Git :- It is a version control system.

They are tools that keeps a track of changes in our code.

→ Popular

→ Free and Open Source

→ Fast and Scalable

Can be used at any scale (High or Low)

How it works :-

i) Track the history → Let's say we ~~have~~ are on a project of making a website & we have completed it 80%. But we suddenly found that, we have done some

wrong things in last 20% and have to go back. But on big projects, doing manually such kinds of things is risky. So, git stores each us part of ~~our~~ our history and can take directly to ~~the~~ ~~that~~ 60% mark.

ii) Collaborate → If many people are working on different same project, so git ~~but~~ can give permissions different of access to kids of people, so that there will be no fault in our work.

→ Github: It is a website on which coders can upload their code and ~~see~~ ~~can track it~~ it will be stored and managed using git.

→ When a coder wants a job form company, he/she can give his/her "github id" to them to show the progress and projects on which he/she had done the work so far.

→ We make "folders" on that website on which we save our code, and these folders are called "repositories" or "repose"

→ We can also see other people's repositories on github and can copy their code as well.

→ We need a email (or college vala email) to sign up on git.

→ We can get some benefit through the college email id, but we should definetely make our own email id as well becz, college email id will not work after 4 years.

→ README file : It is a file which we can add with our repository, in which we can give different info. about our file such as why this repository is made, what are the codes present in the repository, etc.

→ README.md
↳ mark down

→ Any change on git ~~/ github~~ is a two step process, first → "add", then "commit".
On github, the add step is skipped and we can directly "commit"

→ "Commit" means taking a "screen shot" and saving it

→ Commit message is basically the name of our commit.

→ In ~~ReadMee~~ README.md :

If you write a line, it will show a +1.

If you then ~~delee~~ delete that ~~stuff~~ from the line and commit changes, it will show a +1 and a -1. +1 if for you have added an empty line and -1 is for that you have deleted the ~~stuff~~ from that line. If you again write something in that line, it will show a +1 and a -1. +1 is for that you have written some stuff in that line and -1 is for that you have the empty line is no more.

→ README.md is not a normal file, it is a special file in which to make changes, there is a special syntax similar to HTML.

→ for next line use
 tag

→ Anchor Tag : is used to add links to your page.

Eg:

→ Remote (Github) , Local (Laptop, PC)

Repository

↓

Process → [Cloning]

→ For cloning

Type :- `git clone < some-link of repose >`

→ `git add file name` → By using this command, one can track their file through git.

Caution : You need to be in that directory in which that file is present through your terminal.

