Administrivia

▪ HW 3 posted later today -> Extra credit due date Wednesday Nov 18th @ 9pm
▪ HW 4 out Monday Nov 16\textsuperscript{th} -> Extra credit due date Wednesday Nov 25\textsuperscript{th} @ 9pm
▪ HW 5 out Monday November 30\textsuperscript{th} -> Extra credit due date Wednesday Dec 5\textsuperscript{th} @ 9pm
▪ HW 6 out Monday Dec 7\textsuperscript{th}, Due end of quarter
▪ Review Assignment #2 out Friday Nov 13\textsuperscript{th}, Due Friday Nov 20\textsuperscript{th} @ 9pm
▪ Review Assignment #3 out Wednesday Dec 9\textsuperscript{th}, Due end of quarter
▪ End of quarter due date Wednesday December 16\textsuperscript{th} @ 9pm
Does your file system look like this?
Issues faced

- Multiple copies of the same files
- Changes leading to scripts being broken with no way to go back
- Accidental deletion of files
- Lost computer leads to loss of files
- Collaboration on a project can be difficult if each collaborator has their own copy
Version Control to the rescue!

- Version control is software that keeps track of changes to a set of files

- Version control enables multiple people to simultaneously work on a single project.

- Places we see version control:
  - Microsoft Word allows us to undo our most recent changes (go back to a previous version)
  - In Google Docs you can see the version history of files, see who made these changes and revert back
  - You (hopefully) back up your computer to a cloud storage like Google Drive or Dropbox
  - Gradescope also allows you to look at your previous submission
Version Control in Gradescope
Decentralized vs centralized storage
Intro to Git

- Git is a version control system optimized for text-based files
- Git ≠ GitHub
- "origin" copy of the repo is stored on a Git server
  - The remote repository is the defacto central repository
  - Remote repositories are hosted on services like GitHub, Gitlab, or Bitbucket
- Everyone shares changes by pushing their changes and pulling others' changes
Repository

- A repository, commonly referred to as a repo, is a location where a project lives.
- Each collaborator has a copy of a repository for the project which they sync with the central copy.

What should be inside of a repository?
- Source code files (i.e. .c files, .java files, etc)
- Build files (Makefiles, build.xml)
- Images, general resources files

What should not be inside of a repository (generally)
- Object files (i.e. .class files, .o files)
- Executables
- Huge media files (e.g. videos)
- Any application credentials
- System files (e.g. .DS_Store in Mac)
A repository's history is a series of "commits"

- Each commit makes changes to the files in the repo

- Commit history serves as a log of the changes everyone made

- Commits are easy, and free! Commit early and often.
  - Ever accidentally deleted something and forgot what it was?
  - If you make a mistake, you can look at the changes since the last commit
What is a “commit”?

- A "commit" is a single set of changes made to your repository

- Essentially a version of the project you want to save

- Also records:
  - The name of the author
  - The date and time
  - A "commit message": short sentence describing what that commit did

- Identified by an ID, or "SHA"
Phases of Git

- **Working Directory**: Working changes
- **Staging Area/Index**: Changes you’re preparing to commit
- **Local Repository**: Local copy of the repository with your committed changes
- **Remote Repository**: Remote shared repository (Usually stored with a platform like GitHub/Gitlab)
Phases of Git

- Working Directory: Get added to
- Staging Area/Index: Are committed to
- Local Repository: Are pushed to
- Remote Repository:
“git” command

- The "git" command is the primary way of interacting with git
- You must "cd" into the folder where your repo is stored, or any subfolder within it
- Used like any other shell command!
- There is a GUI available for most companies (e.g. GitHub Desktop)
- Must be installed on your computer (already installed on klaatu)
Some “git” commands

- **git init**
  - Create a new empty git repo or convert an existing folder to a git repo

- **git add**
  - Preparing edited files to be saved (committed) to a repo

- **git commit**
  - Records (saves) changes to a repo
  - Accompanied by a short descriptive message

- **git push**
  - Update the remote copy of the repo with the local changes and commits
Phases of Git

- **Working Directory**: `git add ...
- **Staging Area/Index**: `git commit`
- **Local Repository**:
- **Remote Repository**: `git push`

**NOTE**: There are way more `git` commands than what is listed here - this is a simplified model to get us started.
Staging and committing overview
Staging and committing overview

```bash
git init
```
Staging and committing overview
Staging and committing overview

Working changes

```
git init
```

```
git repo
```
Staging and committing overview

Working changes

Staging area

```
git add .
git init
git repo
```
Staging and committing overview

Working changes → Staging area

- git add .
- git commit -m "message"

git init

git repo
Inspecting a repository

- **git status**
  - Lists the files which you have changed but not yet committed
    - Working directory
    - Staging area
  - Indicates how many commits have made but not yet pushed

- **git log**
  - Shows the commit history
  - Option to see last $n$ commits by adding $-<n>$ flag
  - Press "q" to exit
Working with git locally
mkdir test_git
cd test_git
git status

git init

touch file.txt

git status

git add file.txt

git status

git commit -m "Add first file"

git status
Working with remote
git commands for interaction with remote

- **git clone**
  - Cloning is the process of creating a working copy of the remote or local repository by passing the following command.
  - `git clone username@git_server_hostname:/path_of_repository`

- **git pull**
  - If we have already cloned the repository and need to update local (only code) respect to the remote server
  - `git pull origin main`

- **git fetch**
  - Fetching is the process of updating (only git information) the local git structure and information from remote repository
  - `git fetch`
Updating changes using pull

NOTE: There are way more git commands than what is listed here - this is a simplified model to get us started.
What’s next!

▪ Branching, checkout

▪ Merging, Merge (Pull) Requests

▪ Conflicts

▪ Interacting with a Git Server (GitHub / GitLab)
Complete working (including fork)
Useful resources

- **Try Git (resources and tutorial)**
- **The Git Cheat Sheet**
- **Stack Overflow’s definitive guide for beginners**

- When you are terribly stuck with git
  - DO NOT PANIC! Even experienced developers get stuck with git issues
  - [https://ohshitgit.com/](https://ohshitgit.com/)
  - [https://stackoverflow.com/questions/tagged/git](https://stackoverflow.com/questions/tagged/git)