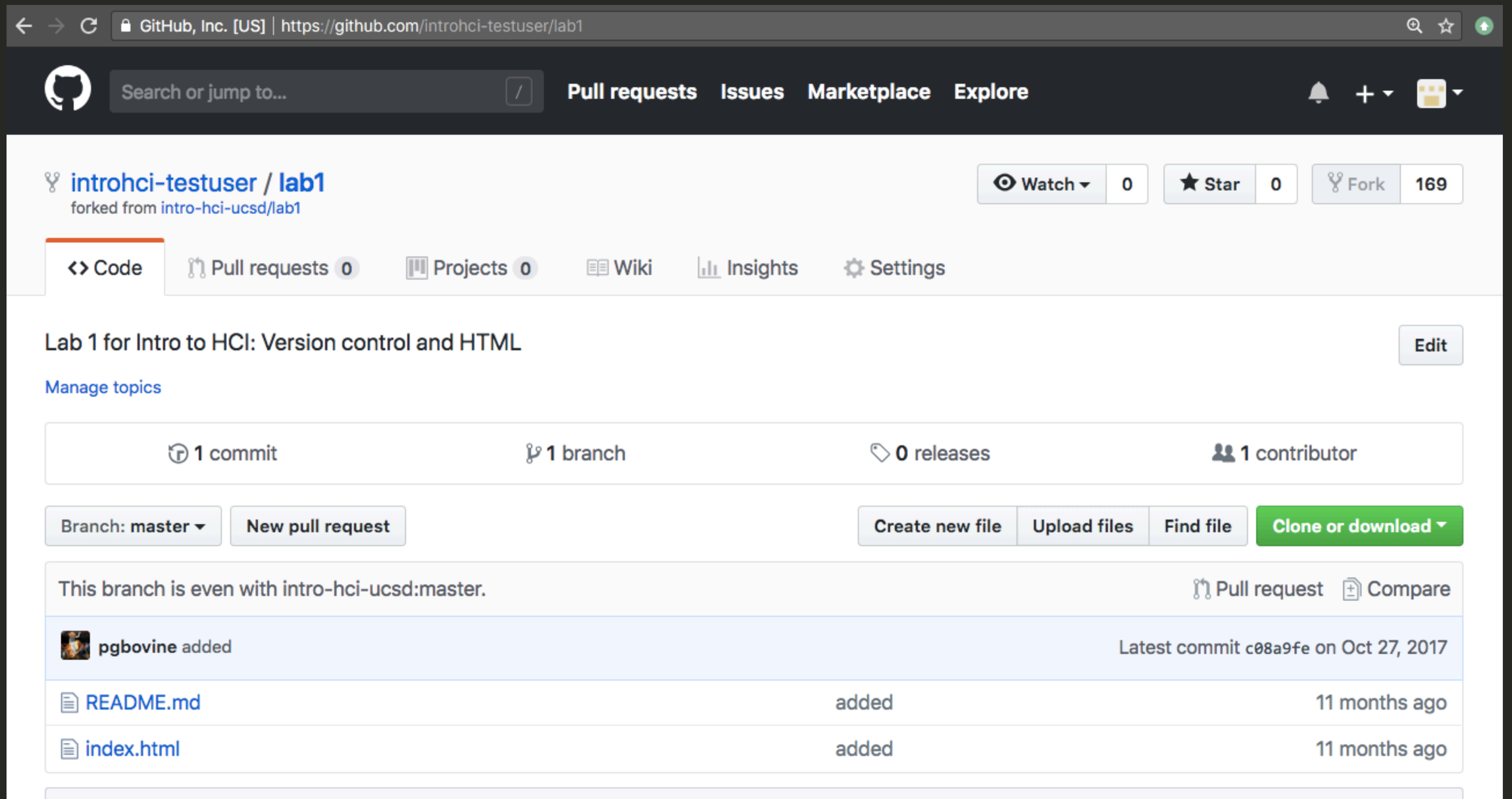
- Find Lab 1 at: https://github.com/intro-hci-ucsd/lab1
- Forking will ***copy*** the repository to your GitHub account



