Git and Github

what is git?

It is a version control system that allow u to store, update, edit our code base. lets say we have app v1.0.0 then we can make changes and can update our app to v1.0 this is what git allow us to do. Basically we can store history of our project and can share it with our people, co-workers.

- It keep tracking changes in the source code, enable multiple developers to work together. It can store any kind of file like txt, csv etc.

what is github?

Github is a web-based version-control and collaboration platform for developers.

- It has huge com

commands :

- use Git bash, if you have installed git or can use command prompt.

1) To check git version
$ git --version

2) To set config (mean adding username & email) :
- It helps other people to get to know your identity.

git config --global user.name "your name here"

3] to set email:
git config --global user.email "your email"

4] To check whether username/email set or not:
- same command i.e. used to set them.

5] To edit username/email:
git config --global --edit → To enter config file

6] Step to exit config file edit mode
ESC → To enter command mode
then,
:wq + enter → To exit

Some Basic Command
mkdir directoryname - to create directory
cd foldername - to get in folder
clear → to clear gitbash screen.

7] To clone repo:
→ git clone repo-link

vim Readme.md → to get into readme file | then press i to get in insert, then
cat Readme.md → to see the update you have made. | ESC after and type :wq to exit.
↑ must be CAP

{optional = if user want to use}

what is respository?

It means a project that you can track with the help of git.

command :

1) if you want a folder in git respository then, follow below step:

step1] (i) git init

{After this command an empty git repo will get initialized in that respective folder}.

(ii) To ~~set~~ see all files in repo {optional}

→ ls

(iii) To see all hidden files {optional}

→ ls -a

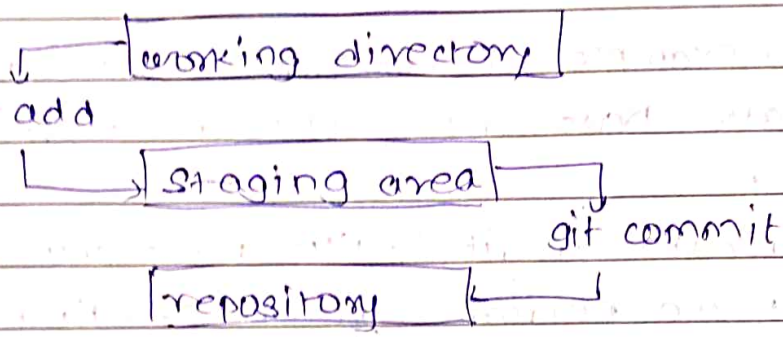
(iv) git status → it shows all changes {optional} made in that respo.

step2] (b) To add files, so that git can track them

→ git add filename {instead of filename if you want to add all file use --all or .}

step3): staging area:

After adding file it goes into staging area. In simple word, it is a place where you keep holding all the changes that you have made before making commit.



Git commit:

step 4) commit: below command is used to commit your work.
git commit *** -m "initial commit"

* { To see all your commits → git log }

* { -m : means you are writing any message regarding commit &
" " : In bern double coat we write our message }
* it should be meaningful.

* Git checkout :

whenever we make commits our history is getting saved.

In git we have power to go back to any of that commit. In simple words if we had made 5 commits & with each commit we have added 1 new file. So git allow me to go to any commit lets say commit 3, at this moment i will have all 3 file since I'm in commit 3.

To do this, we have a command,

→ git checkout Hashcode

{ you can find hash code by using git log command with each commits a hashcode get added to it, for location purpose }

→ git checkout master { to get to your master, master is similar like root node }

{ for example, if you use anaconda there we have base env & we can create multiple env.
so, master in git is like base env & commit that we perform go on creating new env }

Branch:

just like we can make multiple env in py. & we use, let's say conda activate to go /move in multiple branch (env).

Similarly git has branch concept, it shows all branches which include all commit, master, ~~branch~~ other your define branch.

1) To see all branch → git branch
{ * shows current/working branch in list of branches }

2) To change/branch → git checkout hashcode
to get in ↓
{ use hashcode for commit / type "master" for master
{ name of user define branch to get in } branch }

Explanation:

There is a master branch just like root node of a tree & you can make multiple branches from that master branch.

3) To create branch → git branch name/feature of branch
{ name of branch }

Or use
→ git checkout -b branchname/feature of branch
{ -b will create branch & checkout get u in it. }

To delete branch → git branch -d branchname

Git
merge:

To merge → git merge branchname

Merge is used to merge two branches. basically if two branches have different codes. lets say one branch has active app & other has the changes developer made. so to apply those changes we can use merge

what merge will do is, when we merge two branches it will commit all the changes & files that the branch 2 has in branch 1.

Note: other branches will not get modified why performing this.

e.g.

git bash

a22 /c/portfolio (dev)

$ git merge changes

Here, all unique / different / changed files in branch changes will get commited in branch dev.

{Note: All changes are made in different / child branches like dev, changes from above e.g. and once all your development part got finish you can merge those dev, changes file to master & send them to production.}

* git ignore:
when you want that git should not track some file, you can put them in gitignore file.
e.g. we have a file secretkey and we want that git must track all files except secretkey file. so you can put them in gitignore.

It looks like → .gitignore
and {inside this file name all the file that you want to keep hidden}.

* To push existing repository:
1) git remote add origin {copy from github repo}
2) git branch -M master ← you can also change name from origin to userdefine
3) git push -u origin master
   - master → which branch master
   - origin → keep it same that used in step 1)

* Git fork:
lets say your friend has made a repo & he has added some file, now you want to update and add some file in it. Here you can use fork, it basically will create the same repo in your github account.