Eg :- File is present in SHREEDHAR directory, so first do

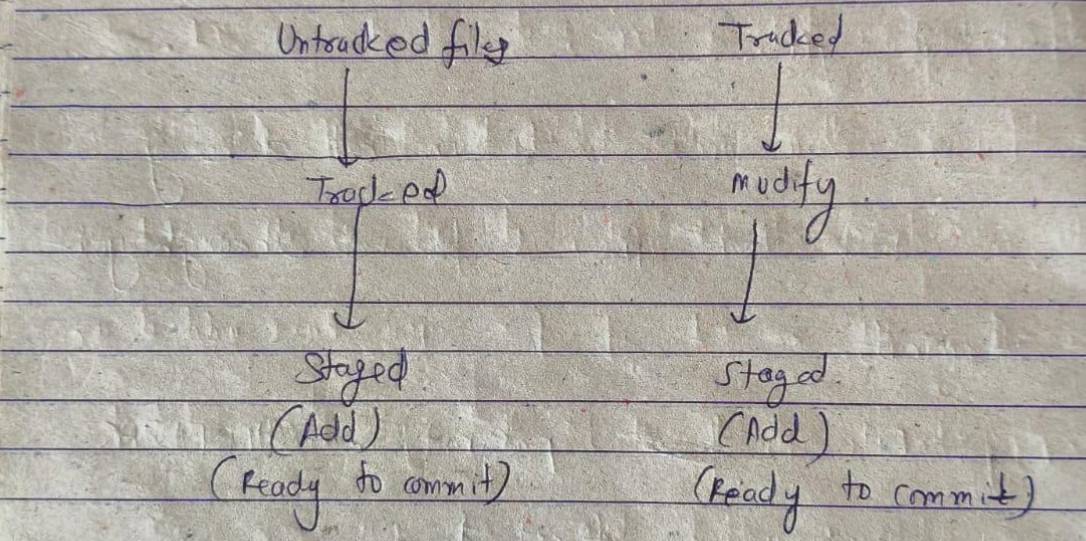
`cd SHREEDHAR.`

on terminal

→ To check status of files :

`git status`

→ What `git add file name` do ?



→ If you have to "add" more than one files

↓

`git add .`

→ To commit your commands

↓

```
git commit -m "Write some meaningful in
                this inverted commas".
```

Eg: You are adding some words.

If you have staged your file, now ready to commit

→ `git commit -m "Adding new words"`

→ These commits are done on just your local device, not on your github (online)

To publish them, use git publish command

→ Push command :- By this, you can push your changes from local device to remote directory.

```
git push origin main
```

(#) How to make directory & code on it & push it to Github online :-

(i) Make folder, make file, & code through VS CODE.

(ii) Then initialize git on the folder by first (cd Folder name), then

```
git init
```

iii) `git add .`

iv) `git commit -m "....."`

↓

These commit are now only to your local device not on GITHUB.

v) Make a new repose on GITHUB & copy link.

vi) On Terminal (VS Code) ~~git remote added~~

↳ `git remote add origin <link>`

vii) To check link right/wrong

↓

`git remote -v`

viii) `git branch`

~~ix) To change branch name → git branch~~

ix) To change branch name

↳ `git branch -M <New name>` //

main

x) `git push origin main` To finally push on GITHUB.

If we use `git push -u origin main`,

~~use this~~ every tym to change the code ~~& push you~~ ; or extend the code & push, you just have to write

`git push` //

→ Let's assume a scenario, in which you made a repose on github github and added a README.md file (or any other file).

Now you have written a code on your pc (git, not github) in some folder & tried attach it to git and then, ~~to~~ to push it github, it will give an error.

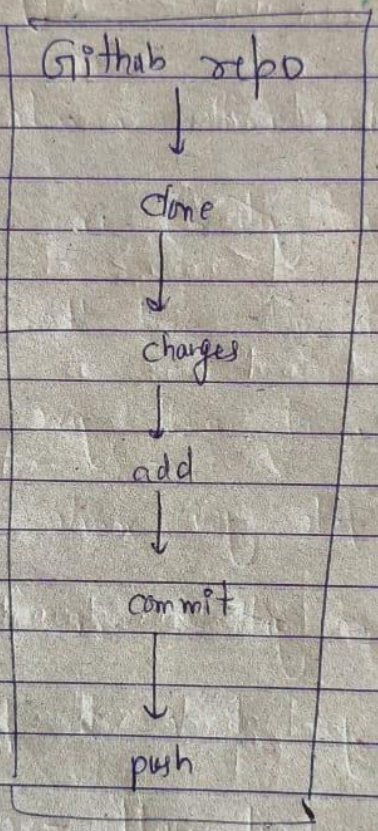
Becz the file you made on your pc & the file on github are unaware of each other.

So, first you have to pull the fill from github by `git pull -rebase origin main` command, & this is called rebasing.

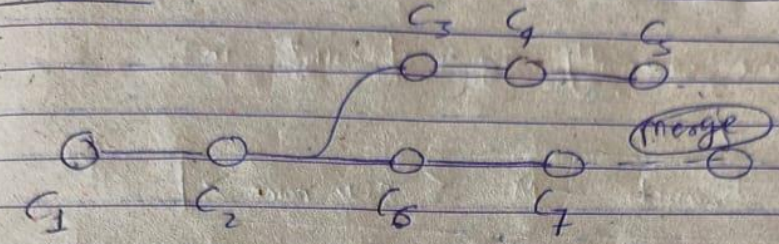
After doing this, your github's file will come on your pc, you can check it through `git log` command.

