**MY GIT SUMMARY**

**INTRODUCTION**

**Git** is a **software** used for free and open source version control system. It is a tool that tracks the changes of your code over time.

**GitHub** is a web site to host your repositories online. It help many programmers to work on a project easily.

**Version Control** is the management of changes to documents, computer programs, large web sites, textbooks and other collections of information. There are kinds:

* Local version control system
* Centralized version control system, where there is a central server that is been managed by individuals directly.
* Distributed version control system, here all contributor have the file locally which they use to manage the central/remote repository like GitHub, BitBucket, …etc .

**TERMS AND MEANING**

**DIRECTORY** is another name for **folder**.

**add** is a github command used to track your files and changes in Git.

**cd** means change directory in your command line.

**CLI** means command line interface.

**clone** is a github command used to get a repository that is hosted somewhere like GitHub into a folder on your local machine.

**commit** is a github command used to save your files in Git.

**dir** is used to print all the files and folders present in the current directory.

**GPG**( Gnu privacy guard) is used to create a privacy means to send information through the network. it is mostly used for signin tags and commits.

**pull** is a github command used to download changes from remote repo to your local machine.

**push** is a github command used to upload Git commit to a remote repo, like Github.

**remote** is a git command that is used to refer to remote repository like github, Bitbucket,.. etc.

**Repository** is a project or folder where your project is kept.

**SSH**( secure socket shell or secure shell) is used to create a secure channel of sending information, mostly through the internet. it is mostly used for authentication.

**Stashing** is a way of saving some changes in a temporary location for user to perform other actions like switching, reverting, … etc. These changes can be reapplied somewhere elses.

**Terminal** is another name for **Command Line**, which is an interface for text command. Go to your PC search tab and search ‘cmd’, then hit enter to see what I meant.

**INSTALLATION AND LOGIN**

**SIGNUP AND LOGIN:**

1. Firstly, search **‘Github’** in your device browser
2. Then click the command that will take u to sign-up. To sign-up, you need **Username**, **Email** and a **password**.

**INSTALLATIONS**:

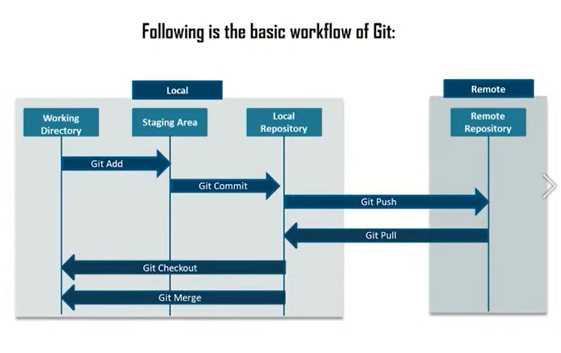
**Installing Git**:-

Search for how to install git to your operating system, it is very easy that is why I didn’t bother adding the steps here.

**Installing Editor(s)**:-

Just download and install Visual studio code, it will also help you as a developer.

**COMMANDS AND THEIR USES**

****

**Image of git repo stages.**

**#** is used to add main header to text.

**.md** is a file extension associated with github’s readme file( README.md). it is an abbreviation for **markdown**.

**GITHUB COMMANDS:**

**Your profile** command takes you to **your about page**. Here You can edit your profile like picture, other account, email, … etc.

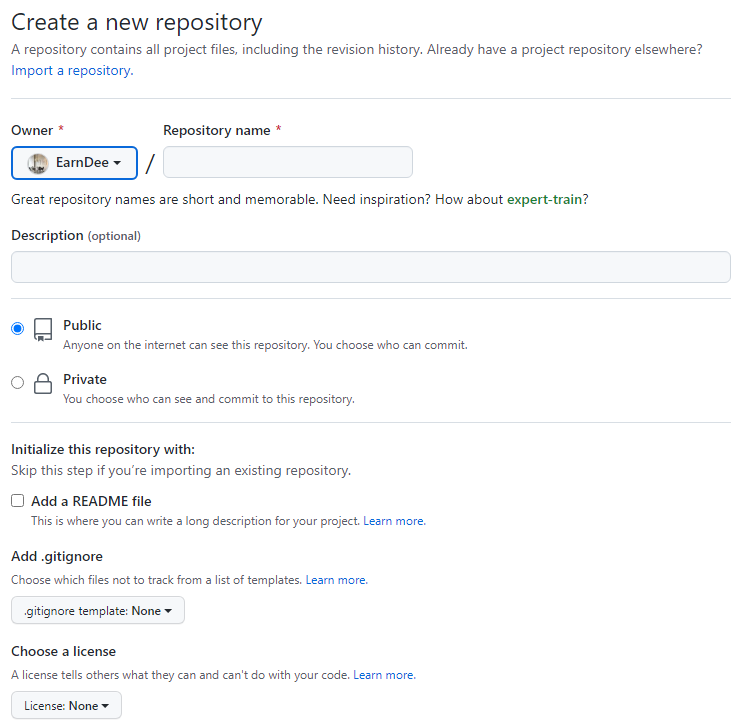
**GitHub Logo** when clicked, take you to your **dash board**.

