Git:

- Git is a **Version Control System.**
- **Version Control System** is a set of tools that helps to track changes in the code.

Git helps us to:

- Track history.
- Collaborate.

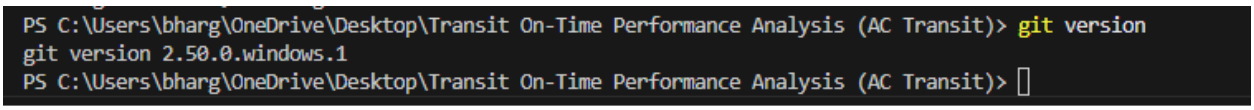
GitHub:

- GitHub is a website that allows developers to store and manage their code using Git.

Setting up Git:

- Install VS Code.
- For Windows users, we need **Git Bash.**
- For Mac, we need **Terminal.**

After Installation, run the code **"git --version".**



Configuring Git:

- git config -- global user.name “My Name”
  (Enter your GitHub username to link and global means changes affects all the repositories).

- git config -- global user.email “xyz@email.com”
  (give the email associated with GitHub).

- git config –list
  (stores all the credentials)

Clone & Status:

- **Clone –** Cloning a repository on our local machine.

  git clone <-link->

- **Status –** displays the status of the code.
  Modified or added or yet to be committed etc.

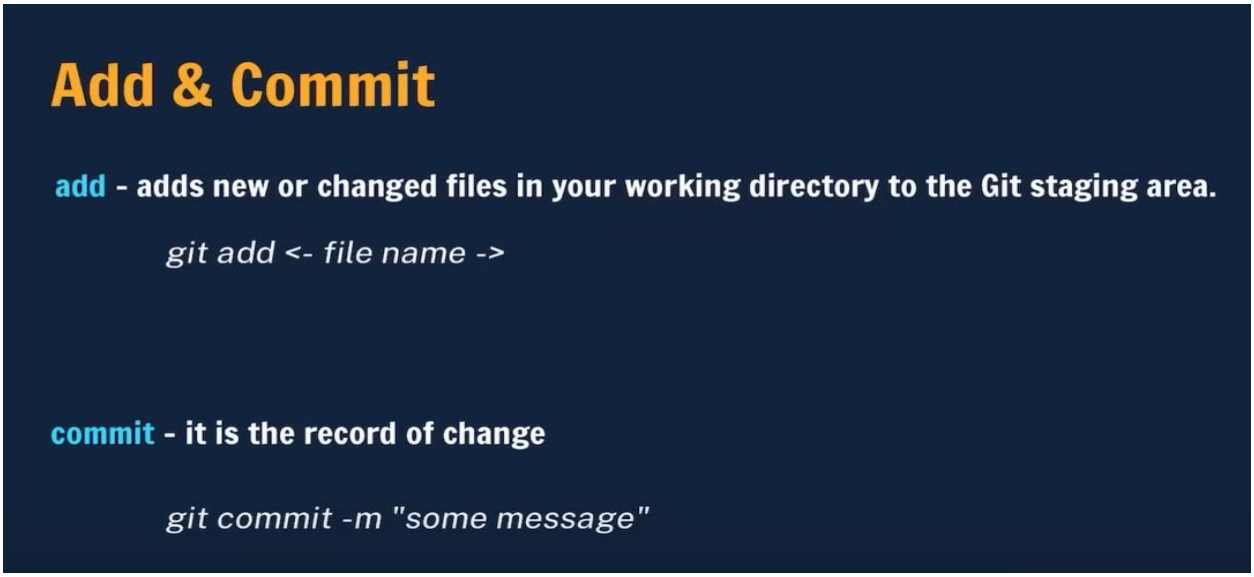
- ls : used to display all the files under the folder.

- **ls -a**: used to display hidden files on mac, **ls -h**: used to display hidden files on windows.

- A GitHub folder or repo will always have a .git file to track.

Types of File Status on GitHub:

- **Add & Commit:**



Using “git add .”, we can add all the files at one go into the repository.

- **Push Command**:

All the local changes which are added and committed to the local machine repository now need to be pushed onto the Online GitHub. Because local repo is now one commit ahead of the remote machine.

git push origin main:

- **push –** upload local repo content to remote repo.
- **origin –** that repo which we cloned on our local git.
- **main –** we are pushing our code onto the main branch.