Now do, `git push origin main` command to push the files on github.

Workflow :-



Git Branches :-



C → Commit

⇒ In companies, developers work on the same program, so, to reduce tym, the make different branches on github & make commits. At the end, they merge all their Branches to main branch. By doing this, one do not have to wait for the other & hence, time gets reduced.

→ `git branch` → To check the branch you are currently present in.

→ `git branch -M <new name>`

↳ To give new name to any branch.

→ `git checkout -b <name>` (give)

↳ To create a new branch and enter into it.

→ `git checkout <branch name>`

↳ Switch to another branch.

→ `git branch -D <branch name>`

↳ To delete a branch

(But you cannot delete a branch in which you are present, first you have to go out of the branch to some other branch.)

→ If we have one file, and two branches are attached to it → "main", "feature1".
If we do some change on feature1, add it (git add .) and commit it (git commit -m "...."). Then we ~~commit~~ can see a new branch named "feature1" on our repository, with a "Compare & pull request".

→ Comparing two branches attached to same file:

Initially, file was same, but we made changes & but we attach it to diff. branches and do made changes in the branches named "main", "feature1".

Now if we are on ~~feat~~ ~~fe~~ git (local device) on "feature1" branch, to compare the differences with main, we can type `git diff main`. Then the portion written in green will be the content of "feature1". And the portion ~~wri~~ written in white/red will be of "main".

→ No, the changes are made on our github (both on "main" branch and "feature1" branch). But the changes (means merging) are not happen on our local device. To do that we have to use "pull command".

→ Full Command :- `git pull origin main`

This fetch and download our content from remote and immediately update it on our local repo.

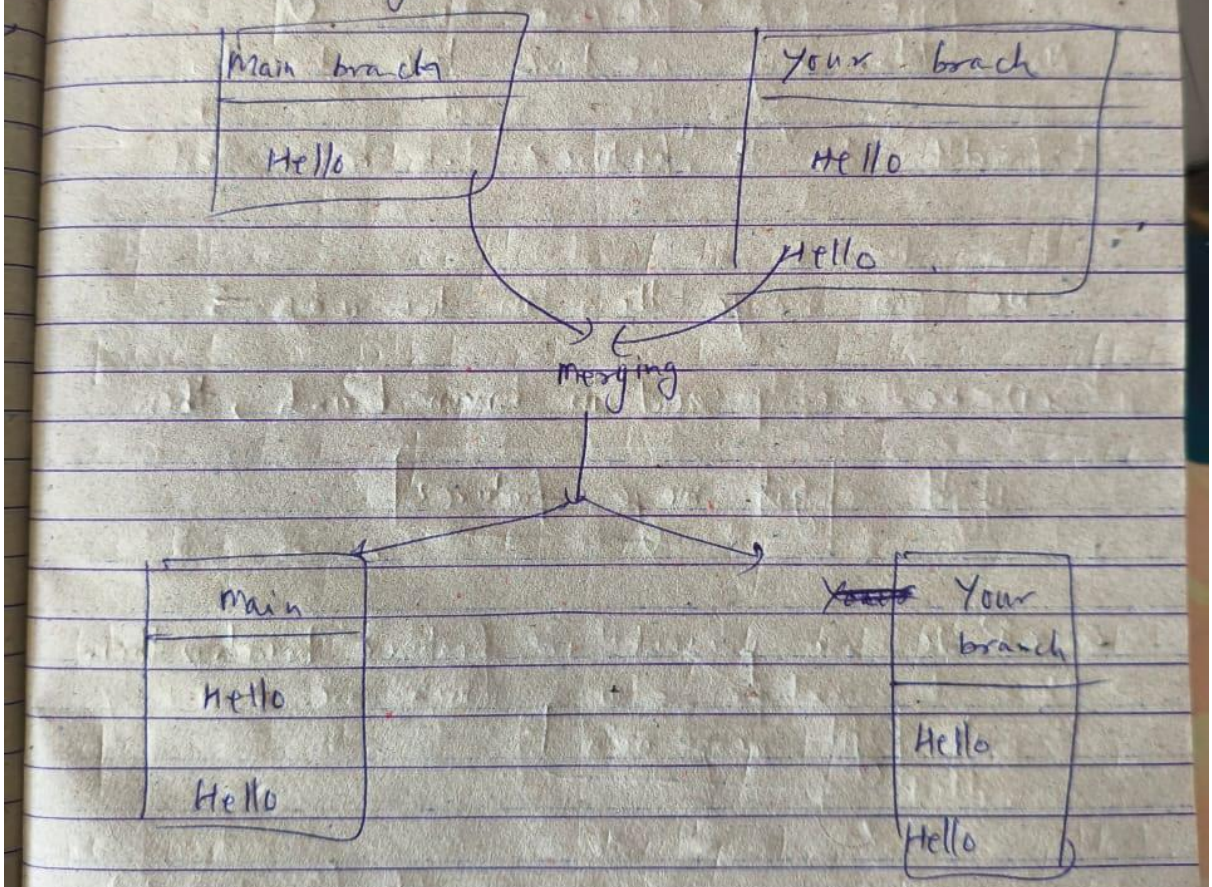
→ Clone	Pull
If there is some file present on github, not on our local device, & we here want that file on our local repo as well, so we use `git clone <link>`	If there is some file present on both our github & on our local device. And by some way, we changed that file on github, let's say by merging or by directly changing it on github. And we need that same change on local device, then we use :- `git pull origin <branch name>`

