



Git

//
→ Hitesh Choudhary

hitesh dot com → X or twitter

OR

find me on youtube → Hitesh Choudhary

→ A talk on git

→ git & github are different

software service

→ Version Control System

→ track files for changes

→ Learning Path

→ Get the basics

→ Use it daily

→ face the Problem → solve the problem

→ I have covered more than basics in this

⇒ Repo

→ Git on system VS tracking (Repo)
git --version



→ Git history { Demo }

⇒

git status

git init (one time per project)

.git → a hidden folder to keep history of all files
& sub-folders

Don't re-shape without
tagging me !

\Rightarrow Commit



check point (like game)



\Rightarrow Stage

\rightarrow git init

Create file or files

git add file1 file2 || git add .

git status

⇒ Commit

`git commit -m "a good descriptive message"`

`git status`

→ Repeat 2-3 times

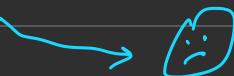
`git log` (log of commits with commit hash)

⇒ Atomic Commits

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

→ Present or Past commit message

- Depends { Present tense, Imperative }
- give order to code base
- Don't care



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

→ changes default editor to VSCode

git log -- one line (too much in docs of log 

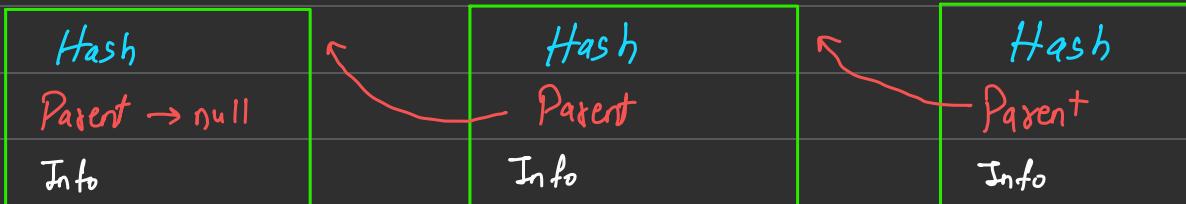
⇒ *.gitignore*

→ Don't want to track some files

→ node modules, API key, secret

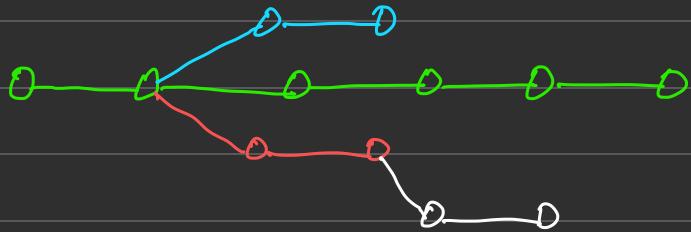
→ get template online, patterns can be tricky

⇒ Commit behind the scene



⇒ Branches

→ Like an alternative timeline (not from Dr. Strange)

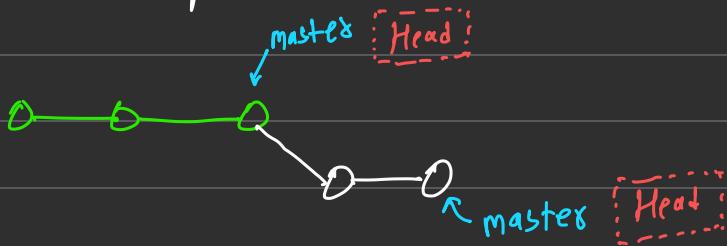


you are always on some branch

is main really special?

⇒ Head → master

Head points to where a branch is currently at



git branch

git branch bugfix

git switch bugfix

git log

git switch master

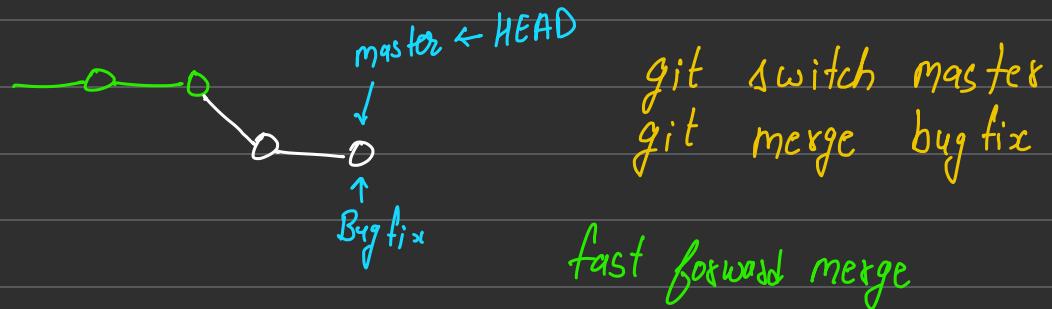
git switch -c dark-mode (Create a branch & more there)

git checkout -b pink-mode

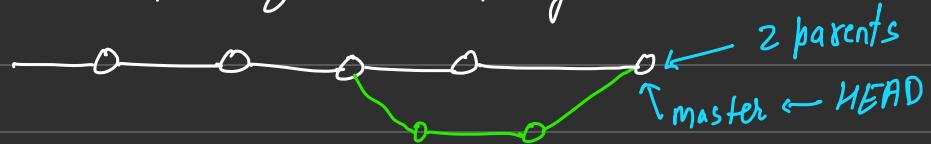
→ Commit before switching to another branch

