Git - version control

- GitHub
- bitbucket

both
- Theoritical
- Practical

Website git-scm.com

Git - Software itself.
GitHub - service provider

* Terminal (Warp)
Command line

→ language independent (only Text language) we are learning here.

→ A talk on git

- git & github are different

git → software
github → service → as like a dB (database)

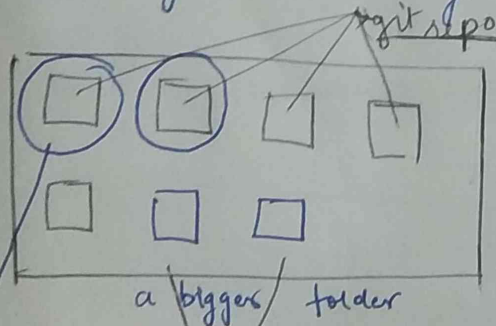
Git- Version Control

→ version control System
(• tracks files for changes)

→ Learning Path
- get the basics
- Use it daily
- face the problem → solve the problem.

Terminologies

1) Repo | Repository | folder
↳ Git on system Vs tracking (Repo)



.git repo (these are tracked by git software)

containing small folders
→ all the folders are not tracked

tracking of each folder is independent

not tracked

- only whose where you have mentioned git are tracked

→ if you have exclusively mentioned not to track a folder it will not be tracked.

Command

1) git --version
(gives the version of git you are using)

How to track git folder

- → git init (initialize the git) (can be used one-time per project)
- → git status (to check the status of git software)
- → .git is hidden folder to keep history of all the files or subfiles)
- * never-ever try to make changes in this manually.

1) Check git status
Cmd 2) git status
o/p: fatal error
2) enter the folder
Cmd - cd gitone / foldername
Cmd 3) again git status (make it a habit use this cmd often)
4) initialize the git
Cmd - 3) git init (it initialize the empty repo
5) Cmd - git status
- now, there is no fatal error
- instead - it shows nothing to commit

this means our folder gitone is being tracked now

Cmd- line commands

- → cd (enter a folder)
- → ls (list of all the items in the folder) (Get-ChildItem)
- → ls - la (also list the hidden item in the folder) (Get-ChildItem -Force)
- → cd .. (go to the one above folder)
- → pwd (present working directory)
- → mkdir (to create a folder)

Commit

(checkpoint like levels in game)

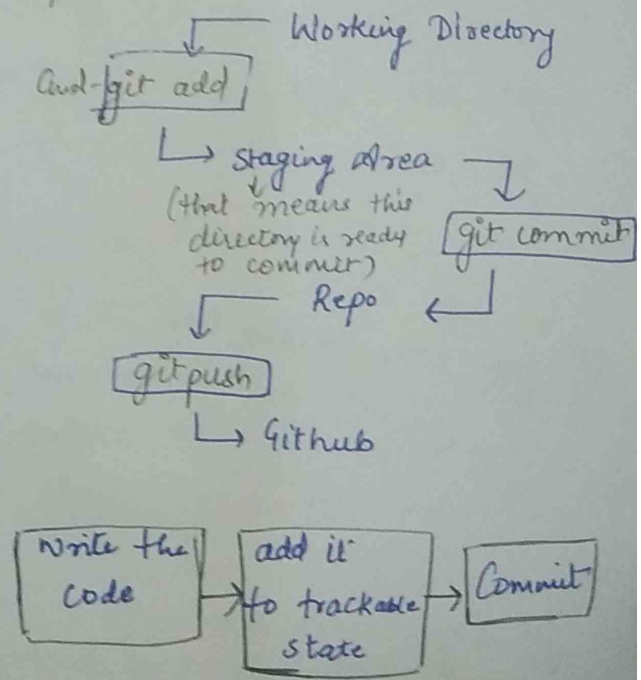
~~Staging Area~~

Cmd- line Commands

touch - to create new files in a folder

* To check if it is untrackable/ not in VS Code - file name will have (U) at the right side which means the file is untrackable to the git

Steps to do Commit



Cmd

4) git add → it will make the file trackable

git add filename ____ ____

multiple files can be added at once.

→ this add the file to the staging area.
(to be commit format)

* this is the part where we can add the same btn (to move together) or where we can choose not to commit and make it unstage.

* git add .
↑
this dot will add all the files.

