GIT & GITHUB

git is a popular Version Control System.

∴ practice of tracking and Managing changes to Software Code.

Repository → folder where git tracks your project and its history.

Clone → to local Machine

git (working)

folder → initialize git → Repository

Repository → files updated → Select Modified files to Stage

Select Modified files to Stage → Staged files are Commited → Save a Snapshot

Staged files are Commited → Track the changes

→ Git Allows to see full history of every Commit.

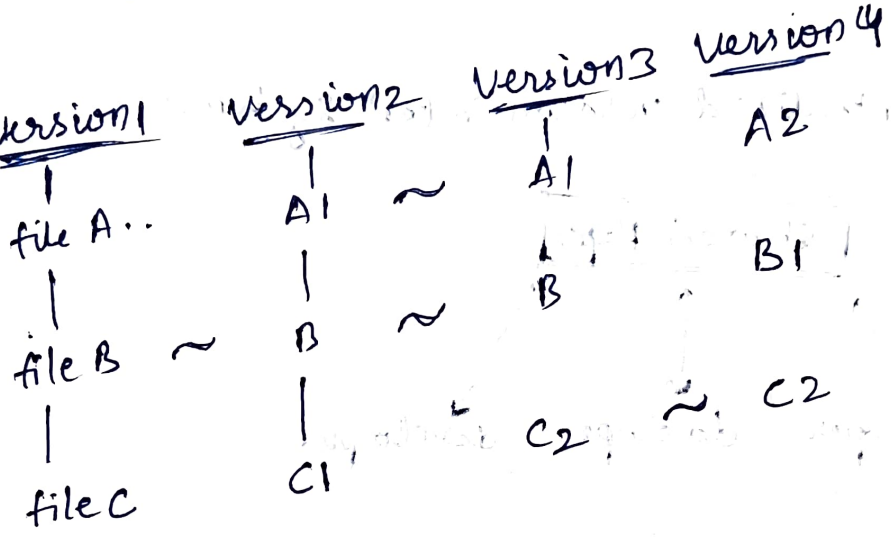
→ Convenient back to previous Commit

1. Install git
2. Windows → (git Bash) → terminal

⇒ ls → show all the folders/files in that directory

⇒ git --version

git (every time we commit, it saves a snapshot of what all files look like at that Moment and store the reference to that snapshot) if the file is not changed it just links the same reference.



GIT Configuration

`git config --global --list` → lists your username & user email

Set username & email

```
git config --global user.name "ashish"
git config --global user.email "ashish@gm"
```

① Create a directory in local Machine

② Commands:

⇒ mkdir projects → cmd/P.S

⇒ ren oldName newName → cmd.

Rename-Item oldName newName → PS.

⇒ move projects D:\ → cd.

Move-Item projects D:\ → PS

⇒ rmdir projects → removes directory. (empty folders)

rmdir /s /q projects → remove directory (with contents)

⇒ Powershell:-

Remove-Item projects

Remove-Item -Recurse -Force projects

⇒ cd → current directory.

⇒ dir → list all directory / files.

① ⇒ git status :- fatal: not a git Repository.

② ⇒ git init (directory → repository) (creates .git folder)

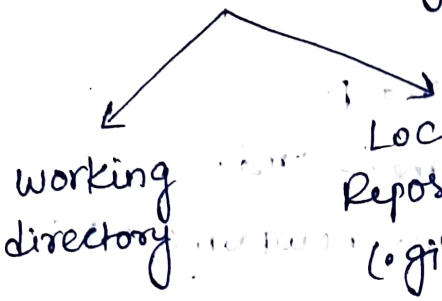
⇒ git status:- On branch Master.

No commits yet.

by default on master branch.

2a git init -b main → On Main branch

Project Directory



Add New file → PS: New-Item capital.txt

CMD: type nul > file.txt

remove file → P.S: Remove-Item Capital.txt
(rm →)

CMD: del file.txt

SHOW CONTENT in File

type Capitals.txt → CMD

Get-Content Capitals.txt → P.S

cat README.md → Git Bash

now Staging & Committing

③ git add capitals.txt → (staged in local repo)
→ to be commited.

git rm --cached capitals.txt → unstage
No need ☺

④ git log → All the Commits List.

⑤ git commit -m "My first commit"
↳ commits the staged changes

after commiting it assigns a (checkSUM) a hexadecimal code to the commit to. maintain Integrity.
(yochan)

git log → show all the commits.
with data (Config)

Author : user.name, user.email
Date / time
Message

Now if we Make any change to our code in working directory, and enter git status.
(It shows changes not stage for Commit)

modified : capitals.txt

git add capitals.txt

git Commit. -m "This is my second Commit"

(6) git commit -a -m "third commi"

↳ we can directly commit without staging.

(7) git diff → changes made but not staged.