The **plus-sign** at the top right or the **new green button** helps you to create a **new repository**.

**FILE CREATION AND MANAGEMENT**

**CREATING A NEW REPOSITORY**:

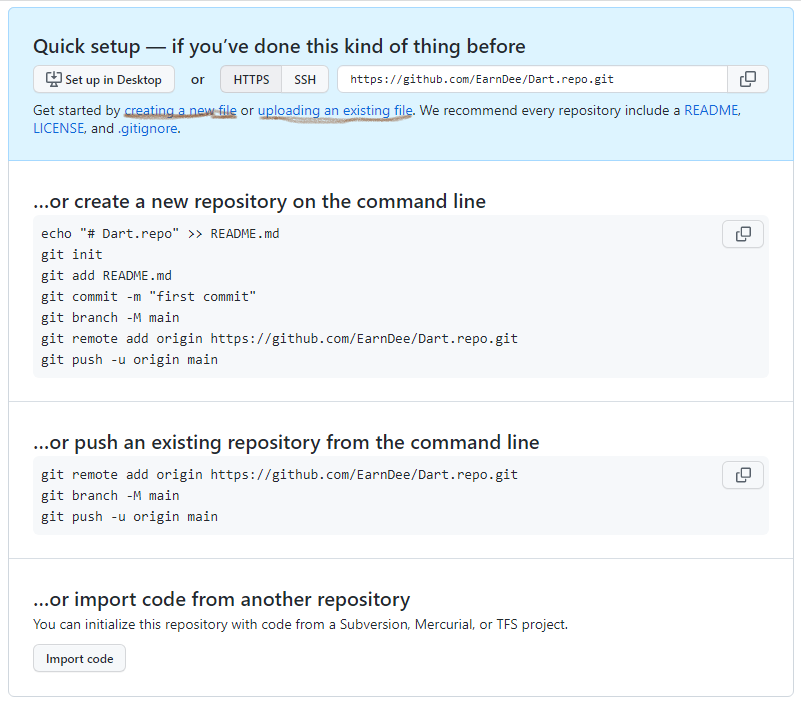
1. Click the green button to create a new repository. You will be brought to the below page.



1. The only required input in the above image is repository name, other inputs help you to make some restrictions and add flavor to the repository you want to create.
2. Description is a small message that explains your repository.
3. public let every one view and use your repository.
4. private make you select who can view or use your repository.
5. Add README file creates a readme file where you specified more description, do and don’t that are associated to your repository.
6. Add .gitignore helps you to select files that won’t be pushed or pulled to or from your repository.
7. choose a license help you to make your repository an open source.
8. Repository name is adviced to be **short**.
9. Then click **create repository** below, to move to the next image below.

**ADDING NEW FILE**:

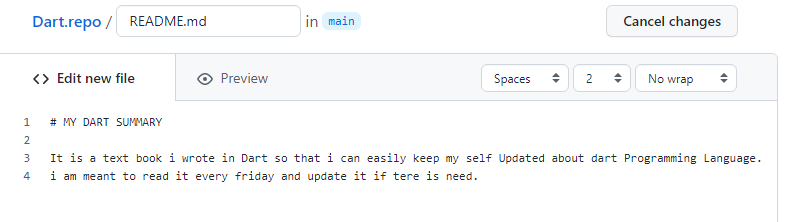
1. Click on **create a new file** under **quick setup** as in the image below to create a new file.



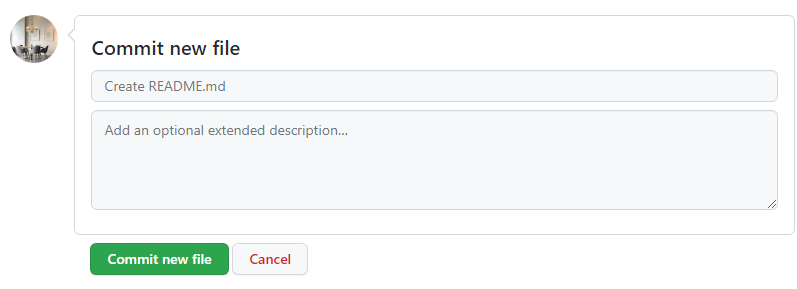
1. Click on **uploading an existing file** under **quick setup** as in the image below to...
2. The last two sections in the above image are just another way to create or add new repository or file.