5) git commit
(commit always needs a message)

we should not directly use git commit instead git commit -m
↑
message

should be used

Eg: git commit -m "add file one"

Stage
→ git init
create files / file
git add file1 / file2 || git add.
git status

Commit
→ git commit -m "a good msg"
git status
~~Repeat 2 or 3 times~~

6) git log
it gives the details like
→ Commit Id
→ Author: who made the commit
→ Date: on which commit is made

git log --oneline
(it gives the info in just oneline)

Rules for Commit Messages

Atomic commits (one commit for one change)

→ Keep commits centric to one feature, one component or one fix.
Focused on one thing

→ Present or Past commit message (which should be used)

⇔ • depends { Present tense, Imperative } (write a msg as if you are giving order to code base)
• give order to code base

git Configuration file

7) git Config

(refer: docs.github.com)

→ get started

(scroll) → git basics (or type set your username)

→ set your username
(then follow the commands)

→ git config --global user.name "Mona Lisa"
enter

→ git config --global core.editor "code --wait"

this commands changes the default editor to VS code

you can set your git username for single repository also.

8). gitignore

• don't want to track some files (because it contains some sensitive info) keep it away from git ecosystem
• node modules, API key, secret
• get template online, patterns can be tricky

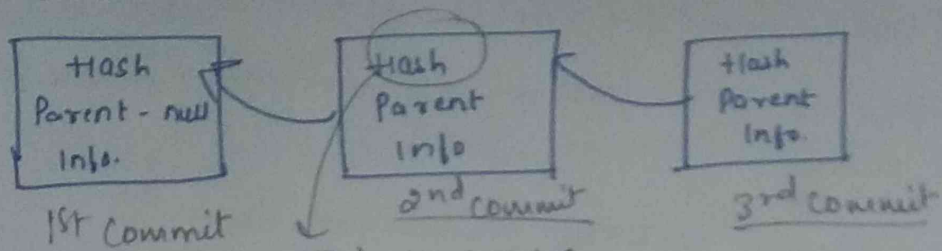
→ To know which files to be ignored visit:- gitignore generator

* go to c:/drive/users/91859 → and here use command

9) Cat .gitconfig → this will give the info about email & author of git & the editor being used

* Every commit is depended on its previous commit

→ Commit behind the scene (every commit is the result of previous one)



1st Commit

2nd commit

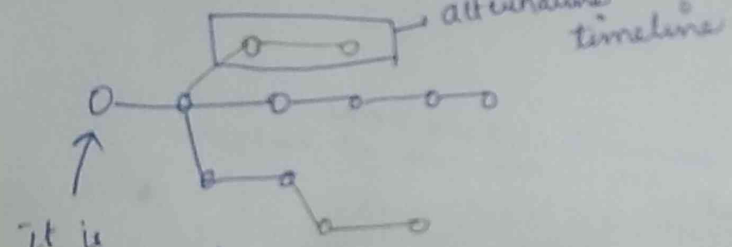
3rd commit

This Hash value is depended on the 1st Hash & Parent and also at some level it depend on Info.

Branches

↳ it is like a alternate timeline

(Jo bhi persons same code p work kr rhe they prefer to choose different branches so, it won't affect their actual work flow)



It is called main / master branch

10) git branch (It shows all the possible branches that are available)
it gives the branches that are involved.

→ To name a branch

4) git branch nav-bar (To create new branch)

a new branch of named nav-bar is created

→ To move to different branch

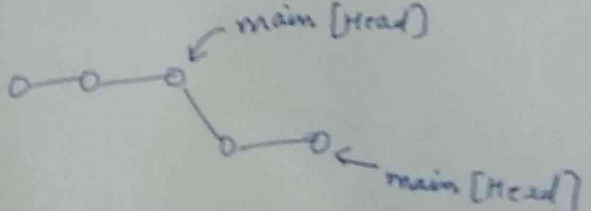
(2) git switch / Checkout nav-bar

switched to branch nav-bar

{ To check if it really worked give cmd git branch and * is of nav-bar }

=> Head → Master

(Head points to where a branch is currently at)



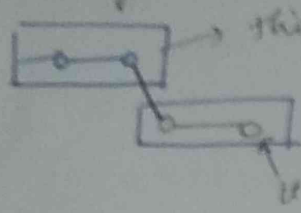
→ We can also point to head to another branch as well ← it can also be checked by (git log)

git checkout branch_name ← this will make head to point at this branch.

13) git switch -c branch name / git checkout -b branch name.
(this will create a branch and move you there)
does the functality of
- git branch
- git checkout

* always
  - commit before switching to another branch
  - go to .git folder & checkout HEAD file

Merging the branches



fast-forward merge

our task is to combine/merge these branches.

