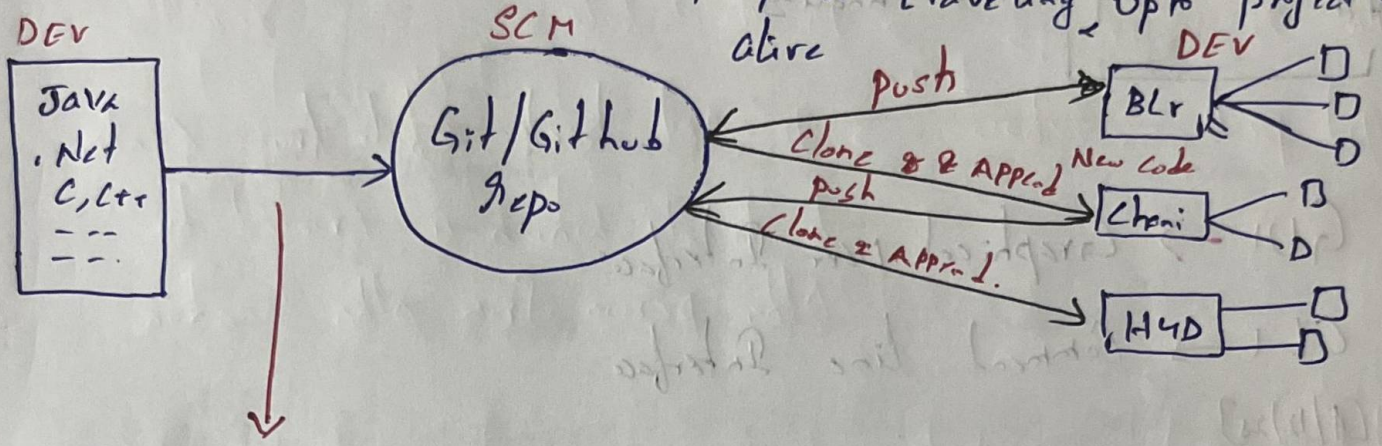
3/12/24
~~Monday~~
TUESDAY

Day-1

GIT & GIT HUB

→ Git and Git hub are in Source Code Management System / Version Control System. [SCM / VCS]

→ Code will travel from 1 one location to other it will keep on traveling upto project is alive [Updating]



After developer develop the [Repository] code it will sent to a Centeral Repository called GitHub.

→ We need to track what is going on, what was happening to know that we need to track, we need to know tracking ID, In Git tracking ID is known as

Commit ID

For this we need to download Git bash

Git

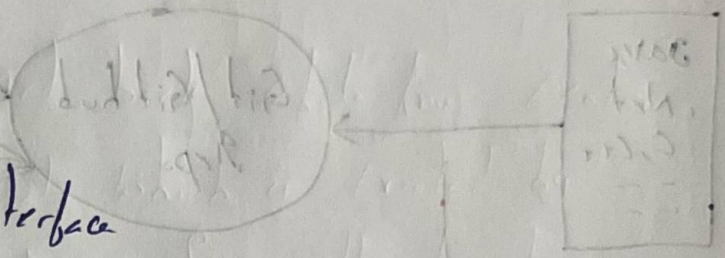
① It is a Software

② In this developer will develop code and Move code in 3 phases and Commit it

Git hub

→ Then it will be in Github which is an Central account/ web base platform which will Extend Git functionality

