Version Control Systems and

git!

Why use a Version Control System (VCS)?

- Non-trivial software projects involve a *lot* of files

- Not only that, but there are lots of *interdependent* files
  - Changing the name of a function or method means *every file referencing it needs to be updated consistently, too.*

- When working on a project, you want to be able to:
  - Try out changes and refactorings *without risk of losing known-good work*!
  - Review the history of a project to know when and why certain changes were made or determine when some issue started to arise
  - Easily collaborate on *complete* snapshots of projects and features with a team
  - Switch between versions (v1.0 vs v2.0) to go back and fix bugs
  - ... and more!

What do Version Control Systems do?

- Track and store changes to files in a project's "source code **repository**" over time
  - Typically via atomic snapshots for the set of *all* files in a project
- Have workflows for collaborators to contribute to the same source code repository
  - Features for working independently and then "**pushing**" your work to a shared repository
  - Features for "**pulling**" others' work into your independent repository
  - Policies and procedures on how to handle **conflicting changes** between collaborators
- Are built with features for you to explore ideas without impacting stable version
  - In git this is the purpose of **branching**
- Version Control Systems can be centralized in a client-server model or distributed
  - git is **distributed** -- when you clone a repository you have a complete replica of it!

Why **`git`** over another VCS?

- Initially developed in 2005 by Linus Torvalds, creator of Linux, to be the version control system *for* the Linux operating system's code.

- In the last decade, **`git`** won out as the de facto VCS of engineers.
  - Previously: SVN (Subversion 2000) and CVS (Concurrent Versions System 1990)
  - Contemporary: Mercurial (2005)

- Why did **`git`** win?
  - It's fast... remarkably performant compared to prior VCS systems.
  - It's distributed... everyone has a project's *complete* history, no internet needed.
  - It's *immutable* by default... it takes effort to mutate existing commits.
  - It's *append-only*... it takes effort to delete old work since new changes are *appended*.
  - It's *robust*... it ensures integrity of data to avoid corruptions.

What is GitHub versus `git`?

- git is Version Control System software you install and use locally
- GitHub is a social web site for sharing and collaborating on projects whose source code is maintained with the git VCS
- You can use git without using GitHub, but not vice-versa.
- You should make a personal GitHub account and:

1. **Update your full name, location, and UNC affiliation in your profile**
2. **Add a profile picture of yourself**
3. **Add your UNC e-mail address if you signed up with a personal e-mail acct**