14) git merge branch name

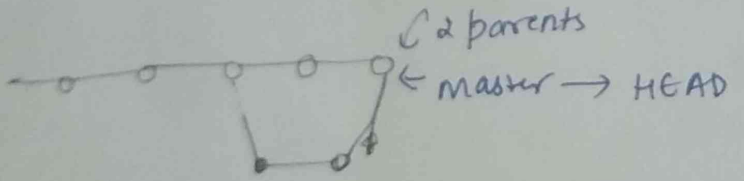
* We have to make sure before merging the HEAD is pointing to the main branch

15) git branch (-d) nav-bar
↑ it deletes the branch.

Step by step procedure

1) git checkout -b footer
2) git branch → pointing to footer
3) git add footer.html
4) git commit -m "add footer section"
5) git checkout main
   ↑ Pointing back to main
6) git merge footer

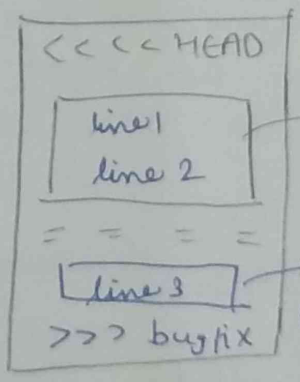
these are done under footer branch and main branch has no idea about it



Not-fast-forward merge

* git tries best to resolve conflict

Conflict situation



yeh us branch ka code h jispe we are currently at

conflict that came from another branch

* In this case, manually edit krna pdega (jo bhi special / un-necessary symbols h like ==, HEAD, <<, >> unko remove krna) → just keep the code which is needed

Git diff (informative command)

This command compares the before and after versions of same file

(before staging was done) (after staging was done)

How to read diff

- a → file 1 (before) & b → file 1 (after)
- --- file 1 (before) [indicates changes in the file
- +++ file 2 (after)]
- changes in lines & little preview of it

16) git diff --staged

It is run after git add . / git commit

It gives the changes made b4 & after staging.

Git stash (means store temporarily. bina current branch p kiye changes ko commit kre, next p nhi ja skte it will show conflicts) error

- create a repo, work & commit on main
- switch to another branch & work
- conflicting changes do not allow to switch branch, without commits

17) git stash (means keeping the changes on hold, not exactly applying it)

↳ git switch another branch

↳ git branch ~~~

(it will indicate that branch have been changed)

18) git stash pop (the stashing can be applied to other branches as well)

↳ it will bring back those changes that are not staged before moving to other branch.

19) git stash list

↳ gives the list of stashing done

20) git stash apply stash(id) file name

(it will apply changes & keep them in stash)

21) git checkout <Hash> (how things look like at this point)
↳ means remove the head
& move it to new branch
(whose hash value is <Hash>

22) git switch main
(point head to main branch)
(move back to the main branch)

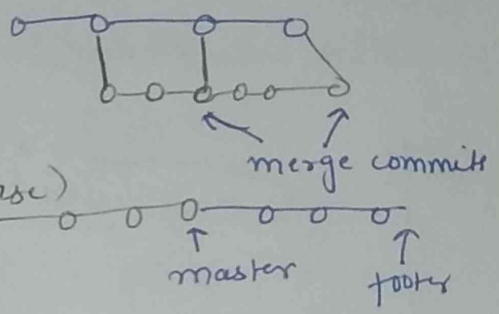
23) git checkout HEAD~2
(look at 2 commits before
how it looked before)

24) git restore filename
↳ get back to last
commit version

25) git reflog
↳ it moves head to
back to wherever it was

Git Rebase

→ It is alternative to merging
→ clean-up tool (clean up the commits)



* If you are on Main / Master branch never run this command.
→ it is meant to be run from the side branches.

27) git rebase main
(it means make my branch same as main branch)
or dissolve my branch into main branch.

26) git commit -am "__"
shortcut to add & commit at same time

*

* Never Rebase commits that you have shared
* Pushed to Github → Never Rebase

GitHub

→ visit docs.github.com
→ search SSH < generating a new SSH key & ~~adding it to ssh-agent~~
Adding a new SSH key to your github acc.

→ then set the SSH key (refer Chatgpt for more)
→ save it to github.

GitHub Discussion

Git is a software & Github is a service To host git online

• Github → collaboration + Back-up + Open source

Gitlab & Bit Bucket are the two other platforms that provides such services.