Git → Global Information tracker



GUI → Graphical User Interface

CLI → Command line Interface

Day-2 [4/12/25]
WED

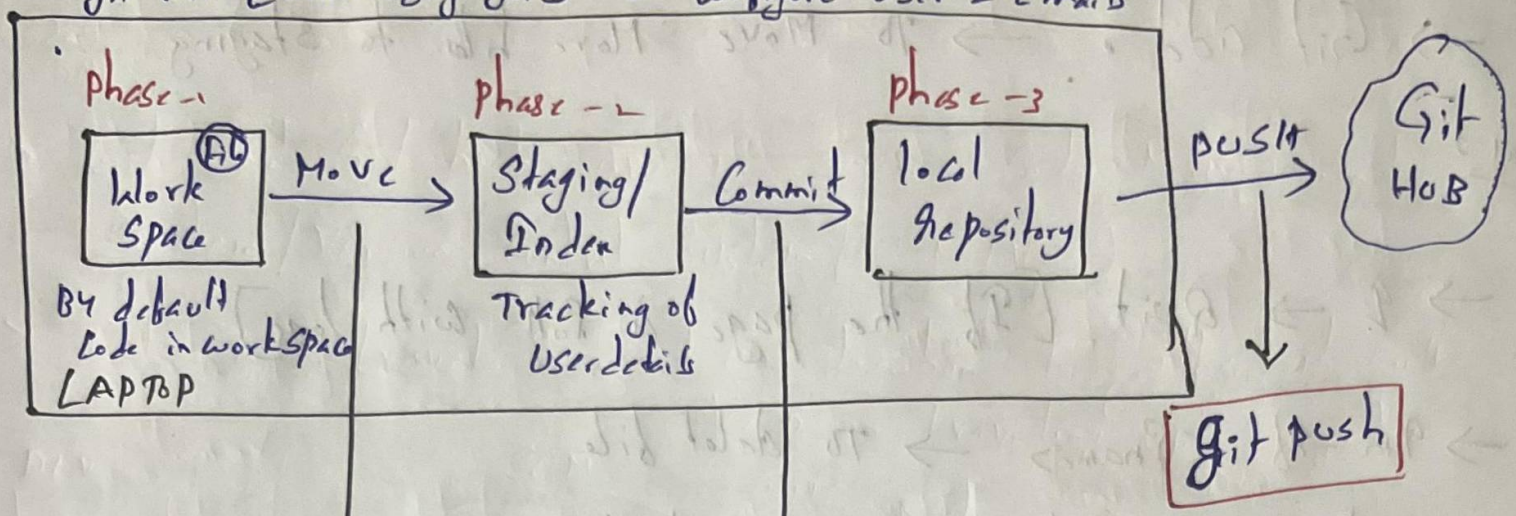
⇒ Initializing Empty Git Repository

→ Code will Move in 3 phases

* Git Init → Initializing Empty git in lap top [Hidden] .git
* git Config --global user.name "Name" } Configuring
  git Config --global user.email "Mail" } auther details, user details, Emails

By this in Case of change code the auther Can Commit with that Mail ID

git init [Initializing git] & Configure User & Emails



git add fname

git Commit -m "msg" fname

To check Status → git Status

if file in workspace it will show as Un tracked Status.

To check auther ID, time stamp when Commited

git log | git log --online

To check what files Commited → git show <Commit ID>

Note:- Need to login Github account

→ Before pushing Code to Github we need create a Repo ~~[already created]~~ in Git hub and need to clone the Code then move in 3 phases & push to Github → In real time project Clone Central Repository. → ~~git clone~~ git clone <URL>

↓

https lin

Note:- No need of Gitinit as we are working on Central Repository. Gitinit → local

→ Git add . → To Move More file to Staging

Day-3

→ q → Quit [If the page is full with logs]

→ rm -rf <fname> → To delet file.

→ git pull --rebase → To get back a file from center to local which is created by other person from other location.

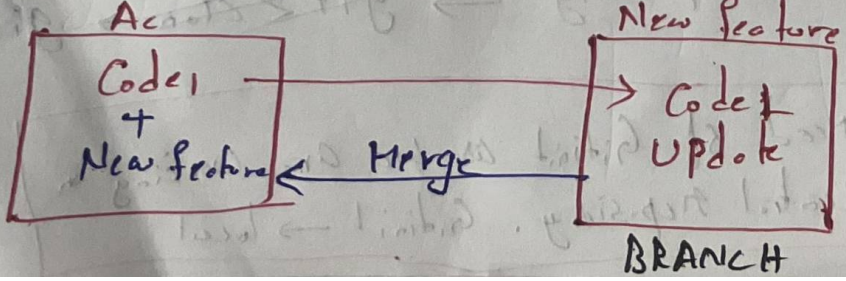
> Note :- Before pushing a code we need to check any Extra file is there in Center which is not there in your local then we need to pull that into our local before pushing.

