**Introduction and setup**

Git is: A version control system

Website: <https://git-scm.com/>

A shell is a program that lets you interact with your computer's operating system. It lets you:

* Create, modify, delete files
* Run programs
* Wikipedia link: <https://en.wikipedia.org/wiki/Shell_(computing)>

For Linux and Mac, these operating systems have a program called Terminal; it is a shell.

There are many kinds of shells; the most popular one is Bash.

For Mac, Terminal is located in the Applications/Utilities folder.

For Windows, there is no program *exactly* like Terminal; something close is PowerShell. Instead, there's the program [Git Bash](https://git-scm.com/downloads). This is a Bash shell for Windows. It comes with Git.

For Mac, Git is packaged with the XCode command line tools. This package contains a collection of essential programs for developing code on a UNIX (Mac, Linux) system.

You can install the XCode command line tools by opening up Terminal and running this command:

* xcode-select --install

See [this post](http://osxdaily.com/2014/02/12/install-command-line-tools-mac-os-x/) for more details.

Moving to Git:

Git allows a single researcher to:

* Organize their projects
* Document their projects
* Back up their projects
* Share their projects

**Starting a repository**

What is a git repository?

A git repository is just a directory that you tell git to track changes in.

To make a directory a git repository, run the following command:

* git init

**Adding files to a repository**

You can check on the current state of your repository with the following command:

* git status

You can *add* files to your repository with the *add* command:

* git add

In order to save these files, you need to *commit* them. Do this with the *commit* command, which comes with a message:

* git commit -m "This is a new commit."

To look at the history of commits you've made, use the *log* command:

* git log

Press q to exit the commit history.

**Making a remote repository**

So far, this has all been on our personal computer in a *local* repository. We can store this in a *remote* location using git hosting services like GitHub or GitLab.

These services should have a "New" or "New repository" button that lets you make an empty copy of your local repository in the remote location.

These services then give you instructions to copy-paste on your personal computer to back up your local repository at the remote location.

Typically, this looks something like this:

* git remote add origin <URL>
* git push -u origin master

Now that your repository is backed up remotely, you can download it and work on it from anywhere.

**Working with a remote repository**

To download a remote repository, use the *clone* command:

* git clone <URL>

To save changes from your local repository to the remote location, use the *push* command:

* git push -u origin master

To incorporate changes from the remote repository into your local copy, use the *pull* command:

* git pull --rebase

To restore your repository to the state it was in at a previous commit, use the *reset* command:

* git reset --hard <commit hash>

This will restore your *local* repository to the state of that commit, but the history after it will still exist in the remote repository.

**Staging files for a commit**

Git gives you some ways to make it easier to stage files for a commit in bulk. These are:

* git add <directory>
  + Adds all untracked and modified files in <directory>
* git add .
  + Adds all untracked and modified files; note the period after "add"

If you accidentally stage a file that you don't want to commit, you can use the *reset* command:

* git reset
  + Un-adds all files staged for commit
* git reset <file>
  + Un-adds this specific file

**Ignoring files**

If you have files in your repository that you don't want git to track changes for, you can tell git to ignore them in a file called .gitignore.

This file is a plain text file that contains files, directories, or patterns of files that git shouldn't consider to be a part of your version-controlled repository. Often these files are big datasets or random word documents or PDFs that you want to keep in your repository on your local computer, but that you don't want to back up remotely through Git.

Here are some example lines of a .gitignore file:

* ignore.txt
  + Ignore this particular file
* \*.csv
  + Ignore all CSV files (or literally, all files with names ending in ".csv")
* output
  + Ignore the output directory and everything in it