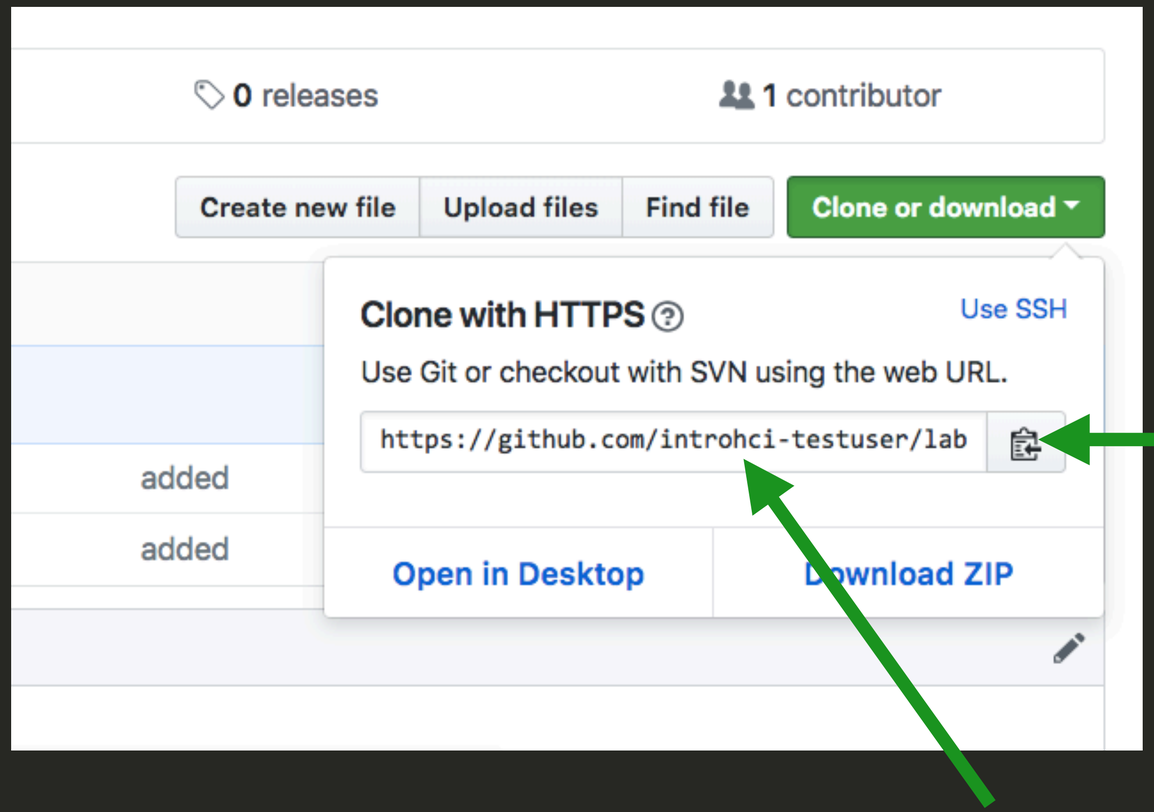
Fork (copy) your first repository



Why create a fork (copy)? Because you have full control over this repository so that you can later modify its contents. (You *don't* have write permissions to intro-hci-ucsd repository.)



Copy your own git URL to the clipboard



Use https. Your Clone URL should look like this:
https://github.com/<YOUR GitHub USERNAME>/lab1.git

not this … https://github.com/introhci-testuser/lab1.git
(since this is my username) Leave this browser tab open!

Open your terminal

- Mac OS X
  - Applications → Utilities → Terminal
  - Or search "Terminal" in Spotlight
- Windows: Git Bash
  - Start Menu → Git → Git Bash
- Create a directory named "introHCI"
  - Using the command-line: `mkdir introHCI`
- Change your directory to introHCI in the terminal:
  - `cd introHCI`
- Typing the `pwd` command should show `introHCI`

`git clone` the repo inside of introHCI directory

("repo" = Git repository)

- Your current directory should be introHCI (type `pwd` in the terminal to double-check). Then type `git clone`
- … add a space, then paste the URL you just copied. Make sure that URL has your own username in it, not my username

```
pgbovine@Philips-MacBook-Air ~/Desktop/introHCI
$ git clone https://github.com/introhci-testuser/lab1.git
Cloning into 'lab1'...
remote: Enumerating objects: 4, done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 4
Unpacking objects: 100% (4/4), done.
```

The parts of the command-line interface in your terminal application:

username:~/Desktop/introHCI$ cd lab1

- Important terminal commands
  - `cd directoryname`: change directory to `directoryname`
  - `ls`: list all files and directories in the current directory
  - `pwd`: which directory am I in?

Windows: how do I paste into Git Bash?

