Intro to Git
Why Version Control?

- **Backup and restore** - because accidents happen
- **Synchronization** - multiple people can make changes
- **Short term undo** - that last change made things worse?
- **Long term undo** - find out when a bug was introduced
- **Track changes** - all changes related to a bug fix
- **Sandboxing** - try something out without messing up the main code
- **Branching and merging** - (better defined sandboxes)

https://betterexplained.com/articles/a-visual-guide-to-version-control/
Git

• For CSC207, we will create repositories for you

• These repositories live on a department server

• You will clone your remote repository, work on files locally, add and commit changes to your local repository, and push changes to the remote repository
Remote repository

- This is the repo that lives on another server.
- By convention we call it the *origin*.
- (In Github terminology, you may have an “upstream” repo and a forked copy of the repo called the “origin”, but we are working with a simpler model.)
The “clone” command makes a copy of the remote repository on your local machine.

Now there are two repositories. (Git is a distributed version control system)
Clone to get local repo

- The “clone” command makes a copy of the remote repository on your local machine.
- Now there are two repositories. (Git is a distributed version control system)
Local repository

- The actual repository is hidden. What you see is the working copy of the files from the repository.

- Now you can create new files, make changes to files, and delete files.

- When you want to “commit” a change to the local repository, you need to first “stage” the changes.”
How to get work done

- Make changes to files, add new files. When you are ready to commit your work to your local repo you need to tell git which files have changes that you want to add this time.

```
git add file1 file2

git commit -m "adding feature x"
```
Staging changes

- `git add` doesn’t add files to your repo. It just prepares changes to add to a commit.

- This means that when you make some changes to a file and then add and commit them, the next time you make some changes to a file you will still have to run `git add` to add the changes to the next commit.
A file can be in one of 4 states:

- **untracked** - you have never run a git command on the file
- **tracked** - committed
- **staged** - `git add` has been used to add changes in this file to the next commit
- **dirty/modified** - the file has changes that haven’t been staged

TIP: Use `git status` regularly. It helps you make sure the changes you have made really make it into the repo.
Typical work flow

- Starting a project:
  - *git clone <url>*

- Normal work:
  - After you have made some changes
  - *git status* (see what has really changed)
  - *git add file1 file2 file3*
  - *git commit -m “meaningful commit message”*
  - *git push*
Typical work flow

- Starting a project:
  - `git clone <url>`

- Normal work:
  - After you have made some changes
  - `git status` (see what has really changed)
  - `git add file1 file2 file3`
  - `git commit -m "meaningful commit message"`
  - `git push`

Save changes to local repo

Push changes to the remote repo
Distributed Version Control Systems: Paul joins the team

Server repo

- **git clone**

  - Local repo
    - Project
      - Justin's computer
  - Local repo
    - Project
      - Lindsey's computer
  - Local repo
    - Project
      - Paul's computer
Distributed Version Control Systems: Justin works on his current feature

Justin works on his current feature on his computer.

- **Local repo**
  - `git add FILENAME`
  - `git commit -m "message"`
  - `git push`

- **Server “origin” repo**
  - `git pull`

- **Local repo**
  - `git pull`

- **Project**
  - Justin’s computer
  - Lindsey’s computer
  - Paul’s computer
Distributed Version Control Systems: Paul has a conflict

Server “origin” repo

- git push

Local repo

- project

Justin’s computer

Local repo

- project

Lindsey’s computer

Local repo

- project

Paul’s computer

- git add FILENAME
- git commit –m “message”

Conflict in a file!