Cmds → run this and you can bring the repo to your local device

→ git clone <URL> link of the repository
→ git config --global user.name " "
→ git config --global user.email " "

⇒ Set up SSH keys to connect with github, github uses SSH to allow you to push code. Password based code push is not allowed

→ Checks instruction for your OS on github website as that's best & updated resources

28) git remote -V
↳ it shows whether there's any repository on your local system that is connected to your github.

29) git remote add name url
It add the website with the given url & name remotely to github

30) git remote rename oldname newname
To rename the repository

31) git remove name
To remove the repo.

32) git push <remote name> <branch name>

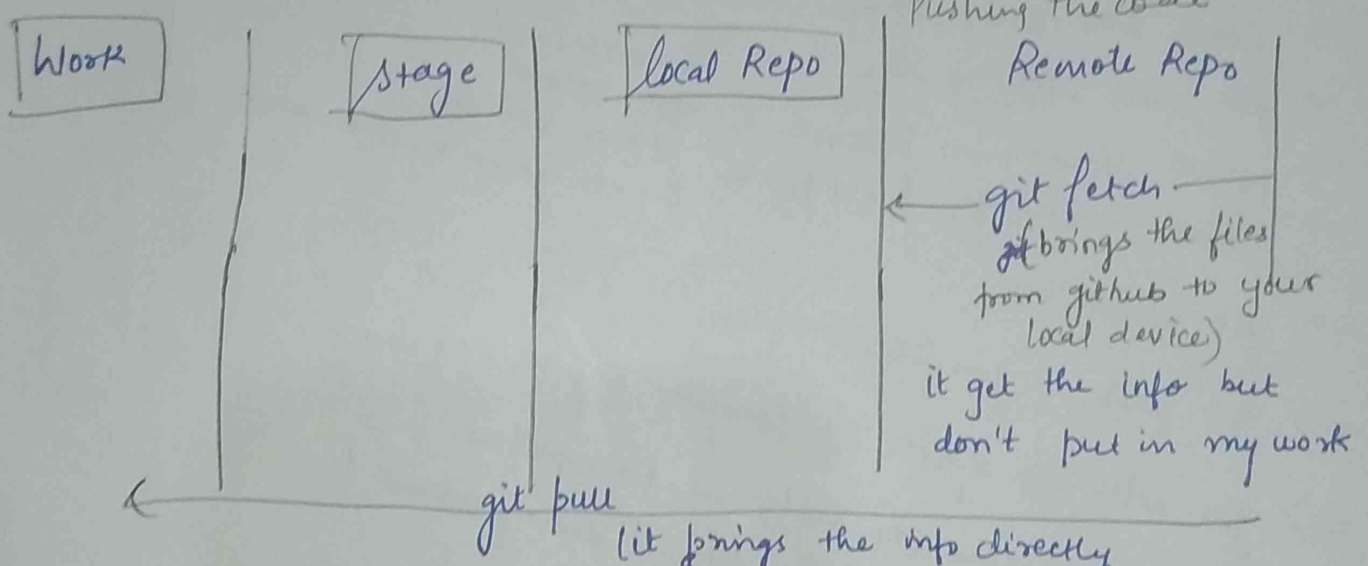
eg git push origin main

To push file into github

33) git push (-u) origin main

This sets a upstream taaki baar-baar origin | main na likhna pde aur sirf git push se hi file push hojaey.

→ When you clone a repo, you get just main branch connected, rest of remote branches are not configured



git pull = git fetch + git merge

34) git pull origin main

↳ it will merge the changes to main

Discuss github features on websites like

- adding collaborators
- Readme files (Readme.md) Mark down
- Mark down format
- Adding Gists
- Codespaces
- Dev: Containers

Steps to Open-Source Contribution

1) Talk to the owner / community (ki aap konse feature / bug ko fix kre h)
2) Open an issue (may be some functionality)
3) get the issue assigned to you
4) work and **add value** to the open source
5) ~~Have patience~~ Make PR (pull-request) and iterate over it
6) Have patience — be ready to get review / critic over it
7) Making PR is (not) a job guarantee

To make pull-Request

→ ~~fork the code~~ repo.
→ copy the SSH / HTTP
→ git clone ~~copy the URL~~
→ create your own branch. [In terminal: · open folder · check branch · create your own branch]
→ make changes (add file)
→ git remote -v
→ git push origin your branch.
⇒ you will get option of Compare & pull request
· make sure the title is thoughtful.
· also add a relative detailed discrip. about it.
→ select create pull request