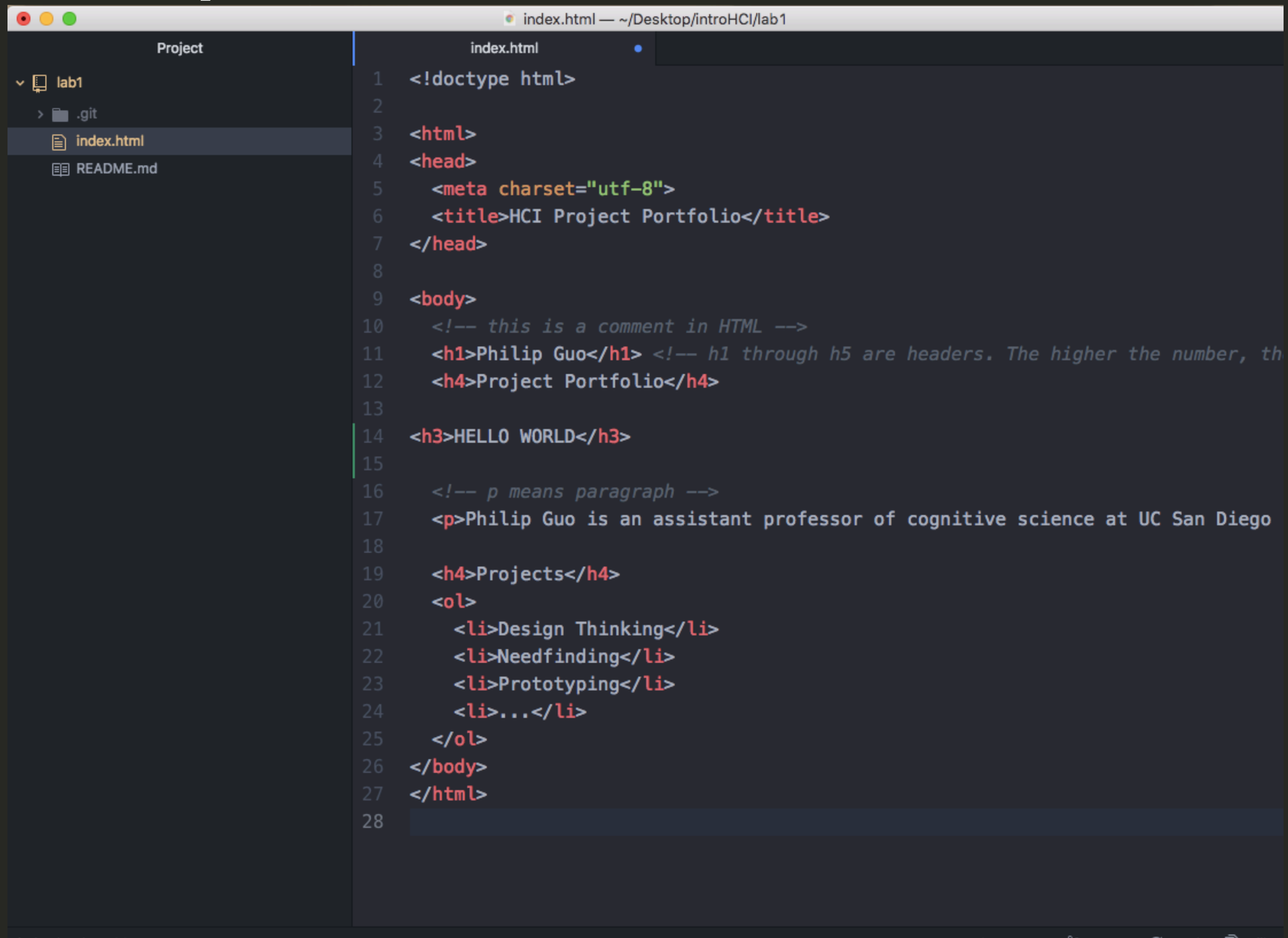
- Click on the terminal icon in the upper left of the window: Edit, Paste. For a shortcut: Alt-spacebar, then e, then p. If that seems needlessly complicated, you can also enable QuickEdit mode by clicking on the terminal icon in the upper left of the window: Defaults(or Properties). Go to the Options tab and enable "QuickEdit Mode". Now you may paste by right-clicking in the terminal.

