**Git:**

* Share code, multiple people
* Version Control System
* Tricky
* Easy to use on your own
* The Problem: how do we have many people work together on the same files, without losing anyone’s work and keeping work across all members consistent
* VCS
  + Version - its some state of the files --every we save we get a new version
  + Control system - how to relate versions to each other and make sure all saves ever made are still accessible
  + One benefit( besides having our files saved) is we can look at the difference between any two versions
* Easiest level, we can easily give files to a collaborator by storing a git project on a remote git repository
* Branches, its a divergence of code with the intention of being added back to the main after completion. ( enable advanced collaboration)
* Fork repository, rebase them

**Is Not GitHub**:

* Remote repository for git projects
* With tools to enable teams to communicate about those projects

**How to Use Git**:

* Git clone some\_url.com/toAGitProject.git ( this is for downloading remote code onto my local machine)
* We made some changes and want to save them
  + Git add <some path> this command tells git what changes you want to prepare to save
  + Git restore does the opposite of git add, and removes a file from saving preparation
  + Git status checks the status of our project and its save state
  + Git commit
* To share code
  + Git push - it takes all of the commits locally and pushes them to the same branch on the remote repository
  + Git pull - takes all of the new commits from a remote repository and add them to the local repository
* Collaboration
  + This means we are going to be using branches
  + They will not prevent merge conflicts, but should make them easier to see and solve
  + These are for when you want to add new code, but not modify the code in the master branch, yet
  + Git fetch -- it gets all data related to branches from the remote repository and downloads it locally
  + Git branch -- like status, can use to make new branches but generally don’t
  + Git checkout <branchname>-- this is for swapping between branches, it has a couple important flags. -t -- track a remote branch, -b -- to make a new branch
  + Git merge <branchname> -- takes a branch and merges the code into your current branch

**Terminal:**

* Is a program designed to allow you the human to interact with the operating system of your computer.
* Shell the program running in the terminal that processes what commands you write
* Bash - Bourne Again Shell

**Most Important Linux Commands:**

* Ls - list, it lists all files in your current directory ( folder )
* Cd - change directory - this is how you move around in the terminal
* Pwd - print working directory - where am I?
* Mkdir - make a new directory
* . - the current folder
* .. - the folder above our current folder

**VI:**

* i -- this enters insert mode for writing
* Esc -- this takes you back to command mode
* : -- for writing a command
* W - write
* Q - quit
* :wq - command vi to write and then quit