Init Command:

Init – used to create a new git repository.

✅ When Do You Use `git init`?

Use `git init` when:

1. **You're starting a new project** from scratch and want to track it with Git.
2. **You have an existing local project folder** (not cloned from GitHub), and now you want to put it under version control.
3. **You're not cloning** from a remote repo — you're initializing your own.

> 🔴 You do **not** need to run `git init` if you're cloning a repo using `git clone` — cloning already initializes the repo.

🧰 How to Use `git init` (with Example)

📁 Let's say you have a folder:

plaintext Copy Edit

```
C:\Users\Bharg\OneDrive\Desktop\MyNewProject
```

🔧 Steps:

bash Copy Edit

```
cd "C:\Users\Bharg\OneDrive\Desktop\MyNewProject"
git init
```

This will create a hidden folder `.git/` in `MyNewProject`, enabling Git tracking.

✅ After `git init`, What Next?

bash Copy Edit

```bash
# Step 1: Add files
git add .

# Step 2: Commit your changes
git commit -m "Initial commit"

# Step 3 (optional): Link to remote repo
git remote add origin https://github.com/username/repo.git

# Step 4 (optional): Push to GitHub
git push -u origin main
```

📘 Summary

Task	Command
Initialize repo	`git init`
Add files to staging	`git add .` or `git add filename`
Commit changes	`git commit -m "Your message"`
Add remote repo	`git remote add origin <repo-url>`
Push to GitHub	`git push -u origin main`

Init Command

init - used to create a new git repo

git init

git remote add origin <- link ->

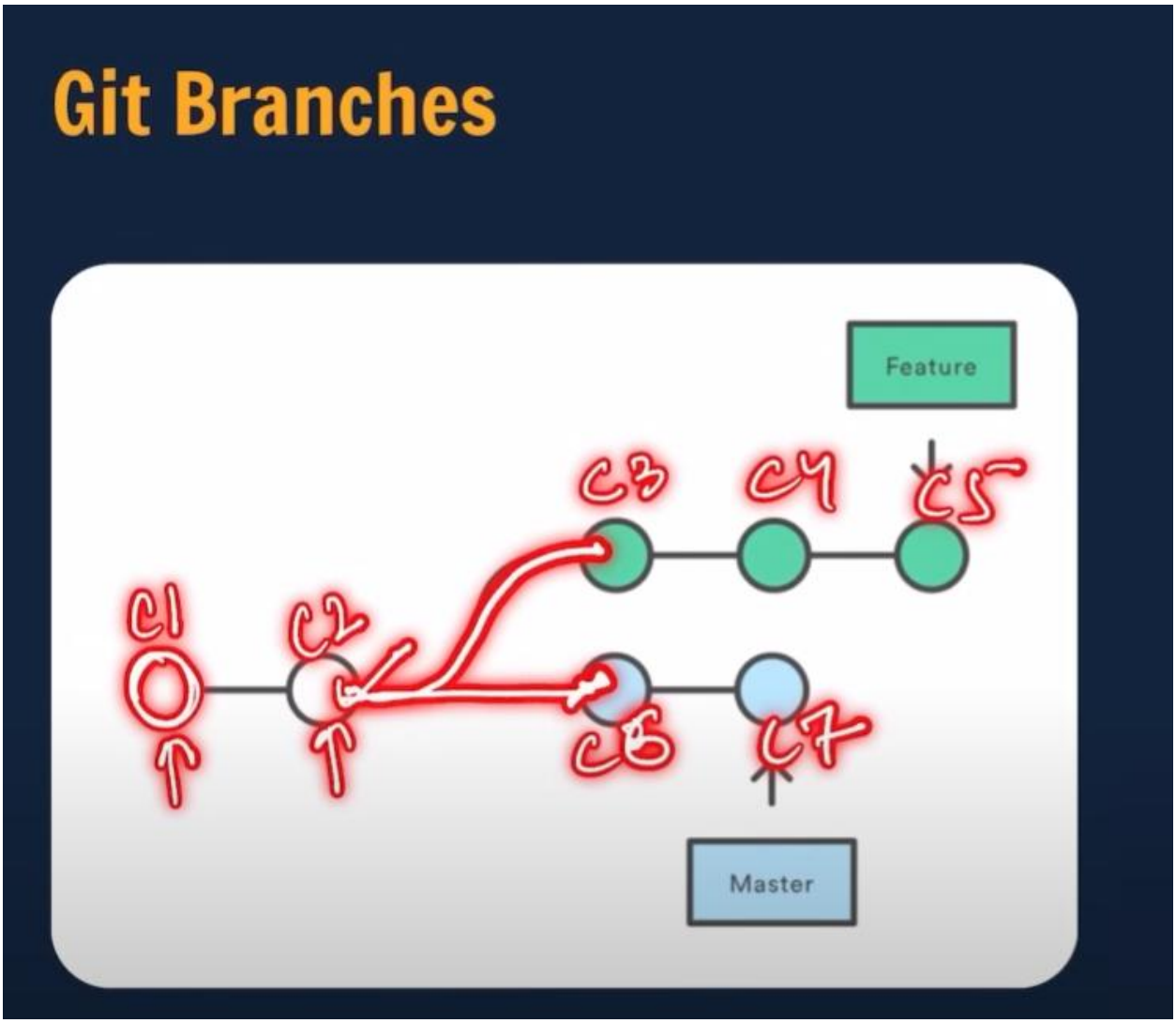
git remote -v (to verify remote)