**EDITING FILES**:

1. When you click on create a new file, you will be take to the below image. Here, you can edit filename like mine in the below image is **README.md**.



1. **#** is used to make **MY DART SUMMARY** bold.
2. After you have inputted your text, click commit new file as in the below image. The Add an optional extended description in the below image is used …



**CLONING A FILE**:

Cloning a file is a means to download file(s) to your local machine so that You can use it offline. You can clone both your file and other peoples file provided that it is public.

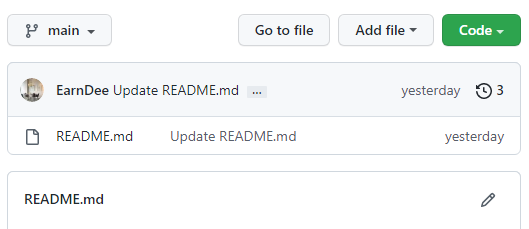
GitHub provides three secure roots to clone your files, which are https, SSH, and Github ClI( used by contributors).

To clone a file, open the repository and click the **green code button** as in the image below.

**https** is mostly used by those that have no **ssh** in their github account to clone a file to a local device.

**SSH** is used by those that have included it in their github account.

**GitHub CLI** is used …



After selecting the root you want to clone your file from, go to your PC command line (cmd) . Search with **cd** to the file you want to include the repo on, then type:-   
git clone the\_link\_you\_copied

then hit **enter**.

NB: You must have **git** installed and a **network connection**.

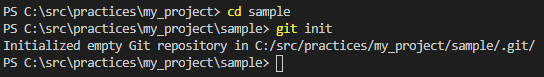
1. **GUIDE**

**Initialing git in a particular directly:**

**git init** command is used to initialize a new repository in git locally(without internet).

This is a task you must do before you can start using git as you version control. You should navigate to the directory you want to initialize in your commandline before you called **git init** command.

Example:



As you can see, I navigate till I was at **sample** –which is the directory that I want to work on—before calling **git init** command.

**Configuring our git:**

**git config --global user.email “EarnDee1@gmail.com”** command is used to set your git account email address.

**git config --global user.name “EarnDee”** command set your git account Username.

**git config -l** command is used to list out all configurations u have made like user.name, user.email, … etc.

**Adding files or folders to the staging area of our git:**

**git add .** command is used to add every files that are not added to th staging area of your git account.

**git add <file\_name>** command is adds one file to the **staging area**, as shown in the above image.

*NB: This must be done before you can commit the file.*

**git restore .** command is used to delete all new changes that have not been add to the staging area.

**git restore --staged .** command is used to restore/ unstage all new changes that have not been committed to the staging area.

**git restore <file\_name>** command is restore/delete all changes made in the file that are not added yet.

**git restore --staged <file\_name>** command is restore/unstage all changes made in the file that are not committed yet.

**committing files or folders to our local repository:**

**git commit –m “small description about the file”** command adds/commits your file that are **added** to the **staging area** with git add command to your **local repo**.

*NB: This must be done before you can push your files to a remote repository.*

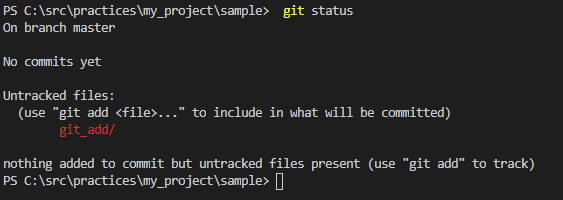
**To reverse a commit**

**git revert HEAD~3** command reverses to the fourth last commit.

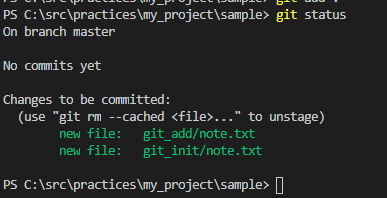
**git revert --help** command helps you to see more that you can do with git revert.

**Checking the stage where our files and folders are:**

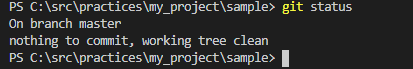
**git status** command tells you more about your account adds, commit, … etc .

****

This is the where files are yet to be **added** to the **staging area** of our git.



This is where you have **added** the files to the **staging area** but yet to **commit**( add) it to your git **local repo**.



This is when you have committed the files to your local repo. You can now push to a remote repo.

**To check for your commits and it’s info**

**git log/log branch\_name**  command gives you a summary of all the commits, like time, id, email, associated to the branch you are in or you specified.

**git log --oneline** command helps to print out log info in single line for each commit.

**git log --online branch\_name** command helps you to print the log associated with the specified branch\_name no matter the branch you are presently on.

