**Git Basic Command**

**Git workflow:**

* There are 4 stages in git workflow

1.Working Directory

2.Staging Area

3.Local Repository

4.Remote Repository

1. Command for Untracked file i.e., Working directory to Staging Area: **$git add filename(start.txt)**
2. To check status: **$git status**
3. For moving file from Staging area to Repository:

**$git commit -m “Commit message” or $git commit** (This opens editor for writing commit message)

1. To Push a file from Repository to Remote (GitHub):

**$git push origin master**

1. To create new Repository: **$git init Reponame(fresh-project)**
2. To list all the files and folders: **$git ls -al**

* In this a stands for all and l for listing.

1. To create the new file: **$git notepad++ filename**

* In this the notepad++ is my default editor so I used that instead you can use any editor you are comfortable with.
* By creating the new file there will be untracked file so we have to add and commit that file.

1. To remove the repository and navigate it to another folder:

**$git rm -rf Reponame or $git rm -rf Reponame foldername** (in which you want to shift repository).

1. Add git to an existing project:

* First unzip the downloaded folder: **$unzip ~/Downloads/filename**
* Then move the folder to another folder: **$mv filename newfilename**
* To initialize the empty git repository: **$git init**

1. To add all files from working directory to Staging Area: **$git add .**(here full stop is compulsory with space in between)
2. If there is no .git folder then the project will not be manages by git.
3. Fork: It is a copy of repository that we manage. It let us make changes in the project without affecting the original repository.
4. To clone a git repository: **$git clone URL**
5. To know which are the folders and files that are managed by git

* First go into. git folder**: $cd. git/**
* Then list files and folders: **$ls**

1. After every commit we make we will get specific identification code or number [master 8ac7723]
2. Git status keeps track of origin master.
3. To know that the local repo is up to date with remote repo or to pull the newly added resources into remote to local: **$git pull origin master**

Origin is a remote repo and master is the branch name.

1. Push is used to put the commits that have been made in local repo to remote repo: **$git push origin master**

* Make a habit of using pull first and then use push because if there is any update in remote repo it will pull the resources first and then push function will be applied.

1. For changing the git configuration: **$notepad++ ~/.gitconfig**
2. To do add and commit in one command: **$git commit -am “commit mess”**
3. Git tracks the changes in the files that are staged and also the changes that are not staged in same file.
4. Recursively add files to git repo: **$mkdir -p level1/level2/level3/level4**
5. Backing out changes:

* Suppose if we made changes in level.txt and we added file to staging area, now we realize we don’t want changes and we want to unstage the file: **$git reset HEAD level.txt**
* By this the file will unstage the content of the file will remain same.
* But we want the original content back and don’t want the added text: **$git checkout --filename**

1. Renaming and Moving files:

* To rename the file in git: **$git mv oldfilename newfilename**
* To rename the file in operating system only not git: **$mv oldfilename newfilename**
* By renaming a file git status will show 2 changes that old file is deleted and new file is created. Now to add file recursively or the files that have been moved or renamed: **$git add -A**
* If we want to backout from changing the filename: **$git mv newfilename oldfilename**
* To move the file from one directory to another: **$git mv filename newdirname**
* Similarly we can rename it from file explorer also, but this will give ../DS\_store on git status. So in order to get rid of this error we have add the file to staging area and update the index of file:

**$git add -u**

1. Deleting files.

* If file is not tracked by git: **$rm filename**
* If file is tracked by git: **$git rm filename**
* Backout changes: **$git reset HEAD filename** only done in staging area it will not show text file
* To show file after backing out: **$git checkout -- filename**

1. History or Help.

* To get to help page: **$git help commandname**
* To see the commit history: **$git log**
* To get commit id short: **$git log --abbrev-commit**
* To compress entries in oneline: **$git log --oneline --graph --decorate**
* To get specific commits only: **$git log ae6frde…654rdse**
* To get history of particular file only: **$git log -- filename**
* History of renamed file: **$git log --follow -- filename**

1. Set up Git Alias.

* Basically, used to make the commands shorter than the actual command.

**$git config –global alias.hist “log --all --graph --decorate --oneline”**

* Here hist is the newname for the command in “”.
* To see all the alias: **$notepad++ ~/.gitconfig**

1. Ignoring unwanted files.

* Create a file .gitignore: **$notepad++ .gitignore**
* Format for this file:
* Specific file: myfile.txt
* File Pattern: \*.txt (all .txt files)
* Folder: myfolder/
* Then add .gitignore file and commit it.