Git & github:

Git ~~is~~ a ~~sof~~ version control software that enables ~~to~~ developers to 'save snapshots' of their project over time.

Github is a web-based platform where developers ~~can~~ ~~use~~ who use Git can connect and upload their work.

Git commands.

(Ex: Marriage → guests → photographer clicking picture of guests)

- git init
  (it'll create a .git directory).
- ls -a
  . .. .git
  (touch name.txt) (same as linux)
- git add name.txt → message
- git commit -m "name.txt is added"
  ↳ (Taking picture)
-

for adding something in file use:

vim name.txt
i → Sonali
esc → :wq

- git status (to check if something changed or not)

for redoing something already happened (like if a file's snapshot is already taken)

→ git restore --staged names.txt

// for checking history of all the commit

- git log

~~ify~~ if you want to delete the commits.
run.

copy the hashcode id
& run

- git reset paste the hashcode id.

(The commits / ~~files~~ delete will be in unstaged area)
(It's same like the picture clicked by the photographer is being deleted.)

(what if you don't want to delete those changes)

→ Hey, your photo is yet not being taken, go to the backstage and whenever we want you to come back we'll get you.

the thing is when you don't want to commit ~~delete~~ those changes (renamed, modified blablá) or loose but you want them to go backstage & whenever I want, I'll call you back
so for that we use.

- git stash

→ for bringing people from the back stage

- git stash pop.

- for deleting them from backstage run
- git stash ~~cle~~ clear

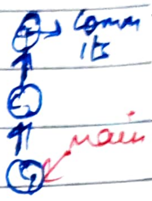
mean you're working with URLs
name of the URL is origin

for adding the link of your repository to your project

- git remote add origin paste_url.
- git remote -v (shows all the URL's attached with your project)

(showing)
for making changes in your repository

- git push origin your_branch_name
(branch looks like a acyclic graph



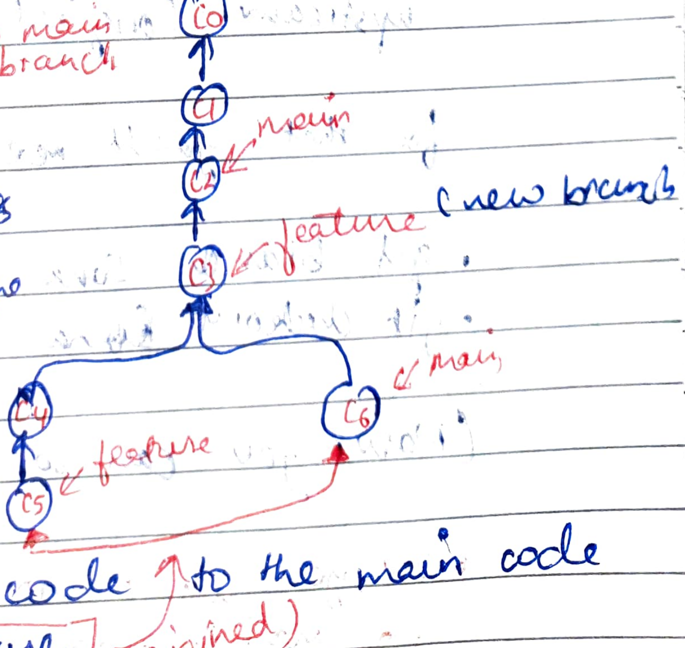
- git branch feature

NOTE whenever you're working with some other features / new feature or resolving a bug →
always create a seperate branch. [***

Head → It's just a pointer that says all the commits that you have made will be added on the head (the new branch you've created).

when someone else commit to main branch

- git checkout main
- git commit
(now your feature & main branch will go side by side)



- for merging your code to the main code
  - git merge feature (joined)

we use fork due we can't directly make changes to the original project of that organization as we don't have permission

for copying some existing organization project we use fork for that.

then clone that project in our repository.

- git clone paste_url_link.

→ for making changes in the original projects of other people we use pull request.

upstream URL → It is the URL from where you've forked the project.

- git remote add upstream url here

Now you want to make changes in the upstream (original project).

for that you'll need new branch.

- git branch Sona
- git checkout ~~Sona~~ (This will change the header)

(Now you can commit changes)

Now you want to merge your new branch to main's project main branch.
↳ This known as pull request

NOTE 1 Branch → 1 pull request

(cuz once one of your pull request is accepted then all the commits on that branch ~~where the~~ from which you sent pull req will be commited in the same so it'll be difficult to review your new commit ~~so~~)

So, for ~~evr~~ every new feature you're adding make a new pull request that means you've to make new branch every time.

if you by mistake made the new commit in the same pull request you can redo it by

- git log
- git reset copy-paste the hash ~~code~~ id before the last commit
- git status
- git add .
- git stash

(you're to force push the changes) → f

- git push origin sona $\boxed{-f}$

~~(The~~

(Now you want to make changes that you've previously made in the original project to be shown in your forked project too)

so, for that we use fetch

Step: 1

all the branches (the ones are deleted they'll also be fetched)

- git checkout main
- git fetch --all --prune

Step: 2

- reset the main branch of my origin to the main branch of upstream.

- git reset --hard upstream/main

- git push origin main

(That's how you keep in sync with the main branch)

Or you can use → (internally does the same thing as git fetch)

- git pull upstream main

- git push origin main

Or you can directly fetch upstream ~~from~~ in the github.

1 squashing commits

if you're a lot of commits you're working on and you want to merge that into one commit.

- git branch temp

• git checkout temp

~~ta~~ touch 1 ; git commit -m "1"
touch 2 ; " " "2"

• git log
, copy the last hash_id
• git rebase -i paste_the hash id.

(it will open something, there you'll get option to pick or squash (s)

so don't change the 1st line & change all other picks to → 's' as you want to merge all the commits into one

to exit use esc → :x → it'll ask you to write a message "commits merged" → :x

• git push origin temp.

11 merge conflicts & how to resolve them?

Suppose you made a change at line 3 & someone else also did the same so git will get confused (which change do git take?
git will ask you for help

So, it'll depend on the person who's taking the pull request to take what changes to ~~wat~~ take.