Git and Github

- What is Git and Github and why we use it? ~~use it?~~

→ Git is a version control system, which allows multiple developers to collaborate on projects efficiently.

Git allows developers to track changes to their code base, manage different version of their software, and co-ordinate work with other team members.

Github : It is a platform (web-based) built on top of Git, here we can hold git repositories remotely, so that all the people across the world can collaborate and view my projects.

- Open git bash Terminal,

(1) mkdir : make directory
Command : mkdir project [project is the name of folder] / directory.

Now, Here we are working on GIT-Learn destop folder now I want to work on project directory. So we use

(2) cd : change directory
Command : cd project

Now, we wanna see how many files are there in my directory project, so we use.

(3) ls : List command
Command : ls

Now, in git, it monitors our changes so g to do so git has its own repository (folder) named as ".git" But ".git" is hidden.
To get this folder we use

4. git init : git initialize
Shows all the hidden folder;

5. ls-a this command shows all the files including the hidden one.
NOTE: any folder that starts with '.' is hidden.

We can also see what is inside '.git' folder by using "ls .git".

✱ Now we gonna make change in 'project', we are adding txt file named as "names".
To do so we use.

6. Touch command, used to create new file. (Linux Command)
Command: touch names.txt
'names' text file is created.

To check wheater the history of project is maintained anywhere, To know it we use command

7. Status : show the status of the project, any change or modification done on the 'project'.
Command: git status.

Now as you did git status we see an untracked file, Currently this file is not tracked by git, so to make git track this file we use

(8) add command; its a process of commiting that this file should be tracked.
Command: ① git add . [if we use only . is tracks all the files]
② git names.txt [only specifice file will be tracked]

Now to save it permanently we can write a msg. Command.

9) git commit -m "names.txt file added"

The commit is the order you give to save the change you made in the git repository.

Now, we wanna go in our names file to add few names or to make any change.

10 Vi names.txt ; with this command i can in the txt file & created to add any text i want.

11 Cat names.txt ; it display what all names / data have been added in the txt file.

Now we need to commit again about the changes in the names file.
we write "git add ." so now git knows about the changes.

Now if we see that we don't want to commit these changes so we remove them by.

12 git restore --staged names.txt

after this command the names.txt files changes is again unstaged.

To now see the entire history of the project we use

13 git log ; this command gives all the date and time when these changes were made.

To delete a file we use 'rm' command.

14 rm -rf names.txt ; with this names file is deleated.

Now, this deletion is added and commited by you as a mistake;
Here now in the file we see 3 commits.

- NOTE : you cannot remove 1 commit from middle to restore the previous commit; to remove the latest commit ~~we~~ we ~~have~~ have to copy the hash of the commit above which we wanna restore and then the command.

15. git reset "the hash" ef59543909f504e9da0cff6065fc...

now to check when we press git log we will see only one commit which has been selected.
Rest all the commit went to unstaged or say backstaged.

* Sometimes its like sending some of the changes to backstage where you don't wanna delete it but it must be kept safe away from main project then we use commands.
here we changed 2 commit so first we use 'add' command then 'stash' command

16. git stash.

Now, to bring all the stashed file back (i.e from backstage to front we use.

17. git stash pop.

Now you think the stash files are not important and you don't need them, then you can permanently remove them by

18 git stash clear

Now Learning Github:

First we create out github account, then create a repositary.

Now we want to add remote repositary to local repositary to do so we use

19 git remote add origin "URL" [here URL is the copied URL of your repo in Github]

- add here means adding a new URL.
- origin is the name of URL you are going to add.

Now we use

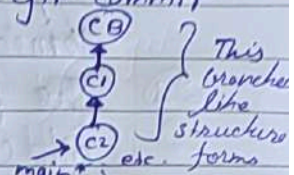
20 Git remote -v ; this will show the URL attached to this folder. (all the URLs).

21. git push origin master ; This will add your local file to remote repositaries. in other words it will push local file to remote repositaries

as you do this command it will ask your github password for Github OAuth Application (Git credential manager) with gist, repo, and workflow scopes was recently authorized to access your account.

After this your names.txt file will be added to your repo.

Branches ; as you add commits or write git commit command, new branches get added like



By default the name of this is called main previously it was known as 'master'

Proper Defination : In Git, "branches" refers to creating seperate line of development that can contain commits, changes, and files independent of the main branche (typically named "master" or "main")

Uses: Branches allow multiple developers to work on different features or fixes simultaneously without interfering with each other's work.

NOTE: main branch is the default branch which is used by people, we never commit on this as our code which is not completed or finalised must be created on seperate branch.