→ go to .git folder & checkout HEAD file

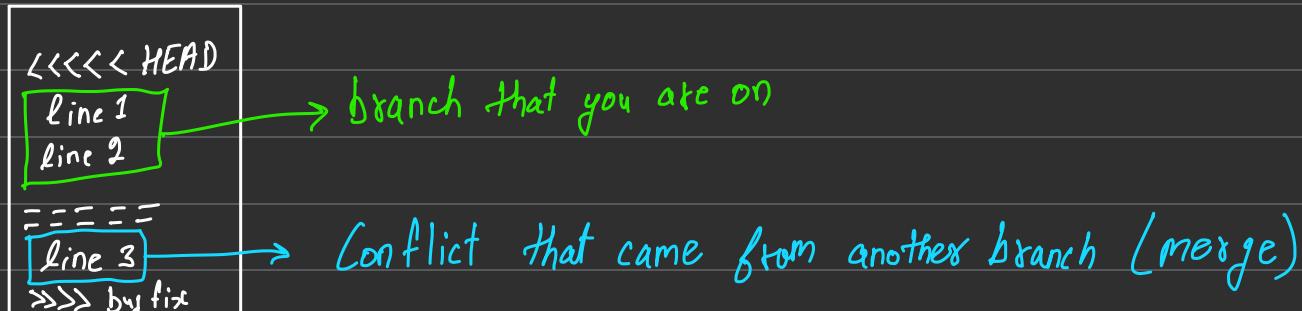
⇒ Merging the branches



→ Not fast forward merge



→ git tries best to resolve conflicts



Keep whatever you want, remove markers & save



⇒ Git Diff (informative command)

git diff (Compare working with staging)

→ How to read diff

- a → file1 & b → file2 (same file over time)
- --- file1 { indicates changes in file }
- + + + file2
- changed in lines & little preview of it

⇒ Git Stash

→ Create a repo, work & commit on main

→ Switch to another branch & work

→ Conflicting changes do not allow to switch
branch, without commits

`git stash` (you can switch branch)
`git stash pop` (bring back those changes)
optional
`git stash apply` (apply changes & keep them in stash)

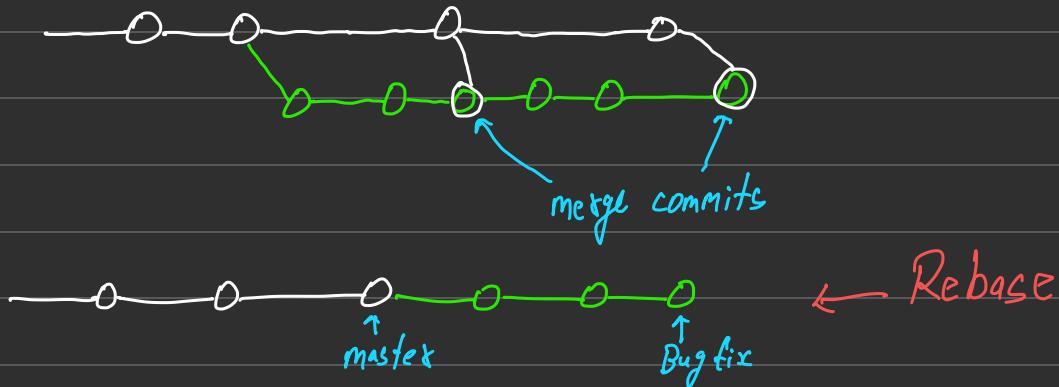
⇒ More Commands

`git checkout <Hash>` (Detach Head) : new branch
`git switch main` (re-attach Head)
`git checkout HEAD~2` (look at 2 commit prior)
`git restore filename` (get back to last commit version)

⇒ Rebase → ~~-~~

→ alternative to merging

→ Clean up tool (Clean up commits)



git init

work & commit on main branch

git switch -c footer

work & add a footer in feature branch

git commit -am "add footer text"

git switch main

work & add hero section on main branch

git switch footer

git merge main

work & add more info in footer

if there is more work on main, do more merge

→ get on footer branch & rebase

git rebase main

work & add more on main branch

move to footer branch

git rebase main

Never Rebase commits that you have shared
Pushed to Github → NEVER rebase



some companies have strict guidelines regarding commit messages like ticket id to be added. You can use rebase to re-write your git history, before sending a pull request
git rebase -i HEAD~5 (order is reverse)
pick, reword (your hash also gets a change)
there are many more commands