Day-4

Branch :- Branch is a copy of data / Branch is for seperation

Eg :- A dev want to update Audio call code so he will write the code ~~in a Separate~~ Separtly Nothing but branch.

After testing and all the code will Merge in Main / Master which is Main code [for Prod deployment]



What is Branching Strategy:- Branch is a Copy of code

Branch is for Separation, Why because we wont work on Main branch, If any issue created on code the code will corrupt. So that we will separatly create a folder/branch for new feature then after after passing all tests we will Merge in Main branch.

For any release of code we copy the code to release branch and we will release code.

If any issue in release branch the we can create an Hot fix branch then we will work on that and then send to release branch and Master/Main branch.

→ git branch — To know status of branches

→ git branch <br.name> — To create new branch

→ git checkout <br.name> — To Enter into new branch,
→ git checkout -b <br.name> — Create New branch & checkout ata
→ git merge <br.name> — Merging feature branch data to Master [It can be done from Master]

→ git branch -d <br.name> — To delet branch. [For local]
-D → force full delete if not Merged.

→ git push <branch> — To push branch

→ git push <URL> <br.name>

git push origin <br.name>

→ git branch -D <br.name> — To delet forcefully

After this it wont update in Remote so for that we use

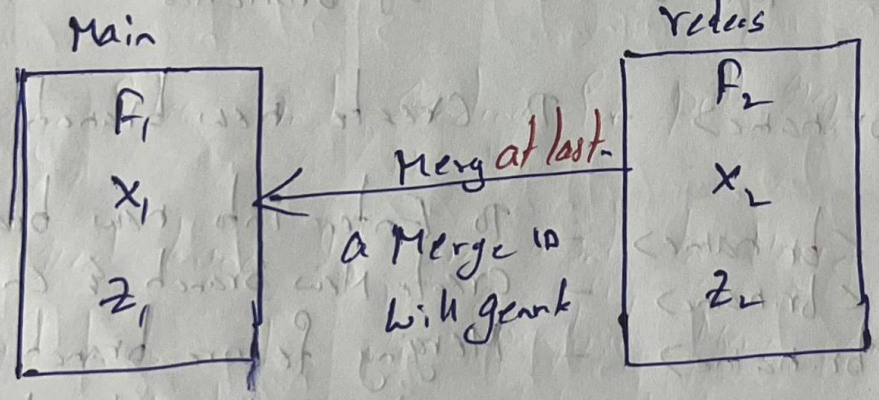
git push origin -d <br.names> [To delet remotly

git remote prune origin → To delet branch in local.

Day-05 Linux File Editor

Real time Senario

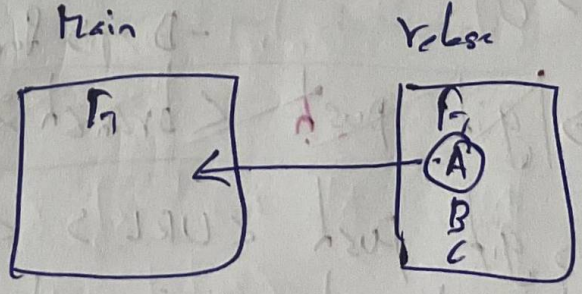
If files are traveling in 2 indipendat branches which are partlelly moving then we want to Merge these at last then there will popup in vi editer we need to save that, then one merge ID will genrate