Open your text editor

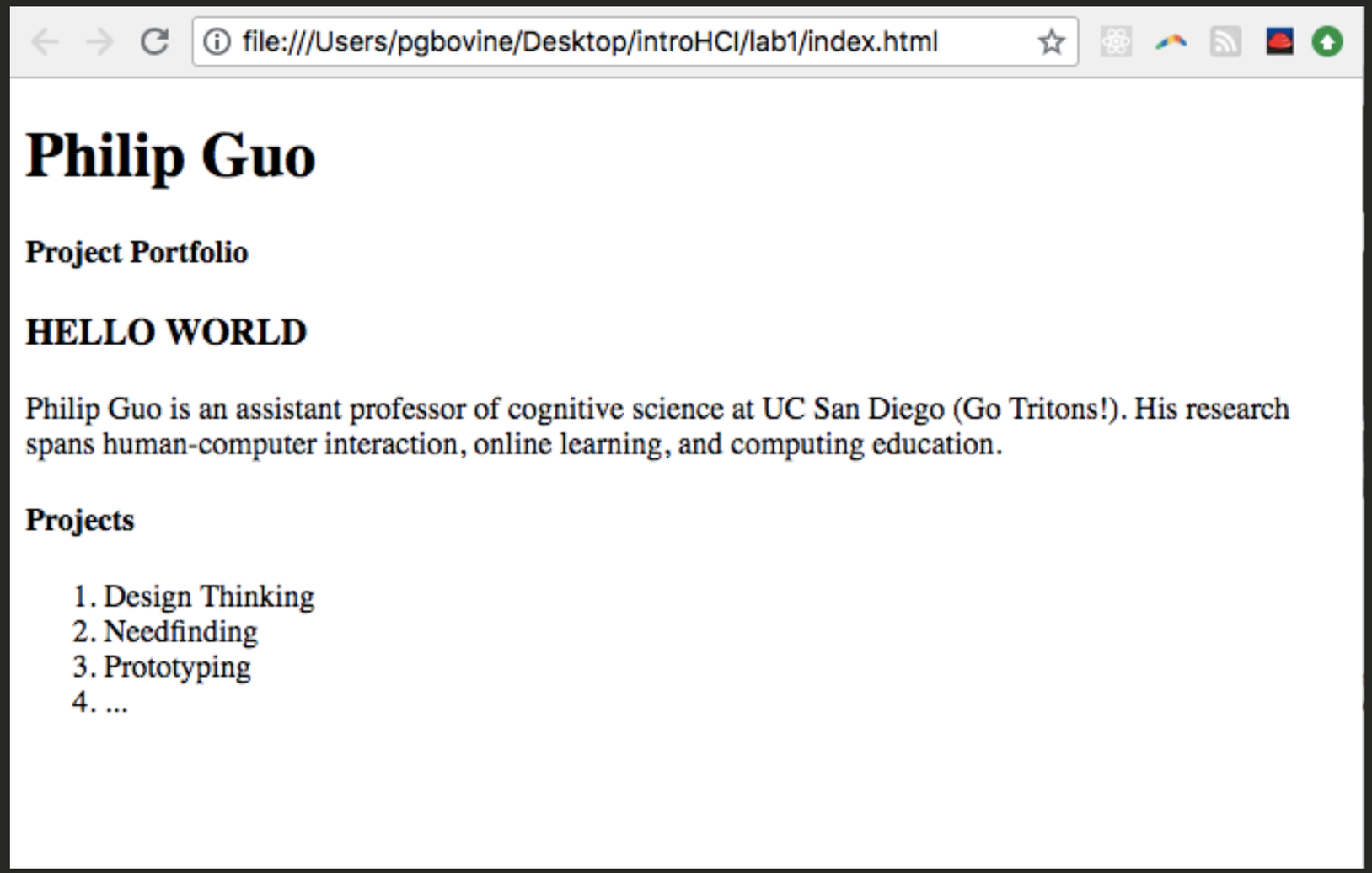
- Mac OS X
  - Applications → Atom
  - Or search “Atom” in Spotlight
- Windows
  - Start Menu → Atom

- [You can also use other text editors if you’d like.]

Add your info into lab1/index.html



Browse to index.html using the file on your hard drive, open it from browser



Lab 1 stretch goal reminder

- Turn the portfolio page scaffold into a full web site and push it to GitHub
- It must contain:
  - Content from your previous projects
  - Images and descriptions
  - Multiple .html files linked together via hyperlinks
- Even better: add styling