git branch (to check branch)

git branch -M main (to rename branch)

git push origin main

Git Branches:

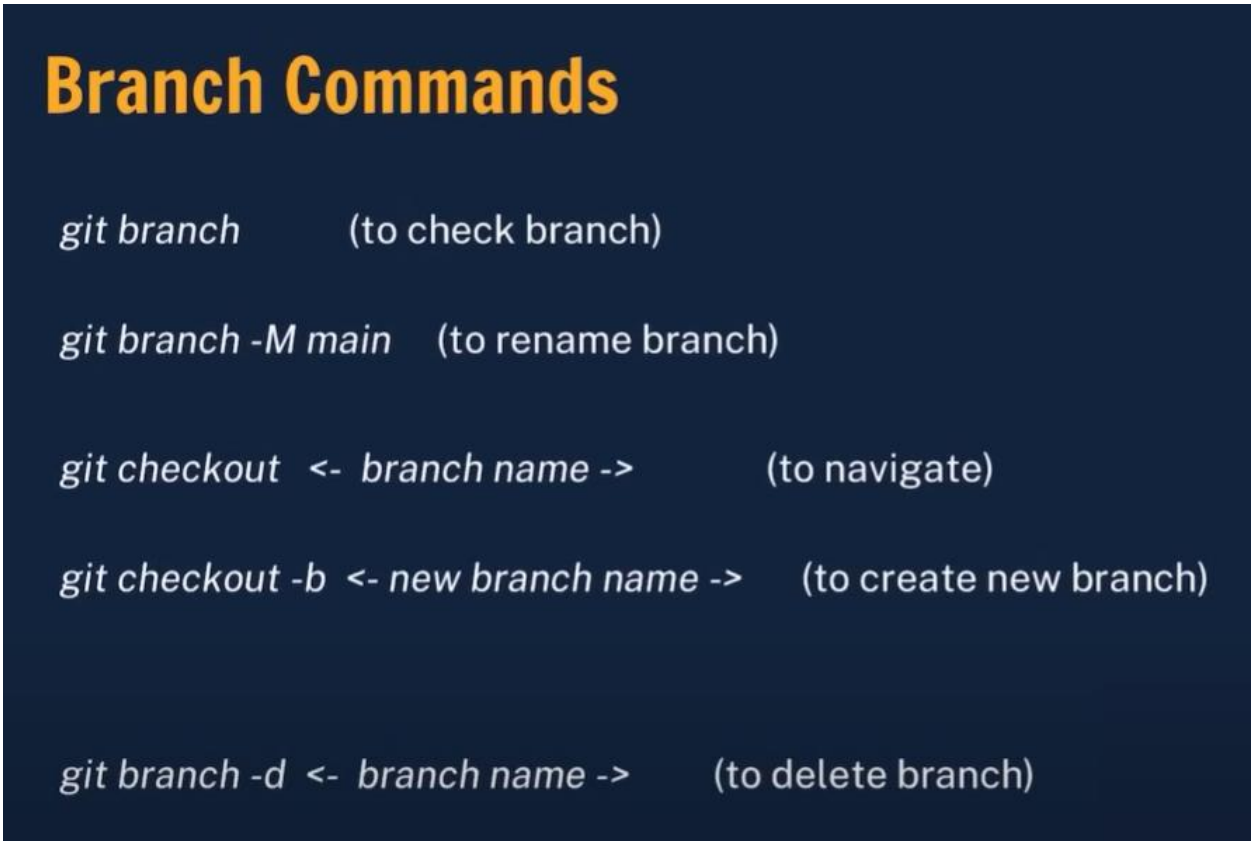


Branch Commands:

- git branch (to check the branch)
- git branch -M “new branch name” (to rename branch)
- git checkout <- branch name-> (to navigate to another branch)