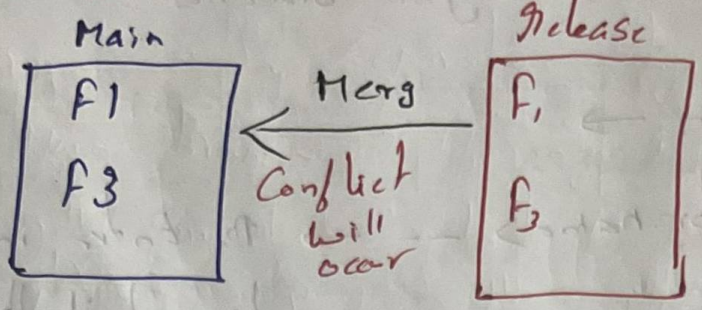
Real time Senario

If want to Merg a Specific File [Commit ID]

git cherry-pick <Commit ID>



⇒ Git Conflict



If 2 developer working on same file with diff content — Conflict

diffrent

If we have Same file in 2 branches ~~then~~ ~~If~~ ~~doc~~ and they are having diffrent code then we are trying to merge the branches Conflict will appear

Then dev need to sort out which is final.

But both branches content will apper in that file with an seperation

After getting conflict

vim <filename> → Edit → git add . → git commit

~~git~~ ← ~~git status~~

by this conflict will removed.

⇒ git branch -r → to show remote branches which are in origin

HEAD is the origin to current branch.

⇒ git branch -- → origin branches

⇒ ~~git <br.name> --~~ →

⇒ git checkout <br.name> -- → To Enter into that branch if there is a file with same name.

⇒ Pull request / Merge request → By pull request we can Merge 2 branches directly in git hub [If Conflict occurs it won't Merge]

Day-6

~~Day-6~~

⇒ Creating branch directly in Git Hub account → Branches → New branch

⇒ Merging files and sending for pull request → office scenario.

In real time If we want to Merge 2 files directly from git hub then we need to Merge and get approval from our higher officials like lead, Manager by pull request.

Then a request will be sent to them the review and allow you to Merge.

⇒ Open Source → Free lancer

While working as a free lancer (Open Source) we cannot upload directly files to a particular branch that we don't have access. So for this we nee need to fork the Main branch [i.e duplicate of Main branch will be in your Repo].

If we get an Expected result we need to propose with that project to work with the and Merge with them

A duplicate of Main branch with ~~dir~~ get in our Repo will all files. If we want to Merge our files we need to ~~get~~ Send pull request

Day-7 [11/12/24]

⇒ Protecting Main branch

Settings → Branches → Add classic protection rule

↓

Protect matching branch ← Branch name pattern [Name] [Click on checkbox accordingly]

⇒ Filtering :-

→ git log -n → Last 'n' commits [where n = 1, 2, 3, 4 - -]

→ git log -n --oneline → last 'n' commit id's in one line

→ git log --author "Padmakar" → Commit id's done by a particular author

→ git log --Since "10-12-24" → Fetching logs from a particular date

→ git log --Since "11-12-24" --Since "1"

→ git log --Since "12-10-24" --author "Padmakar" → Fetching log from a date with some author
MM-DD-YYYY

⇒ Alias [Configuring shortcut name for git command]

→ git config --global alias.sts "Status" →

Config Status as Sts.

⇒ git config --global alias.l "log"

⇒ git config --list → to see all the configurations

⇒ git config --global --unset alias.<name> → to delet alias

⇒ Revert

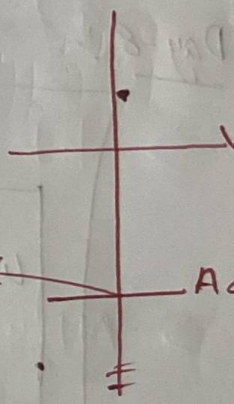
⇒ git revert <last commit ID> → If any application colapsed by last updat the we can use this command and revert.

Tag :- ~~Definition~~ If developers develop a partiular featured application then they will give a label to that commit

DO nothing but tag:

To which the Head is pointing to the tag name also will point

Head → Main , tag: <tag name>



→ git tag <tag name>

→ with help of Commit id we can tag

→ git tag <tag name> <Commit ID>

→ git checkout <t.name> → to check out to that tag.

→ ~~git tag~~ git push origin <t.name> → to push.

→ git push origin -d <t.name> → to delet remotly

→ git tag -d <t.name> → to delet locally.