(8) git diff --staged → changes made but no changes. Stages but not Commited

(9) git add . → Stage all ~~files~~ change in the repo

(we can remove a file in local repo)

(10) git rm --cached file Name → removes

↳ removes a file from git tracking, but keeps it on your computer.

If we remove file directly from working directory (and if it is already commited).

(The file will still be in local repo)

↳ Instead use this and then delete

Need for Remote Repository:

- → for collaboration
- → Public database
- → portfolio building
- → open source Contribution

GIT HUB

↓

Its a web based platform built on top of GIT (distributed version control System)

① Create repository in git HUB

assigns a link in github for repo

→ Pushing files to Github

```
mkdir DSA_JOURNEY
cd DSA_JOURNEY.
echo " DSA JOURNEY PROBLEMS" >> README.md.
Cat README.md.
git init ——→ (by default Master)
git add README.md (staging)
git Commit README.md -m "First Commit"
```

⑪ `git branch -M main` → assing which branch (to Push) files

⑫ `git remote add origin (link)`

⑬ `git remote -v` → Verify

origin ↓ like a Nickname for link repo

This command links local repo to Github Repo

⑭ `git push -u origin main`

Next time when we want to push files.

```
git push origin main
```

GIT TAGS

↳ Tags are perpenant / unchangeble pointers to specific commits in the projects history. Important in case of Version Releases. V1.0, V2.0 ...)

→ Unlike branches which move forward as code grows, Tags saves snapshot of codebase --

```
git tag V1.1 -m "27th June release".
```

√ `git tag` → shows all tags.

we Need to push Tags seperately to remote Repo

```
git push origin V1.1
```

tag name

↳ Can show tags in GITHUB

Branching in Git -> Branch is a just a Pointer to a Commit

`git checkout -b feature1` → Creates and switches to a new branch

`git branch feature1` → creates branch

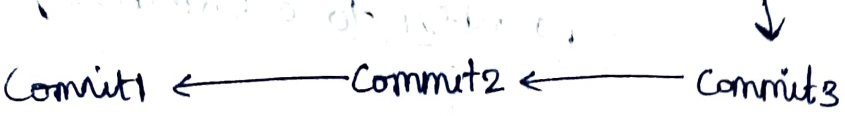
`git branch` → Shows all branches in dir & (* for active branch)

`git switch (or) checkout main` → Switches
(main = branch Name)

`git branch -d feature2` → deletes branch
(feature2 = branch Name)

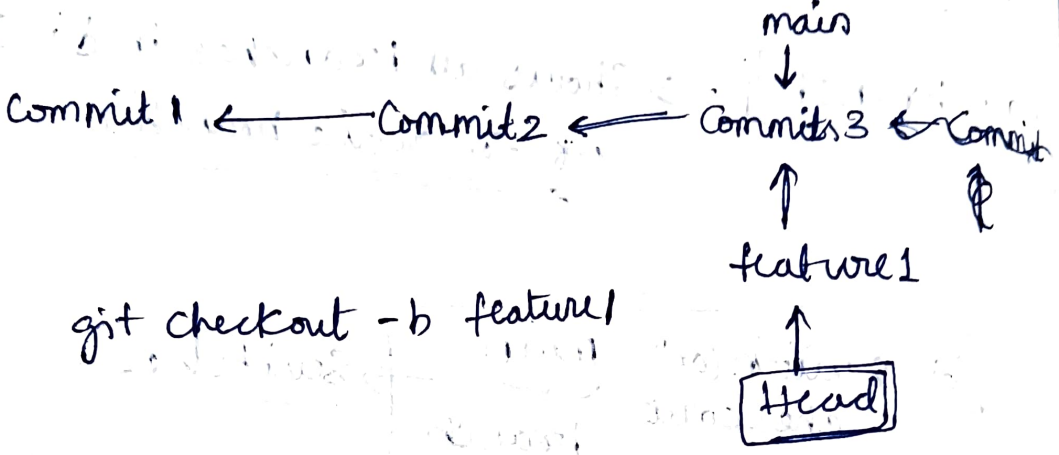
`git push origin feature1` → Pushes to github repo
(feature1 = branch name)

once a file added to a branch in local repo, it only appears when you switch to that branch.



if a new branch is added

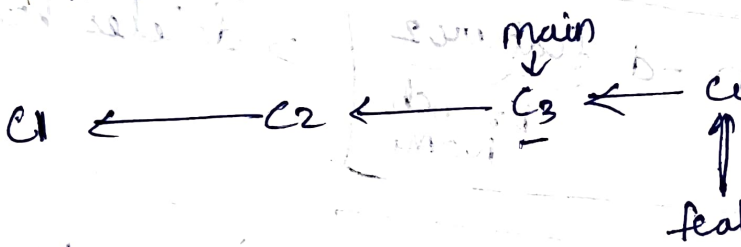
(2)