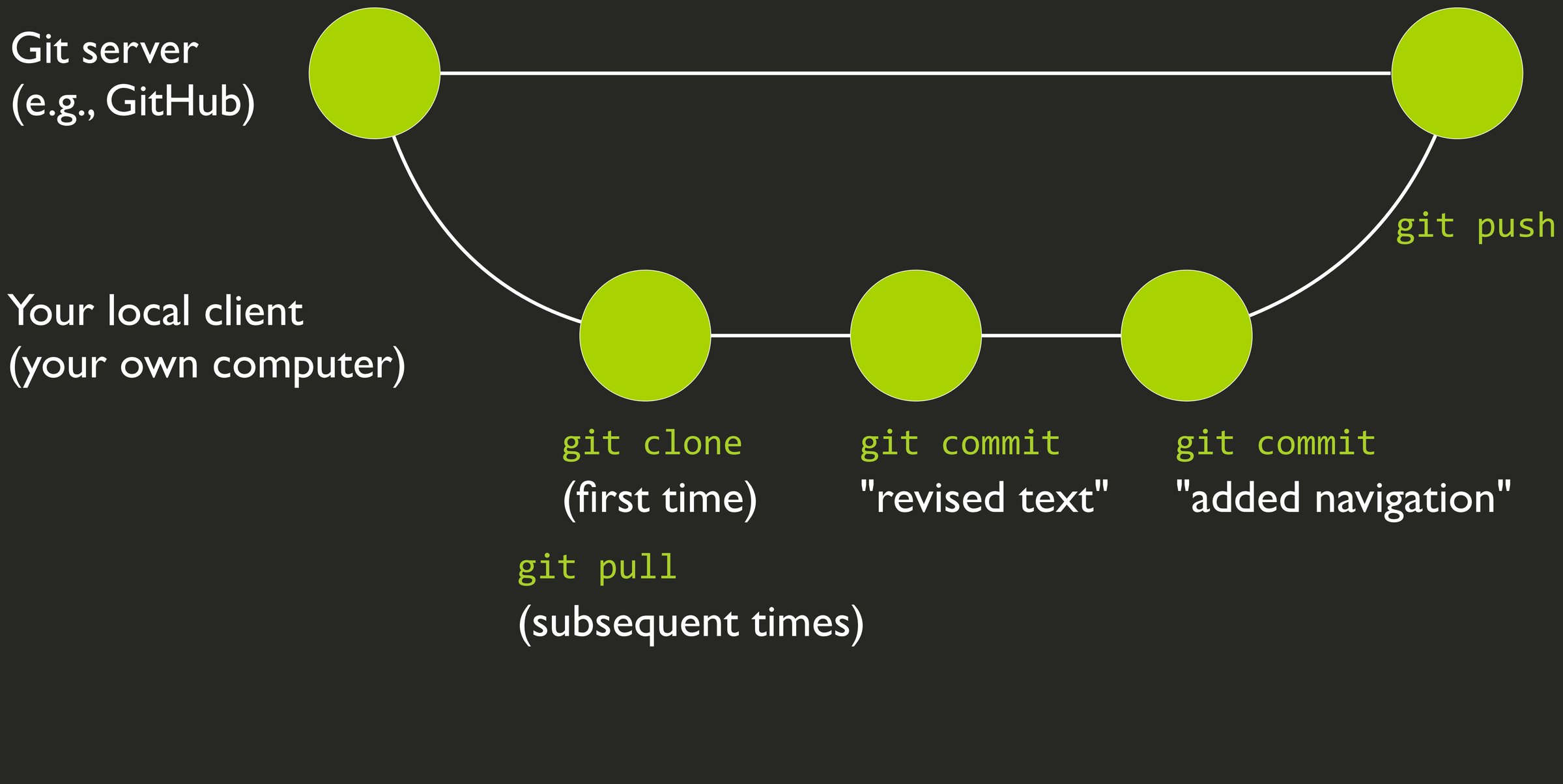
Set up Git so we can commit with sensible messages. Type in these commands in your terminal …

```
git config --global user.name <YOUR NAME IN QUOTES>
git config --global user.email <YOUR EMAIL IN QUOTES>
git config --global push.default simple
```

For example, for Michael …

```
➜  lab1 git:(master) git config --global user.name "Michael Bernstein"
➜  lab1 git:(master) git config --global user.email "msb@cs.stanford.edu"
➜  lab1 git:(master) git config --global push.default simple
➜  lab1 git:(master)
```

Basic single-user version control with Git



`git diff` tells you how your files have changed

```
pgbovine@philips-air ~/Desktop/introHCI/lab1
$ git diff
diff --git a/index.html b/index.html
index a4a9b7b..ad0d9c1 100755
--- a/index.html
+++ b/index.html
@@ -11,6 +11,8 @@
        <h1>Philip Guo</h1> <!-- h1 through h5 are headers. The higher the numbe
r, the smaller the header -->
        <h4>Project Portfolio</h4>

+<h3>HELLO WORLD</h3>
+
        <!-- p means paragraph -->
        <p>Philip Guo is an assistant professor of cognitive science at UC San D
iego (Go Tritons!). His research spans human-computer interaction, online learni
ng, and computing education.</p>
```