⇒ Tags : like a sticky note

git tag

git tag -l "*beta*"

⇒ Light weight tags & Annotated tags ↗^{-a}

git tag v2.0.1

git tag -a v3.0.0

git show v3.0.0

`git tag hitesh 24ab7b289` (tag previous tag)

tags needs to be unique

`git tag -d hitesh` (delete tags)

`git push --tags` (push to remote repo)

⇒ Reflog

git keeps track of where your HEAD is moving,

ex: branch to branch

These logs are local, not shared & are expired

`git reflog show HEAD`

→ Github Discussion

Git is a software & GitHub is a service to host git online

- GitHub → collaboration + Backup + Open Source

Gitlab OR Bitbucket

git clone <URL>

git config --global user.name "hitesh choudhary"

git config --global user.email "hitesh@hitesh.ai"

- Setup SSH keys to connect with GitHub, GitHub uses SSH to allow you to push code. Password based code push is not allowed
- Check instruction for your OS on GitHub website, as that's best & updated resource

git remote -v

git remote add name url

git remote add origin https://github.com/hiteshchoudhary/chai.git

git remote rename oldname newname

git remote remove name

git push <remote> <branch>

git push origin main

git push <remote> localBranch : remoteBranch

git push -u origin main

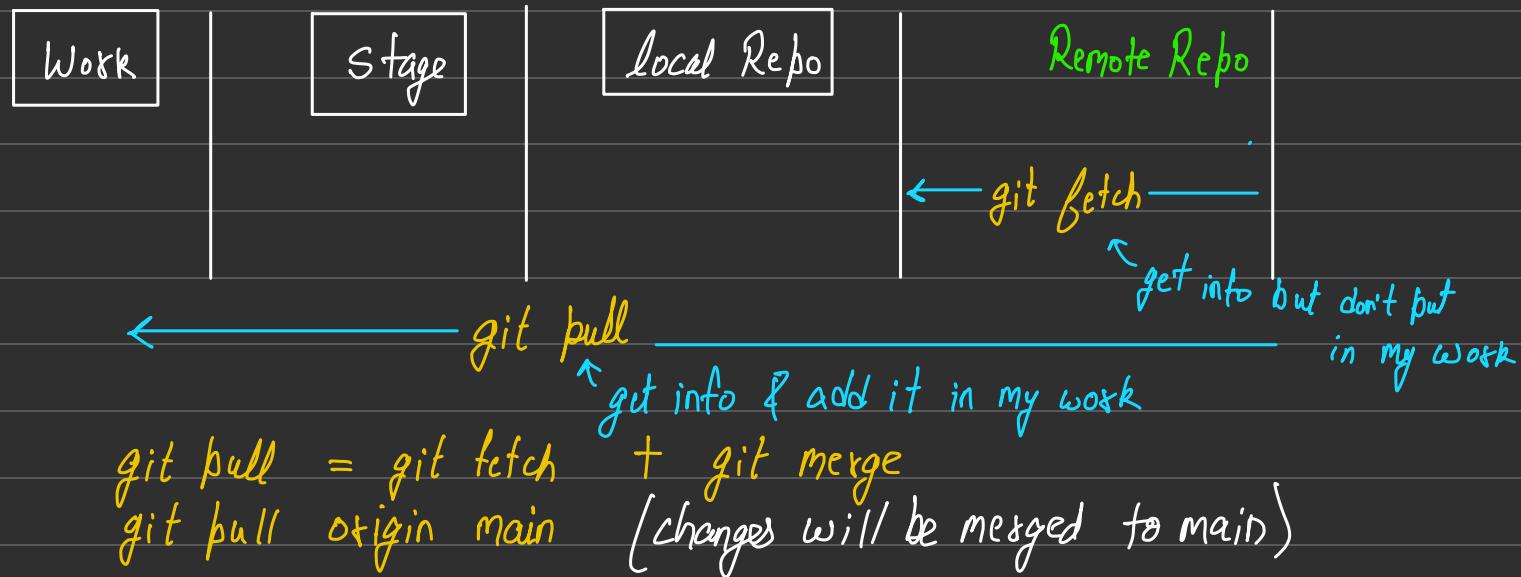
-u set up an upstream that allow you to run future command

git push

& it push the code directly to github

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

`git switch branch-name`
connects remote branch to local
`git branch -x`



Discuss github features on website like

- Adding collaborators
- Readme file
- mark down format
- Adding Gists
- Code Spaces
- Dev Container

All this is discussed in class or video

If you find this being shared, please tag me

"Hitesh Choudhary" on youtube
@hitesh dot com on twitter or X