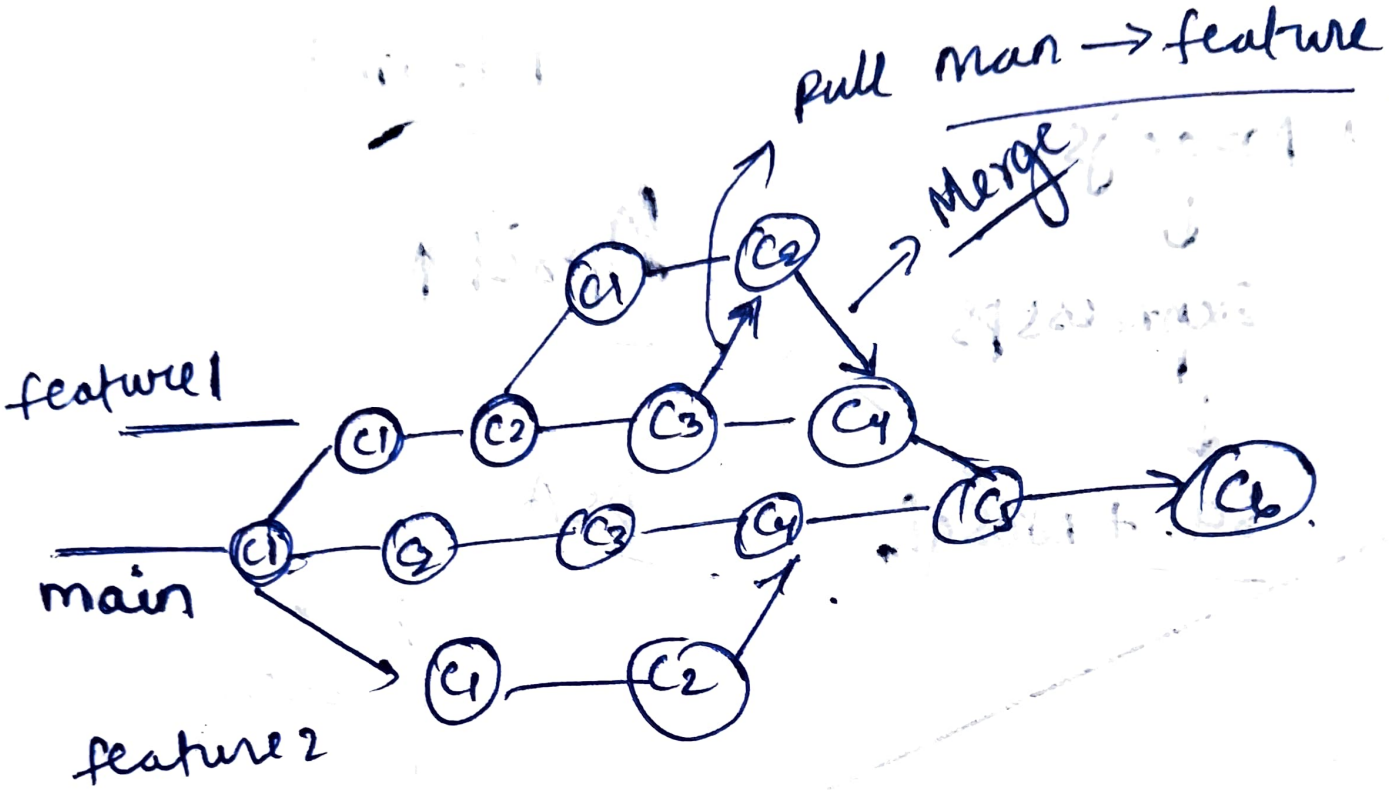
git checkout -b feature1

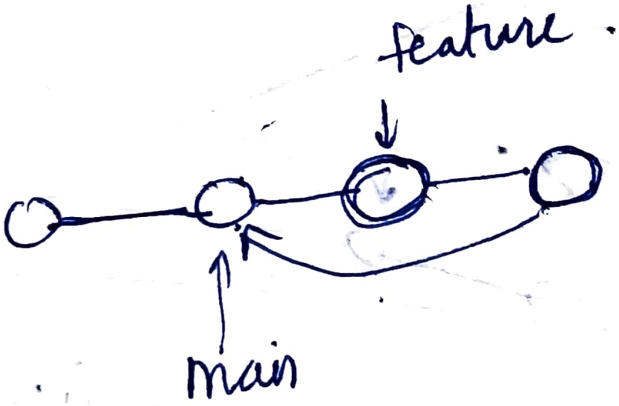
(3)

New commit added in feature1 branch.



→ git commits only store what has changed.


Pull main → feature
Merge
feature1
C1
C2
C3
C4
C5
C6
main
C1
C2
C3
C4
C1
C2
feature2


feature
main