`git status` and `git add`

- `git status` tells you which files would get committed
- `git add <filename>` tell git that a file should be included in the commit

```
pgbovine@philips-air ~/Desktop/introHCI/lab1
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

pgbovine@philips-air ~/Desktop/introHCI/lab1
$ git add index.html

pgbovine@philips-air ~/Desktop/introHCI/lab1
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html
```

Commit then push

```
pgbovine@philips-air ~/Desktop/introHCI/lab1
$ git commit -m "made my first change"
[master 6c38b73] made my first change
 1 file changed, 2 insertions(+)
```

- `git commit -m "Commit message"` performs a local commit of all files that you ran `git add` on earlier. Do this often: it's a save point. This is not backed up to the server yet.
- `git push` pushes (backs up) all local commits to the git server
- `git pull` pulls (brings down) in any new changes from the server (useful when *multiple people* share the same repository)

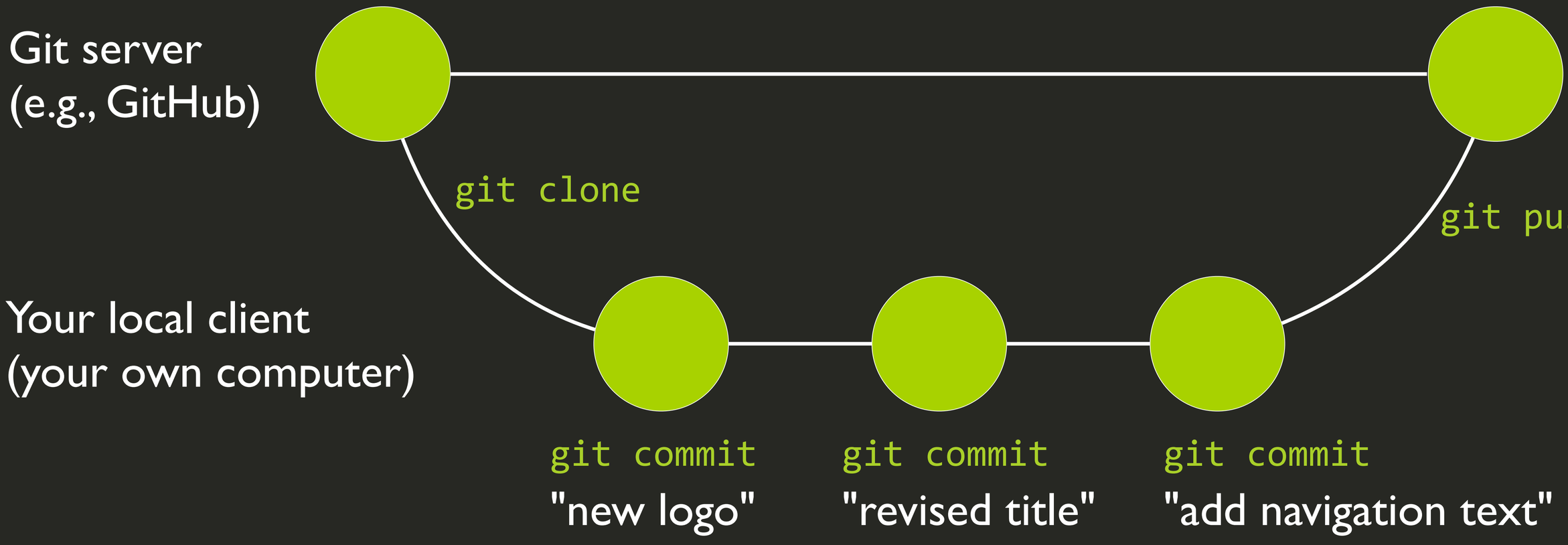
Commit then push

```
pgbovine@Philips-MacBook-Air ~/Desktop/introHCI/lab1
$ git push
Username for 'https://github.com': introhci-testuser
Password for 'https://introhci-testuser@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 336 bytes | 336.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/introhci-testuser/lab1.git
   c08a9fe..c1190d6  master -> master
```

Other useful basic git commands

- `git commit -am "Commit message"` performs a local commit of all files that have changed since the last commit (instead of tediously running `git add` on each individual file)
- `git log` shows your history of commits

