Git and VCS

CS101

What is VCS?

VCS stands for Version Control System. It's a way for developers to make checkpoints while developing that they can later revert to. It's also a useful tool for collaboration between multiple developers or even just one person across multiple devices. There's a lot more to VCS than that, but we will be addressing those features in future workshops.

What is Git?

Git is one of the most popular forms of VCS. You may have heard of GitHub, GitLab, or BitBucket, which are all websites that allow you to use Git remotely and have somewhere to store your code (although a lot of them also support other VCSs).

Why should I use Git?

- Allows you to save 'snapshots' of your code to revert to later.
- Allows your code to be saved remotely so you can access it on any machine.
- Allows you to easily collaborate on programs and projects with other users.

Some (well, a lot of) vocab

Repository (Repo)	Basically, a project's folder. A repository contains all of the project files (including documentation), and stores each file's revision history.
Commit	A commit, or "revision", is an individual change to a file (or set of files). Commits usually contain a commit message which is a brief description of what changes were made.
Pull	Pull refers to when you are fetching in changes and merging them. For instance, if someone has edited the remote file you're both working on, you'll want to pull in those changes to your local copy so that it's up to date.
Push	Pushing refers to sending your committed changes to a remote repository, such as a repository hosted on GitHub. For instance, if you change something locally, you'd want to then push those changes so that others may access them.

Source: https://help.github.com/en/articles/github-glossary

Remote vs Local

Remote Repository This is the version of the repository that is hosted on a server, most likely GitHub (other examples would be GitLab, BitBucket, etc.). It can be connected to local clones so that changes can be synced.

Local Repository The version of the repository that is hosted on your local machine. Repositories don't have to have a remote location, so you can have repositories that only exist on your machine

Getting Started

0. Create an account on GitHub, GitLab or BitBucket.
1. Install git on your machine through the terminal (or use the desktop client).
2. Make a new folder to practice with, let's name it `test`

Basic Git commands

```
git init
git add
git rm
git commit
git push
git pull
git clone
git status
git log
```

(In approximate order you would use it in)

These will all be explained in the next slides :)

`git init`

Once you are inside a directory (folder), `git init` will start a Git repository in that directory. You need to run `git init` before you issue any other commands.

`git add` and `git rm`

As their names suggest, you use `git add` to add files to track and `git rm` to remove files to track.

Examples:

- `git add README.md` - starts tracking the file README.md (or update the file if it's already being tracked)
- `git add *` - update all of the files that are currently being tracked
- `git add --all/-A` - update all of the files that are currently being tracked, and add all untracked files
- `git rm` - stop tracking a file

`git commit`

"Save" a snapshot of the files currently. After running this command you will be dropped into an editor to leave a **commit message** (you can choose which editor you use in your `.gitconfig`)

- `git commit` - save a snapshot, and go into an editor to make a commit message
- `git commit -m <message>` - commit, and leave a commit message inline without having to go into an editor

If you make a mistake in your commit message, you can run `git commit --amend` to drop into an editor and edit your commit message.

`git status` and `git log`

`git status` will show you what files are modified, untracked (not added) and uncommitted. It will also tell you what branch you are on, and if you are up to date with the remote repo (whether or not you need to pull).

`git log` will bring up a list of all of the previous commits. They will include the commit hash (basically the unique ID of that commit), the author, date, and the commit message.

Untracked/Tracked vs. Unstaged/Staged

- **Untracked** The file exists locally but isn't part of the Git repository. Basically, Git doesn't know about it.
- **Tracked** Git is tracking the changes on this file. This happens when you run `git add` the first time.

- **Unstaged** Git is tracking the changes on this file, and you've edited it since it was last committed.
- **Staged** Git is tracking this file, and the changes are ready to be committed. You do this by running `git add` after changing a file.

`git push` and `git pull`

After you've added files and committed your changes, run `git push` to push your changes from your local branch (on your machine) to your remote branch (GitHub/GitLab/BitBucket).

Running `git pull` will do the opposite: it will copy your remote branch to your local machine.

`git clone`

If you want to copy a remote repository to your local machine, you can use `git clone <git url> <optional destination folder>`.

Examples:

- `git clone https://github.com/user/repo.git` - clone repository with HTTPS
- `git clone git@github.com:user/repo.git` - clone repository with SSH
- `git clone https://github.com/user/repo.git foo` - clone repository into the foo directory

A challenge!

0. Log into lab machine
1. Navigate to home directory
2. Make a directory called **`test`**
3. Navigate into **`test`** (and check your present working directory)
4. Make a file called **`foo`**
5. Run **`echo ‘hello’ > foo`** to put ‘hello’ into the file **`foo`**
6. Copy the file **`foo`** to the file **`bar`**
7. List the files in the directory to check you have two files
8. Move the file **`bar`** to the file **`baz`** to rename it
9. Remove the file **`baz`**

A challenge!

0. Make a new repo on GitHub/GitLab/BitBucket
1. `mkdir test`
2. `cd test`
3. `touch README.md`
4. `echo 'hello' > README.md`
5. `git init`
6. `git add README.md`
7. `git commit -m "Initial commit"`
8. `git remote set-url <url>`
9. `git push -u origin master` (you have to do this the first time you push to a remote branch)
10. Go onto GitHub and edit the file
11. `git pull`
12. `cd ..`
13. `rm -rf test`
14. `git clone <url>`

Questions?

Thanks for coming!