**To check info about branch or create a new branch**

**git branch** command prints out all the branches you have with the branch name with asterisk being the branch you are presently on.

**git branch -a** command prints out all local and remote branches you have.

**git branch any\_name** command is used to create a new branch.

**git branch -b any\_name** command is used to create a new branch and switch to it at once

**git branch -b any\_name 5efht78a** command is used to create a new branch from a particular commit and switch to it at once.

.**git branch -b any\_name HEAD~4** command is used to create a new branch from the 4 last commit and switch to it at once.

**git branch –d branch\_name** command is used to **delete** the specified **branch** from your **local repository** but the branch merge have being merged.

**git branch –D branch** command is used to **force** **delete** the specified **branch** from your **local repository** whether the branch is merge or not.

**git branch –m old\_name new\_name** command is used to **rename** your **local repository branch**.

**git checkout branch\_name** command is used to switch from present branch to another branch in your local repo.

**git checkout hash\_id** command is used to switch to a particular commit.

**git push alias --delete remote\_repo\_branch\_name** command is used to **delete** the **remote repository branch** from the **terminal** without opening the remote repo in a browser or in app form.

**git push –d alias branch\_name** command is used to **delete** a remote branch name.

**git push alias :branch\_name** command is also used to **delete** a remote branch name.

**git push alias :old\_branch new\_branch** command is used to **rename** a **branch** in the **remote repo** but you must have renamed the local repo first.

**To get more info about a particular commit made**

**git show few\_part\_of \_that\_particular\_commit\_id** command show more about the changes made by a particular commit done in the local or remote/central repo.

**To list files that are tracked by git**

**git ls-file** command list all available files that can be seen by git in your repo( local or remote). ***Files that are added(not committed) can be seen also.***

**To link and check if a remote repo is linked to your local repo**

**git remote** command is used to print out all the available linked remote repository name (alias).

**git remote add origin remote\_repo\_link** command lets you link a remote repo with your local repo.

**git remote rename formal\_alias new\_alias** command is used to rename a remote repo.

**git remote remove/rm alias** command is used to remove added remote repository from your local repository.

**git remote -v** command helps you to check if you have linked your remote repo to the local repo by printing out the remote repo name and remote path.

**To move file from your local repo to your remote repo**

**git push origin master** command pushes your local repo changes to the remote/central repo. This will request for your Username and password for the remote repo if it is your first time to login. **origin** is the **alias,** which can be any name and **master** is the **branch** that you want to push it to.

***NB:*** *if the* ***remote repo*** *is* ***more updated*** *than the* ***local repo****, you will require to* ***pull*** *b4* ***pushing****.*

*Also if there is no branch name as the one you pushed, a new one will be created with the name in your remote repo.*

**git push –f origin master** command is used to force when normal push couldn’t work because of difference in files. This will override all file and add exactly the files you force push in that branch.

**To download a remote repository**

**git clone the\_link\_you\_copied** command is used to download a repository to your local device.

**git fork …** command helps to get original copy of a repository to your github account repository.

**git fetch origin master** command is used to update your remote branch ref stored in your local devices. Use **git branch -a** to see it.

**git merge --abort** command is used to delete/decline a merge request when u pull a more updated remote repo than your local repo.

**git pull origin master** command pulls remote repo to your local repo for you to easily make change and test.

**To merge to branches together**

**git merge branch\_name** command is used to merge a particular branch (**branch\_name)**with branch that you presently on.

**git rebase master** command is used to change the base of the present branch to include what is in the master\_branch , which will now have different address.

**To compare files**

**git diff file1\_location/id file2\_location/id** command is to compare the changes made in files with different commit id, different location( local repo).

**git diff file1\_branch file2\_remote\_local\_branch** command is used to compare local file with a remote file.

To copy other people’s repository

Go to the account click fork, this copies the repo to your our github account.

Then go to clone to copy the path as you would do with your repo to add it to your local repo with **git remote add alias the\_copied\_https\_ref**.

**To stash files**

**git stash** command is used to save uncommitted files into a temporary local.

**git stash apply or git stash pop** command is used to unstash a file for you the use it. Only unstaged( not added) files can be used in another branch that is not the branch you were when you called git stash. Added file but not committed files can’t be applied in another branch unless you unstage it before stashing it.

**git stash list** command prints all available stashes and their hash\_id.

**git show stash** command prints available stashes with more detail, this can be controlled by adding **--oneline** at the end as done in **log**.

**Others:**

**git archive master --format=zip –output=../file\_name.zip** command is used to archive the master branch or any other branches.

**git bundle create ../repo.bunbler master** command is used to create a new bundle of your master branch.

1. **TASK**
2. **.**