→ When we clone a github repo on local device, it will show us only the "main" branch.

→ When you do [git push origin <branch name>], and accept it on github, it will actually merge the two branches which is "main" & the one in which you are writing that code (on local device)
And it will have no impact on the other branches. Make sure it do not have a merge conflict. eg:- If in main branch it is written "Hello" & in your branch, it is written "He!lo", then it will end up in a merge conflict. Becz at same pt.,

there are two diffent letters, so, before merging it you should check these things.

You can merge this :



→ ~~If merging process, you should take~~

Another way of merging :- (By local device)

First take care of the merge conflicts

Now, lets say we have two branches named "main" & "feature2"

→ We can merge them in two ways :-

Case i) We are present in main branch, then write [git merge feature2]

Case ii) We are present in feature2 branch, then write [git merge main]

→ By doing this, the "main" branch & the "feature2" branch are now merged and having

same content on local device, not on github.

⇒ Now, we have to push them to github.

Write → `git push origin main`

and `git push origin feature2`

on doing these commands, it doesn't matter, "on which branch you are currently present in".

→ To delete a branch, you must be in some other branch first.

→ How to unstage changes which are already staged :-

i) For one file in a branch :-

`git reset <file name>`

(for this to do, you need to be in that branch as well)

ii) For all files in a branch :-

`git reset`

→ How to unstage changes which are already committed :-

i) a) You have made a change in one file (in a branch) and you have commited it, and now you want to make it unstaged.

→ `git reset HEAD~1`

b) You have made changes in two files (in same branch) and committed it at the "same time", and now you want to make these changes unstaged, then also you have to do → `git reset HEAD~1`, both changes in both files will be unstaged.

NOTE: HEAD basically represents the latest commit.

→ `git log` → by doing this you can actually see all the commits of your branch.

→ Deleting Multiple commits from VS Code :-

space space

`git reset -- hard <hash>`

By doing this, it will not show "unmodified" on VS Code.

→ fork :- It is used to copy repository from some other github account online. You can also do changes after copying it on your github account. And can do create a pull request & merge with the original github account's repo (if allowed by the owner of that repo).