Recap so far: single-user version control

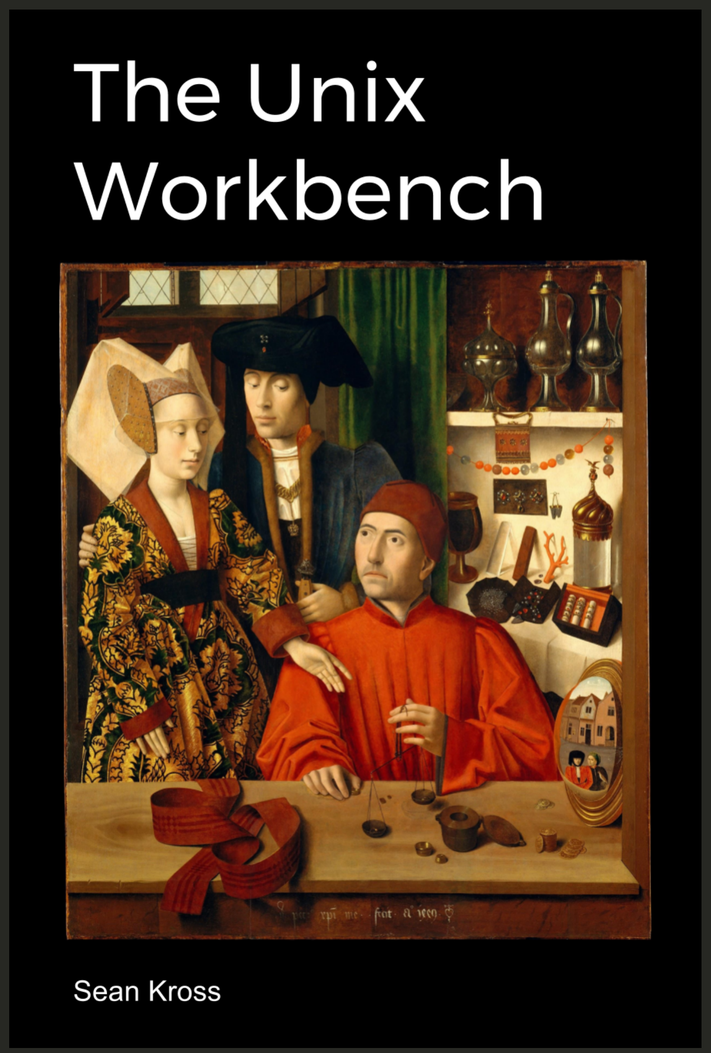


Question: What's the difference between Git and GitHub?

Question 2: Can you use Git without GitHub? Why would you?

EVERYTHING AFTER THIS IS OPTIONAL [not on exams]

To learn more, read The Unix Workbench



Most importantly the chapters
“Command Line Basics” and
“Git and GitHub”