⇒ To Edit Commit message by using amend

~~git commit --amend~~

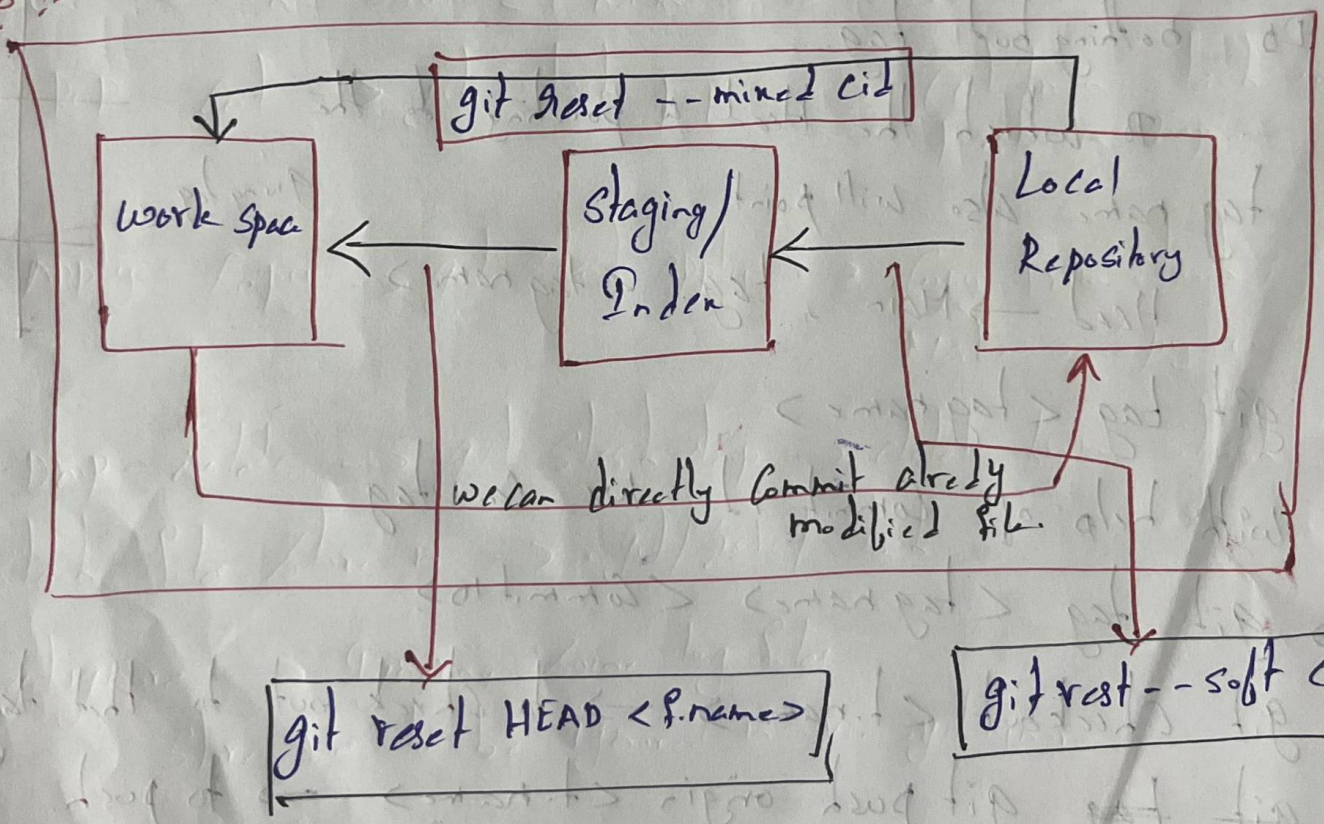
git commit --amend -m <latest commit msg>

If 2 files have different commit id and we want to give same commit ID.

git commit --amend -m "<Commit ID with comit no>"

We cannot amend old commit ID

Day-8 :-



[id should not only reset it will work for older [n-1]

git commit -am "msg" ⟶ will add & Commit

⇒ git Merge

① To intigrate the changes b/w the branches

Git Rebase

① To pull most recent changes from remote to local.

