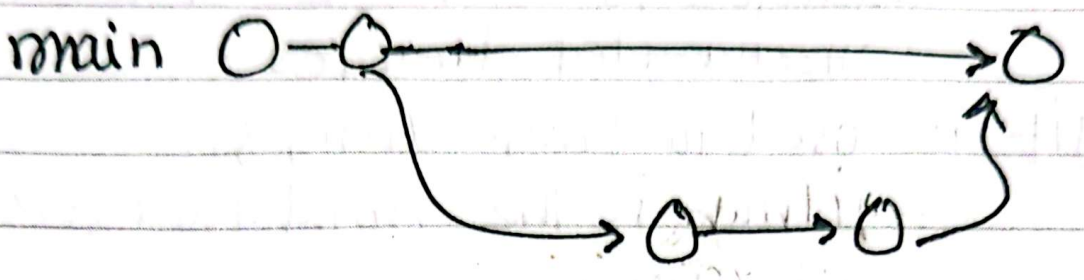
git — commit history / Branches / remote repos.



Branches - parallel version of your project

git status —

git add <file> // added to git

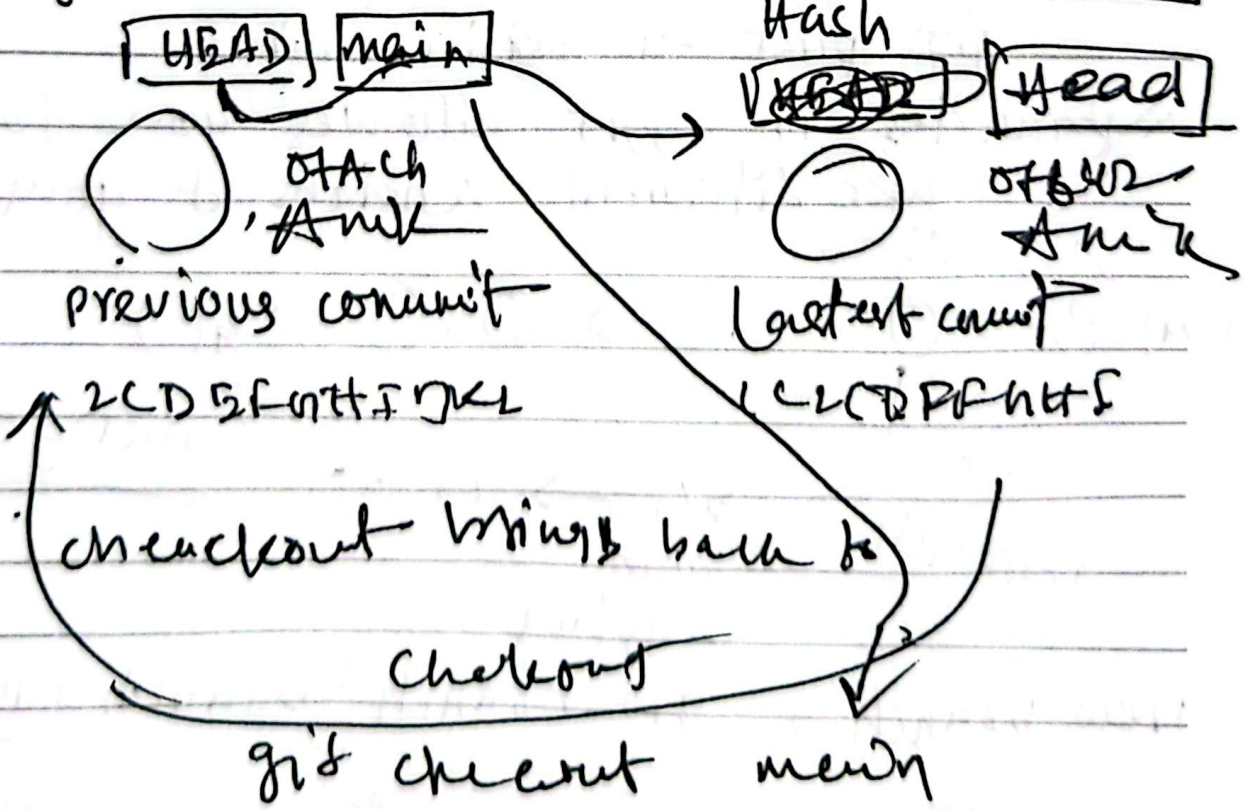
git commit -m " " commit means a snapshot and copy file,

git add . / added all

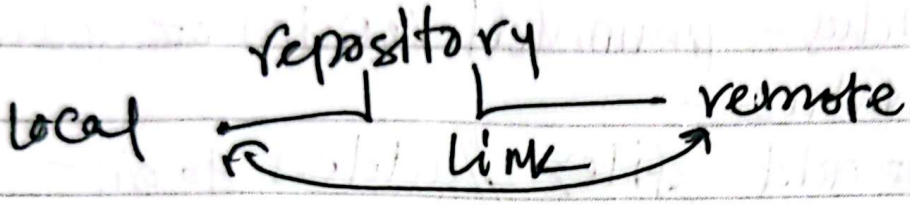
git log gives the history

// switching to older commit and restoring it

git checkout " "