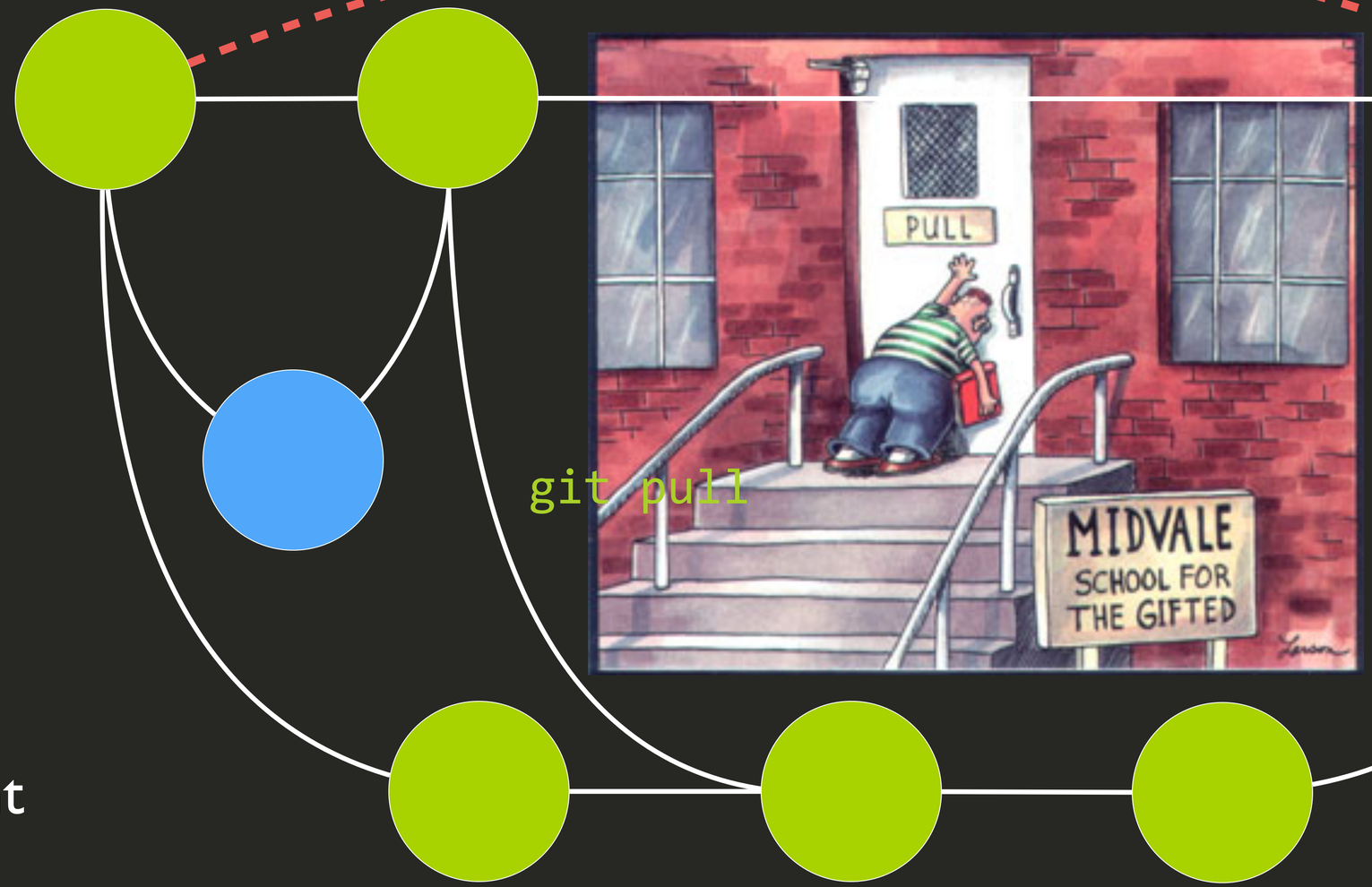
http://seankross.com/the-unix-workbench

Multi-person version control

Git server

Your partner

Your local client




```
git pull
```

Merging and conflicts

- Please wait while we add a conflicting change to the upstream repository

- [we will manually change the contents of index.html so that it hopefully conflicts with yours]

Merging and conflicts

- Add the upstream repository as a remote server and merge it in to discover the conflict [this isn't the most realistic scenario, but it's a quick way to show conflicts]
- git remote add upstream https://github.com/pgbovine/lab1.git
- git pull upstream master

```
➜  lab1 git:(master) git remote add upstream https://github.com/IntroHCI/lab1.git
➜  lab1 git:(master) git pull upstream master
```

Resolving merge conflicts

```
<!doctype html>

<html>
<head>
    <meta charset="utf-8">
    <title>HCI Project Portfolio</title>
</head>

<body>
    <!-- this is a comment in HTML -->
<<<<<<< HEAD
    <h1>Scott Klemmer</h1> <!-- h1 through h5 are headers. The higher the number,
    the smaller the header -->
    <h4>Project Portfolio</h4>

    <!-- p means paragraph -->
    <p>Scott is an associate professor of Cognitive Science and Computer Science &
    Engineering at UC San Diego.</p>
=======
    <h1>John Hennessy</h1> <!-- h1 through h5 are headers. The higher the number,
    the smaller the header -->
    <h4>Project Portfolio</h4>

    <!-- p means paragraph -->
    <p>John Hennessy is president of Stanford University. He takes HCI classes in
    his spare time.</p>
>>>>>>> c17fc57096f3daa4ed0566aab987670de6a874b6

    <h4>Projects</h4>
    <ol>
        <li>Waiting in line</li>
```

Line 11, Column 14; Undo: Insert Characters Tab Size: 4 HTML

Resolving merge conflicts

- Push the merged version back to the repository with `git add`, `git commit`, `git pull` and `git push`

```
➜  lab1 git:(master) ✗ git add static/index.html
➜  lab1 git:(master) ✗ git commit -m "fixing merge conflict"
[master 1180cca] fixing merge conflict
➜  lab1 git:(master) git pull
Already up-to-date.
➜  lab1 git:(master) git push
```

Parting advice: avoid merge conflicts by coordinating with your teammates.

- It's OK to both work on the same file at once, but make sure you're not working "near" each other in the same file. Coordinate manually.

- The advantage of something like Google Docs is that you get real-time sync and multiple cursors so that you won't have merge conflicts.