Git Rebase

When if 2 individual branches travelling parllelly with diffrent files then we want to Merge & Merge ID will create, in case of Rebase one branch of Head pointer will Rebase to another branch.

→ When Ever we need log history then we need to Merge

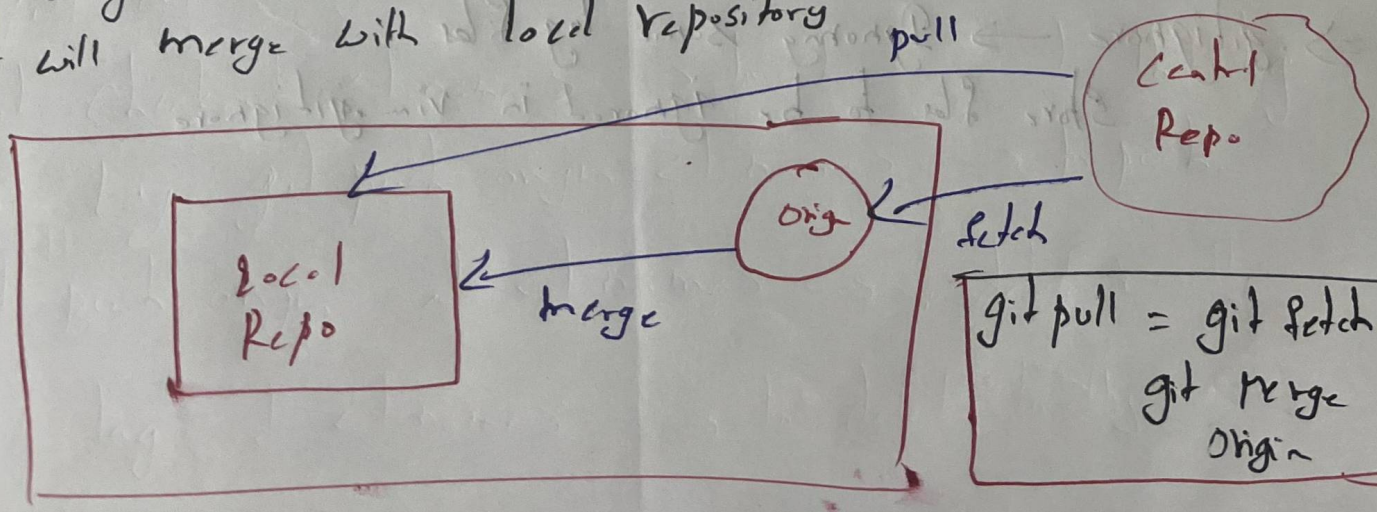
If we need to change / add files from one branch to other then Rebase.

We can use git pull for Merging.

~~If we give git pull~~, ~~then~~ There will origine in b/w local repository and central repository

So if we give git pull / pull --rebase ~~if not~~ il

The Origine will fetch code from central repository then it will merge with local repository



Stash :- It is a temperary storage area.

If we are working on one task suddenly we got anther task need to do but task 1 is 50% done then we can store task one in stash [task will come out of git life cycle] Complet task 2 and bring back task one & do

⇒ git stash list → To see stash list

→ git stash save <f.name> to save file in stash.

→ Pop - cut → git stash pop → Cut from stash and Paste in rep.

→ Apply - Copy → git stash apply → Copy from stash & Make a copy in stash

→ Drop - delete → git stash drop → delete from stash

⇒ Stash files store in → .git/refs/stash.

⇒ .gitignore → Ignoring some log files
store files to be ignored in vim .gitignore.

Version Control System [VCS]

Central version Control system [SVN] [CVCS]

SVN - Sob Version

→ No local Repository

→ It has Central Repo

→ Users will Commit directly to Center. If central Repo is down then users wont get code. [No back up] So application will go down

→ Internet Requird

Distributed version Control system [Git] [DVCS]

→ It includes Remote repo. as well as local Repo.

→ It has Central rep.

→ In this User will clone Central Repo, so user get an Repo in their local, so they will Commit in local after user work complete then push on Center.

→ Internet Requird for only while clone, pull, push.