Git & GitHub Basics
Learning Git

Understanding the Workflow

Initializing a Repository & Staging Files

Comparing Files Changes

Github & Using Clone, Push, & Pull

Understanding the Workflow

There are three key pieces to the workflow, the project directory, the staging area, and the git repository.

The **project directory** is folder where the project files are maintained on your local machine. The **staging area**, formely known as the index, is a pre-snapshot review area for the files prepared to be commited to the repository. Finally, the **git repository** is a hidden sub directory in the local project directoy.

Now the workflow for how files are managed using git: files located in the project directory are manually added to the staging area for review by the user. Once confirmed, files are commited from the project directory to a git repository. This means a snapshop of the project directory is stored in the repository, or the repo. Git uses a technology that requires very small amounts of storage, meaning a large collection of snapshots can be stored at a low cost.

Initializing a Repository & Staging Files

To establish a new git project, use any local directory and, using the Command Line Interface, enter `git init`. This will create a hidden ".git" directory. To see this directory in the terminal, use the `ls -a` command.

Once initialized, files can be added to the staging area in preparation for committing. This is done using the `git add` command. You can specify files, patterns or add the entire directory recursively. Each is shown below using a sample.

`git add fileName1 fileName2` will add the specific files.

`git add *.txt` will add all files ending with the extension .txt.

`git add .` will add all files recursively.

Status, Unstaging Files, & Committing

With files in the staging area, `git status` is used to check the staging area status, prior to committing a snapshot. Files in red represent files not in the staging area. Files in green represent files in the staging area and are awaiting changes to be committed.

If you find files you need to remove from the staging area, use git restore --staged followed by the specified files. Listing files can be done by using the individual filename, patterns, or recursively. It follows similar syntax to git add.

Once the staging area is prepared to the user's liking, using git commit will commit the snapshot within the .git directory. When using this command, add a message to explain to others why the commit was made. This message is done using: git commit -m "Message goes here"

Comparing File Changes

`git diff` allows the user to compare two files and inspect the differences between the two texts.

`git diff --staged` compares the new staged files with the version already on git.

Using the command line, the following information is displayed:
Line 1. States the diff utility is called with using the stated arguments. The later files is the newer file in the staging area.
Line 2. Meta data is presented. It is not important to understand this portion.
Line 3. A legend is provided explaining that changes in the old copy are indicated by "-".
Line 4. A legend is provided explaining that changes in the new copy are indicated by "+".
Line 5. A header summarizes what lines have changed.

`git diff` has more options to use allowing the comparisons of various files. There are also GUI tools for git diff. Some include: KDiff3, P4Merge, WinMerge, and Visual Studio Code.

Github & Using Clone, Push, & Pull

When using a hosting service like GitHub, there are three key commands to learn: `git clone`, `git pull`, `git push`. Setting up GitHub will not be covered in this documentation.

`git clone` is used when a remote repo exists, but a local one does not. `git clone` will take remote repo, clone it, and create a local directory for you to match it.

`git pull` is used to grab the latest changes to the git repo, and merges them with those on your local directory.

`git push` is used to push upstream (to the remote server) the user's local git repo commits.

I will not be going into any further details on these commands. Check out this post to get some more details on these basic commands and GitHub. To learn how to use them, check out this video on Git and GitHub for Beginners by freecodecamp.

Disclaimer	The information provided is only a brief on the tools of git. Each subject mentioned here should be looked into further to be fully understood. As a starting point, check out this video on Git Tutorial for Beginners by Mosh.