Git 101 through Coursera

**DevOps and Git in a Nutshell**

1. What is Git?
2. Git is great for continuous improvement of a project
3. Git allows you to manage many versions of a project
4. Managing Project Version
5. Each version of a Git project is called a “commit”
6. The collection of commits contains the project’s history
7. You can review the history of commits of your project and go back to any version of your project
8. Branches
9. Branch: an independent line of development of a project
10. All commits belong to a branch
11. By default, there is a single branch and it is called “master”
12. Branches let you independently work on a project without disrupting the rest of the project
13. Pull requests
14. Pull requests will merge independent branches into the master branch
15. When you execute a pull request, you are requesting to merge the commit that includes whatever feature you added/bug you fixed into the master branch
16. During a pull request, team members can discuss, review, and approve your changes

**Git Overview**

1. Version control overview
   1. Version control is great. You can easily manage the various version of your project and you can have multiple team members work on the same project at once
   2. There are several types of version control systems
      1. Distributed version control system: In this system, each user has a local copy of the complete history of the project which is known as a repository. Team members can each have a copy of the repository on their local computer but there is a single remote repository that is designated as the “official state of the project”
2. Git overview
   1. Git is a distributed version control system
   2. Git is an open source software
3. Command line vs user interface
   1. Should I learn Git using the command line interface or the SourceTree graphical interface?

**Git Locations**

1. Working Tree: the location on your computer that contains the directories and files of a single commit. This is where you edit and view the files of a project, preparing them for the next commit
2. Staging Area/Index: contains a list of files that are planned to be included in the next commit. You prepare the staging area just as you want it so that the next commit is a meaningful snapshot of the project. You can use the add command to add new or modified files to the staging area
3. Local repository: contains all the commits that have been made for a project.
4. Project directory: where your working tree, staging area, and local repository are saved on your computer
5. Remote repository: Usually located in a data center/cloud. Contains the commits of a project on a remote computer

Committing to a Local Repository in SourceTree

1. When your first place a file in the working tree, its status is untracked. The file is only in the working tree, not in the staging area or the local repository. In SourceTree, you can see a file is untracked by the question mark next to the file
2. Staged content is content that will be added to the next commit. In SourceTree, you can add an untracked file by selecting the three dots and then selecting “Stage file”
3. When you want to commit the staged content to the local repository, you can click on “commit”.
4. When you make a commit, you have to add a message that accurately describes the changes you made to the content. The commit message becomes part of your project history.
5. The History tab will show your commit history.

Create a Remote Repository

1. Remote repositories often have a url ending with git
2. I created a remote repository in bitbucket

SourceTree Push to a Remote Repository

1. There are two ways to start working with a remote repository and it has to do with if you already have an existing local repository
2. If you do not have an existing local repository, you will clone the remote repository to create a local repository that is associated with the remote repository.
   1. You basically go to github, go to the repository you want to clone, click on clone, copy the source url of your remote repository, and then go to Sourcetree, go to add new, click on clone from url, and put in the source url.
   2. SourceTree will basically create a local repository from your remote repository. Often times the remote repository is referred to as origin in the local repository
3. If you do have a local repository with commits that you want to push to a remote repository, you add the remote repository to your local repository. Once you’ve added the remote repository, you can push your commits to the remote.
   1. You basically go to the remotes tab of your local repository, right click, click on “new remote”, and then put in the source url of the remote repository you want to add your local commits to
4. “Clone” can be both a noun and a verb. As a noun, a clone is a local copy of a remote repository. As a verb, cloning is a process of creating a clone.
5. A “push” adds commits for a branch to a remote repository

These were the instructions for creating a remote repository in Bitbucket and SourceTree: <https://d18ky98rnyall9.cloudfront.net/_b94768569a0e4b84b60575439fd79245_lab3_1_06a_sourcetree_push.pdf?Expires=1546992000&Signature=U09RtMkFwHpLanU5GINbMpiMm3Fo06FOYlcITnXFnjM4bgQslu7VryM1H3B~nEvEcXmba4ZAjEKiK0WpwaaRuGEeB8K-f2c60KgNWvZkMUOl26PF78Fzam-51L9-k-mvKown7pwriDL241rqTzolYLmDI7YE6bh938LqjkLa0IE_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A>

So if I make a change on my local computer and I want to add it to my remote repository, I would make a push request

Once I’ve pushed my commits, they essentially become a pull request in the remote repository.

Someone reviews my changes and then merges the pull request.

Once they merge the pull request, my new commits are reflected in the remote repository.

I can then “fetch” or “clone” the lastest verion of my reomote repository to continue making changes.