What is Git?

A version control system

i.e. allows users to collaboratively work on code

https://git-scm.com/
What we will do today

- Learn the basic workflow of Git
- Learn how to push your code to your teammates
- Learn how to get your teammates’ code changes
- Learn (one way) how to reverse mistakes
- Learn how to view the status of your git repository
- Learn how to view the history of your git logs
What we will do next week

- Learn how to fix merge conflicts (!!!)
- Learn how to make independent pieces of development (a.k.a branches)
- Learn all the many ways to reverse mistakes
- Learn how to make pull requests
Some starting things

Vocabulary that you should know
**Important Version Control Vocab**

- **Github**
  - hosting service for Git
  - a web-based way to interact with Git, collaborate, follow other coders, etc.

- **local**
  - code on your machine
  - your version of the codebase

- **remote**
  - code not on your machine
  - the production codebase in Github

- **repository**
  - a project/set of files

- **working directory**
  - the folder you’re working out of
  - ex: /home/Desktop/myRepo

- **staging area**
  - a place for files getting ready to be pushed out
  - think of this like a loading dock
Demo: Github, cloning
2 Checking Git Status

```
git status

git log
```
git status

- shows state of the working directory and staging area
- we’ll be using this as we go through demos today (and you should irl too)
git log

- shows your commit history
Pushing Your Code Changes

git add
git commit
git push
The Git Process

add → commit → push
git add .  or  
git add <filename>
  - add file(s) and changes to the staging area
  - files that you change are *not* automatically staged

```bash
git commit -m "message"
```
  - collect the files in the staging area
  - record changes to the repo like a snapshot
Pushing your changes

**git push**

- upload local repository content to remote repository
- i.e. I’ve made changes and I want my partners to see them now!
Demo: Pushing Files
The Git Process

1. Working directory
2. Staging area
3. Git repository

- **add**
- **commit**
- **push**
Pulling Other People’s Code

git pull
git pull

- fetch and pull files from a remote repository
- updates files in local repository
- can get some conflicts from this – why?
  - *We’ll learn how to fix these next week!*
Demo: Pulling Files
Fixing Mistakes

git reset

git revert

git checkout
Resetting

**git reset <filename>**
- unstage file(s)
- i.e. I’ve done “git add” on file(s) that I don’t actually want to stage!

**git reset HEAD~1**
- remove the most recent commit
- git reset HEAD-\(num\) removes \(num\) commits before
Reverting

\texttt{git revert \{commit ID\}}

- undo the commit from commit ID as if it didn’t happen, but keep a record of it
- This creates a commit!
- i.e. I messed up, but don’t want to just delete the commit. Instead, I want the log to show that I undid a commit.
Checkout

git checkout <filename>

- change a file to the version on the remote repo
- i.e. I want the version of the file on the remote repo. Undo my changes to the file and switch it back!
Demo: Resetting vs. Reverting
The Git Process

1. Clone operation
2. Working copy
   - Edit, Add, Move files
3. Update operation
   - Modify working copy
   - Status and Diff operation
4. Review changes
5. Commit and Push operation
6. Commit changes
7. Amend and Push operation
8. Fix mistakes
Demo: Conflicts
Again, next week:

- Learn how to fix merge conflicts (!!!)
- Learn how to make independent pieces of development (a.k.a branches)
- Learn all the many ways to reverse mistakes
- Learn how to make pull requests
Thanks!

Any questions?

You can email me at
- amf269@cornell.edu