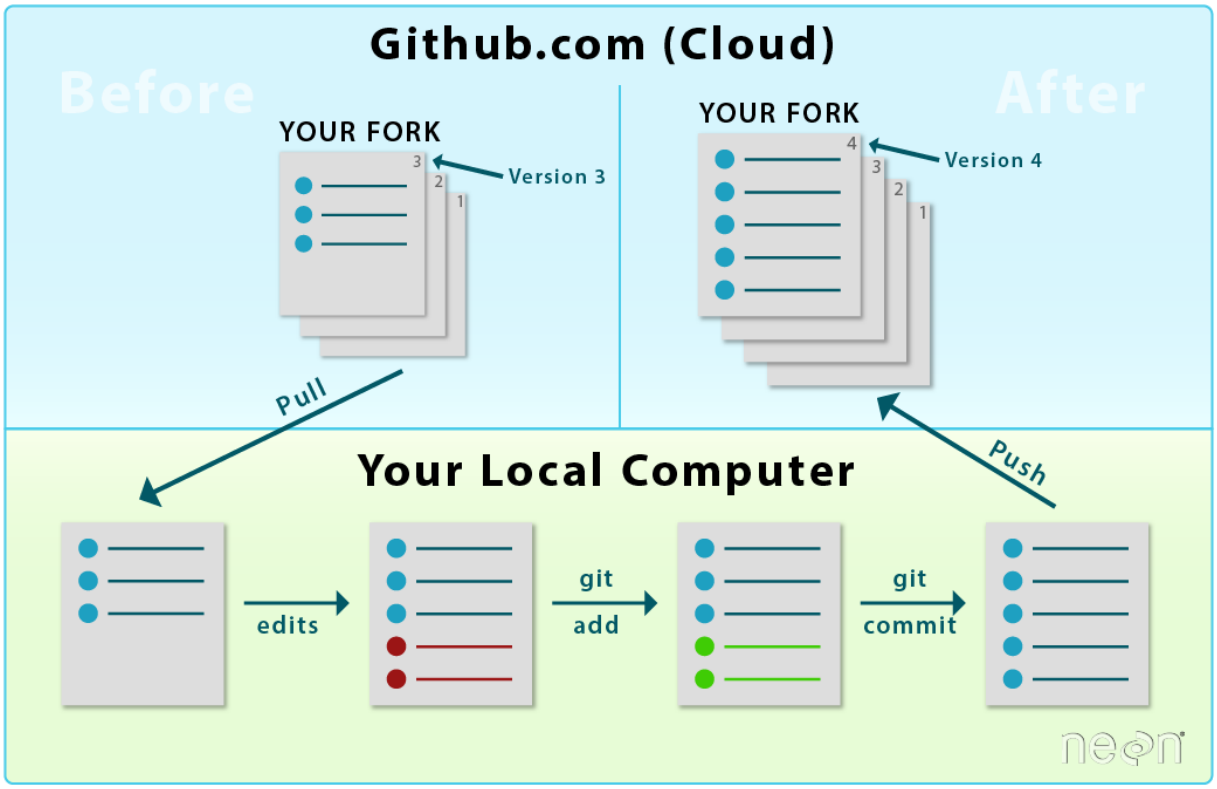
Git / GitHub Introduction

What is Git

Git is the most widely used version control system in the world today. Knowing how to use Git is essential for any software development role you may come across in the real world. There are several Git applications with user interfaces, but learning it through the command line is the best way to thoroughly understand what Git does and how it works. Below is a diagram demonstrating a basic workflow of Git.



Git is important for developing software because it prevents things from going horribly wrong (e.g. someone accidentally created a null pointer which breaks the entire codebase) and provides accountability for the contributions being made to a repository (e.g. seeing who create the null pointer exception). If something does go wrong and you cannot find the change is causing a detrimental error, you can simply revert the repository back to when it was fully working. Sharing files back and forth is not effective in the real world because codebases in production will be massive and many people will be working on them at once. Merge conflicts may arise if people are working on the same file. However, these conflicts can easily be solved. When a merge conflict occurs, you must correct it or else your changes will not be merged.