To create a new branch;
Suppose we working on a feature

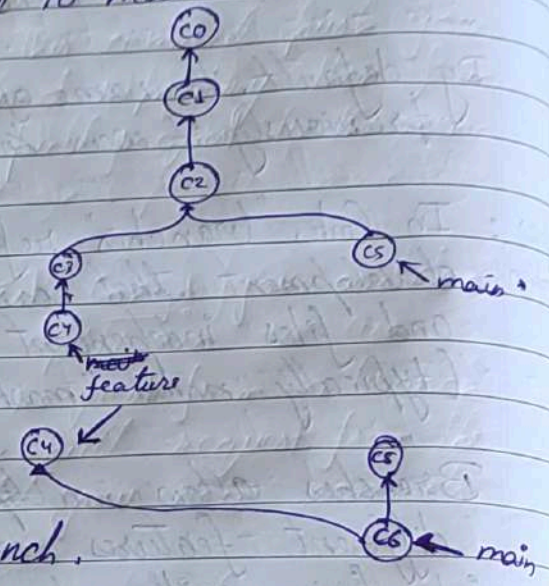
22. git branch feature ; will create a new branch called feature to work on specific feature

[I am learning branch concepts on "Learn Git branching" searched in google]

23. git check out feature ; after this command the star on the main command goes to feature branch, this means my Head pointing to the feature branch.

Head here means a kind of pointer that say the new commit will be made now in feature branch.

- Suppose you are working on your branch and someone added their code to main branch then the situation be like.
git checkout main
git commit



Now to merge your feature branch to main branch we use

24 git merge feature ;
This will merge your branch with main branch.

To add all your commit to master branch

Working with existing projects.

Let someone wants to contribute to any project How he should do ~~the~~ so, as being a random person who don't have access to main project can't change any file directly into the project.

So now what's the way to contribute to the code sample, so now you have to ~~copy~~ create a copy of this project into your own account. → (~~Is~~ That copy can be done by forking).
after this step, you have Deepansh-Deep / project name ; Copy the code and run the command

25 git clone https://github.com/Deepansh-Deep/project.git

NOTE: We can't directly change the project of anyone to do so we have to fork the project first.

The code which you forked from, and the URL you copied is known as upstream URL, now to use on it use command.

26. git remote add upstream "paste URL".

- after we do this command we can see 2 URL's
  origin : which is mine.
  upstream : from where you forked.

after forking use to clone command

→ git clone "URL" [from your account].

Now you can add changes.

★ Now after changes, we now wanna merge our changes to the main project main branch so we create a pull request.

How to do pull request.

① "First create a seperate Branch"; after forking the main project from other account to your main account, then copy the code, then make changes, after cloning the URL.

For changing from main branch to your custom branch we use.

27 git branch Deepansh; creates a new branch named Deepansh
28. git checkout Deepansh; main branch switched to Deepansh branch

Now here we change,
git add.
git commit -m "Deepansh added his details".

② To upload this changes, we run a command.

29. git push origin deepansh ; This command push your changes to your copied repository then from there you can give a pull request to the main project ✓

③ If they accept the request, Congo! you are done 31

NOTE: If you created one pull request from a branch you can't create another pull request from the same branch, it will add your new commits to the same branch.

That's why you should never commit on main branch.

So now when ~~your~~ you are working on different features or project you should create a different branches.

→ Suppose you added a new change to the ~~f~~ branch Deepansh and sent a pull request but now you wanna delete that request to do so we use

~~28~~. First we write "git log"; it will show all the commit.
Let suppose their are 3 commit to remove the Top most commit select the one below it and copy it, then write the command

30. git reset " paste the ~~one~~ commit code" ; this will stash your change.

Now with this do git add. ; it will go to stash ar
~~Tata~~ type to do type →

30.1 → git stash ;

after git stash ; your commit is removed and the change is removed.

Now, to change the new commit which you deleted you have to "force push" the change with the command

21. git push origin ~~Kunal~~ Deepansh -f.

Now, after running this, your latest commit gone.

★ Updating details

If we cannot change the ISTE contributors directly or any main account directly they also can't do it to my account,

which in short means fork is not updated.

2 ways to do it

① By sync fork.

② Manually.

↓

(i) git checkout main.

(ii) git fetch --all --prune ; here prune means the ones that are deleted will also be fetched.
Now, all the changes are fetched

(iii) git reset --hard upstream/main;

After this steps the forked project main branch is same as the orignal project's main branch.

(IV) git push origin main.

That's how we keep in sync with main project

The manually by which we did in 32 steps can be done by single command.

32. git pull upstream main ; git pull command internally does the same thing, its syncs your forked project to the orignal project's main branch

→ Here all the folders are in the local forder not in the upstream, to sync new upstream we use

33. git push origin main

Some new Concepts

Merge Conflicts and Squashing Commits :

- Pick and Squash :

, 1, 2, 3, 4 (Txt files)

Suppose we have 4 commits to merge all commits to a single commit, we use the command.

34. git rebase -i "paste"

here we paste the commit code above the one we need to merge.

→ you will get a dash board with no of commit (4) options

pick d9dd724 1
pick c9 69ec 2
pick 759d644 3
pick 673d4404

To squash; 2, 3 & 4 remove the pick in front and write s instead. this will merge the commits.

- Squash means whichever one is listed as pick merge it into previous commit.

- Merge Conflicts and how to resolve them.

Lets say I changed line 3 and someone else changed line 3 git will get confused whose change should be taken.

To do resolve this issue you have to overwrite the code if you want your change to be taken remove the someone's else change or vice versa, this will resolve the conflict.

BY

-Deepansh_Deep