Two things

• How to write code for web apps.
• How to collaborate and keep track of your work.
A text editor
A text editor
A text editor

Anything that you’re used to using

Even notepad works, but you’ll probably want something a little more.

Most of the staff uses Sublime Text.
A text editor

Vim or Emacs? You probably know what you’re doing.

The rest of you … learning the basics of vim or emacs might be useful
TERMINAL

It’s a bit difficult at first, but learn to use it:

- navigating directories
- starting servers
- Git

Some of our tools will require the use of the terminal.
GIT

a primer in distributed version control
Why Git?

• Want to keep track of your changes
  • file-v1.html, file-v2.html, ...
Why Git?

- Want to keep track of your changes
  - file-v1.html, file-v2.html, ...
- Collaborate with others -- don’t want to overwrite each other’s changes
Why Git?

- Want to keep track of your changes
  - file-v1.html, file-v2.html, ...
- Collaborate with others -- don’t want to overwrite each other’s changes
- Keep your source code in some central repository
Why Git?

• Want to keep track of your changes
  • file-v1.html, file-v2.html, ...

• Collaborate with others -- don’t want to overwrite each other’s changes

• Keep your source code in some central repository

• Easily browse history and debug.
Why Git?

Because we require you to use it in 6.148 :)

... and most companies use it (or something similar)!
Install Git

- **Linux**: `sudo apt-get install git-core`
- **Mac/Windows**: use the installer on the Git website
- We’ll be using the command-line interface (Git Bash for Windows).
Basics

• Git keeps track of EVERYTHING

• Even if you delete a file, it’s likely you can revert it.
Basics

• Git keeps track of EVERYTHING

• Even if you delete a file, it’s likely you can revert it.

• A few dangerous commands -- don’t use them unless you know what you’re doing!
  • git reset --hard
How to start

• `git clone <repo URL>`
  • Takes a repository and “clones” it into your machine.

• After you make some changes:
  • `git add <filename>` for every file, or `git add .` for the entire directory
  • `git commit -m “commit message”`
Try it now!

• git clone https://github.com/kongming92/6148demo.git

• Go into the directory: cd 6148demo

• Make some changes, add and commit them

• What happens if you run the command `git log`?
Collaboration?

• The staff will give you Github repos after Milestone 0

• Git = the software tool you use to manage your changes

• Github = a website that hosts a “central” copy of all your code
Using Github

- After you make some changes, do `git push`, and your changes will go to Github.
- Everyone on your team can then browse the code.
- To get the changes that your teammates made, do `git pull`. 
Sometimes when you pull, you will get things that look like garbage.

This is a conflict -- Git could not figure out how to combine two different changes.

Resolve conflicts manually, then git add / git commit.
Branches

the most powerful feature of Git, once you learn to use it correctly
Branches

• Think of your source code’s history as a tree of commits:
  • New features are created, combined into main product
  • Each time you want to try something new, create a branch!
Branches

- `git branch` -- show all branches
- Default branch: master, usually used for “stable” code
- `git branch <name>` -- create a new branch
- `git checkout <name>` -- switch branches
Branches

• So what happens when you’re done with a branch?

• Merge the branch into something else:
  • git checkout master
  • git merge featurebranch

• Conflicts, just like with pulling
Your turn, pt. 2!

• Using the same example repo you cloned earlier:
  • Create a branch, and commit some changes.
  • Go back to master.
  • Create a different branch and commit some more changes
  • Merge both back into master
  • Can you trigger a merge conflict?
A few final notes

• If you have an MIT certificate: go to github.mit.edu.

• Do it now. This is necessary for Milestone 0.

• You will need to set up SSH keys! Your Athena username/password will not work! Follow the instructions here: https://help.github.com/articles/generating-ssh-keys/
More on Git...


• Collaborative workflows: https://www.atlassian.com/git/tutorials/comparing-workflows/centralized-workflow

• Lots of *amazing* features, none that you absolutely need, but are cool anyways