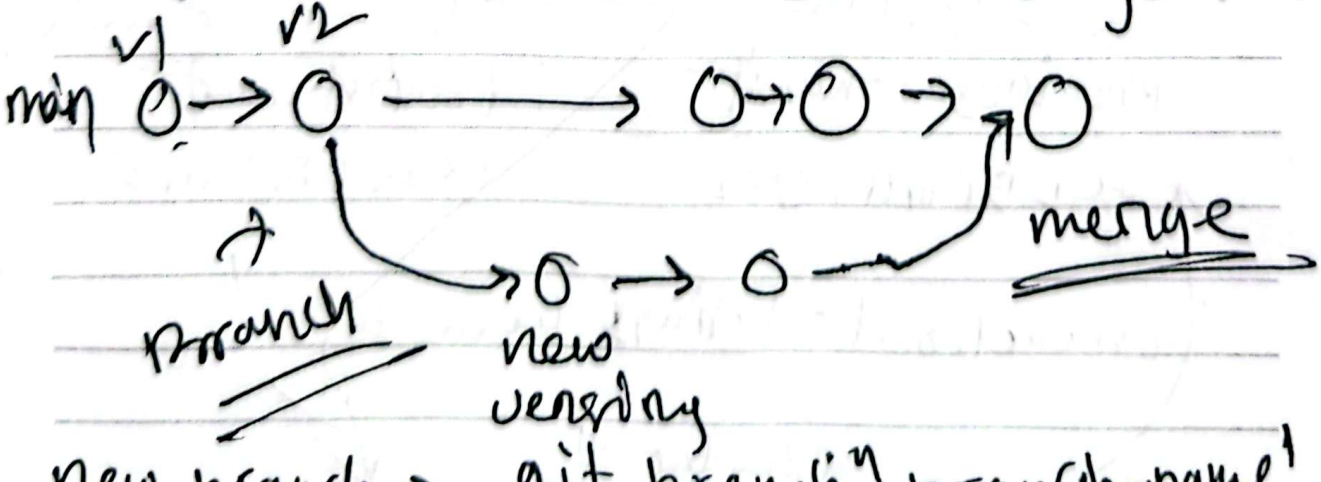


git checkout + main
changes to main even without uncommited changes.
git is used to track changes
- github is the cloud that stores git repos.



git init // initialize local repo
clone repo from github. remote is renamed by git as origin.
Linking remote with local-
git branch -m main
git remote add origin "_____"

git push -u origin main
Branches in git allowes you to make different versions of your pro



new branch → git branch "branch-name"

immediate move to need branch | git ~~branch~~ -b
git checkout -b feature-name

creating new branch, importance on Head - position.

creating from specific branch | git branch new-branch source-branch

commits written in imperative → 1. essential 2. Express a command

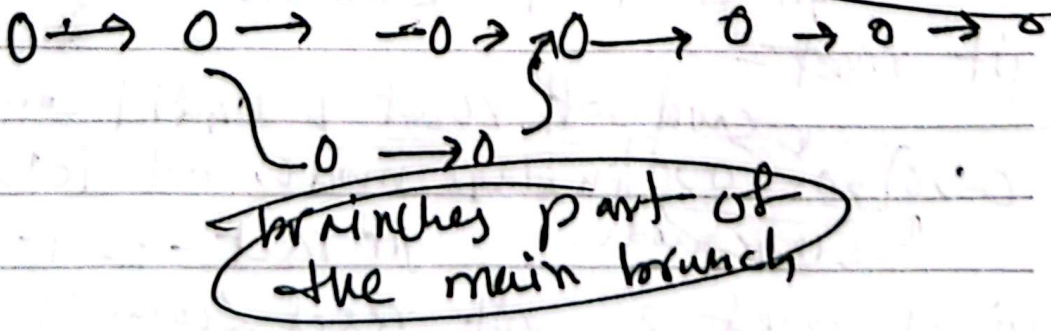
commited - now push to remote →

publish local branch

git push --set-upstream origin _____
b. name
git push -u origin _____
b. name

after that git push, git pull

pull request | merging | same changes with teams for review and approve



branches part of the main branch

used the github, pull request.

git pull – remote to local repo.
↳ git pull origin main.

typical workflow – ① clone repo → ② create new branch from the main – ③ make changes ④ push to remote repo ⑤ open a pull request – ⑦ merge the change ⑧ pull merged changes to local repo ⑨ repat from ②.

Resolving merge conflict →

① ↳ Checkout to the main branch
② go to your branch and merge your branch with main branch so there will be conflict, use ide to chose what remains and what deletes.

git saviour commands –
git checkout <commit-hash>

git reset –

soft reset – unddid commits are in stage mode, not deleted.
git reset --soft <c. hash>

mixed reset → git reset <commit hash>
stage.
working directory

git reset --hard <c-hash>

Hard reset - you know

git revert

Undo ~~[illegible]~~ without losing commit history

git stash

git stash // saving in directory without committing

git stash list // get the list of stash

git stash apply stash@{0}; (stash name)

Version control

recording of all files and it's changes so specific versions can be called later.