A repository is a codebase that will house your files. A repository has branches attached to it. The first branch is the master branch. The 'Master' branch should reflect your current working codebase and will be referenced in production. Then there is a develop branch that you should create but is not necessary. The develop branch will contain changes that will be merged into the master for new a patch or update/release. There can be additional branches for fixes or new features. Below is a standard workflow for branching in Git.

Some useful links to learn more about Git:

ARTICLE: https://hackernoon.com/understanding-git-fcffd87c15a3

VIDEO: https://youtu.be/r63f51ce84A

Basic Command Line Commands

Below is a list of commands that may be necessary for moving around your file directory. You will be with the command line during this course and you should additional explore more commands.

cd <directory>: changes the directory

mkdir <directory name> : creates a new folder

ls (for Mac) or **dir** (for Windows): lists all files and directories in current location

mv <from> <to> (for Mac) or **move <from> <to>** (for Windows): moves a file or directory to a new location

cd .. : moves back up a folder level in the directory

Creating a GitHub Account

1. Create a GitHub account using your TU email
2. Download Git. For Mac, use http://git-scm.com/download/mac and for Windows, use http://git-scm.com/download/win.
3. Set up local configurations to attach your Github account to your commits:

   git config -- global user.name "firstname Lastname"
   git config -- global user.email "yourGithubEmail@email.com"

We highly recommend applying for the GitHub Student Developer Pack. It provides a ton of free resources including courses and also gives you the ability to create private repositories. Apply for that here: https://education.github.com/pack

Cloning a Repository

1. Cloning makes a copy of an existing repository. You can clone a repository using its link on Github.

   git clone [url of repository **]**

   You should now fork the repository at https://github.com/gk-3207-TU/3207-Lab-Introduction. There is "Fork" button in the top right of the page after clicking the link. Forking creates an exact copy of a repository and displays it on your Github. Forking is useful for proposing fixes/features in open source repositories. Clone the forked repository to your computer using the command above. (We recommend that you clone it into your desktop for ease of access. Use the command **cd desktop** to change your directory to be your desktop and then the clone the repo.)
2. The cloned repository has a file named **print_random.c** This is a simple program but is incomplete. You are to develop the function **randchar()** as a separate file (**random.c**) and include it in your repository. The function randchar() is a character function that returns a random character from 'A' – 'Z'. The character is used in the main program to generate a random 7 letter word.
3. Create a new file in a text editor. Make sure it is in the folder of the repo you cloned. You can add that new file for staging in the command line using

the command **git add random.c** Track your first file using the above command.

4. Checking the status of your repository is crucial to know what has yet to be staged or what is currently staged. It also provides information on what branch you are currently on. Use the command **git status** to check the status of your repository. You should currently have one file staged.
5. In your editor or IDE, edit the new file that you created and save the changes. You can also do this through the command using Vim. Check the status of your repository to see that it needs to be staged. Use the command **git add -p** to go through each of the changes and answer 'y' for yes and 'n' for no. If you make every change needed for a branch without committing along the way, the PR will contain 1 commit which may be tough for a reviewer to follow. Going through each change and picking if you want it to be staged or not is useful for organizing each step you made along the way. Use the command **git commit – m "a message"** to commit these changes. The message will be what is displayed for each commit. A Git commit message should be short but still effectively describe the functionality the changes in the commit will bring.
6. To push these changes to the server, use the command **git push to** do so. The reason you may simply use this command is because you are pushing directly to the master branch which your local repository knows about. If you create a new branch on your own, you will need to let Github know to track it. The command for that is

   git push – set-upstream origin [name of branch **]**
7. Congratulations, you have successfully created your first Git commit and sent the changes to the server!