- git checkout -b <-new branch name-> (to create new branch and to navigate to it)

- git branch -d <branch name> (to delete branch)

  “We cannot delete a branch on which we are right now”.



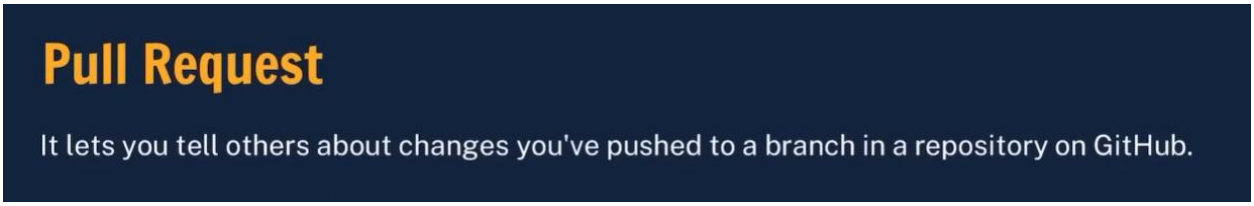
- **<u>Merging Code</u>:**

  Way1:

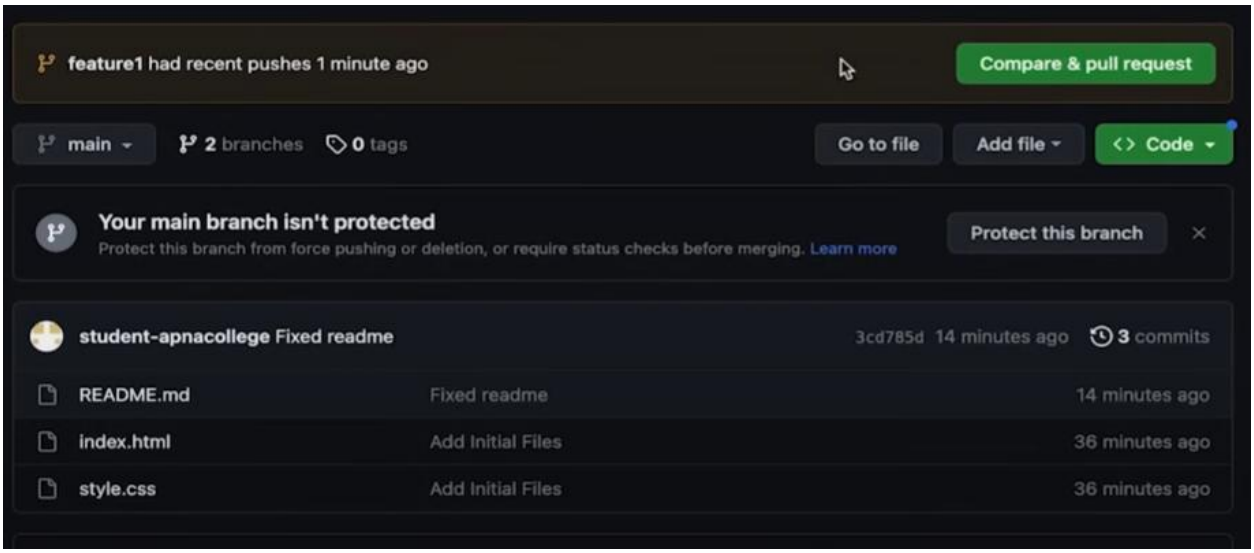
  git diff <- branch name -> (Compare difference between two branches)

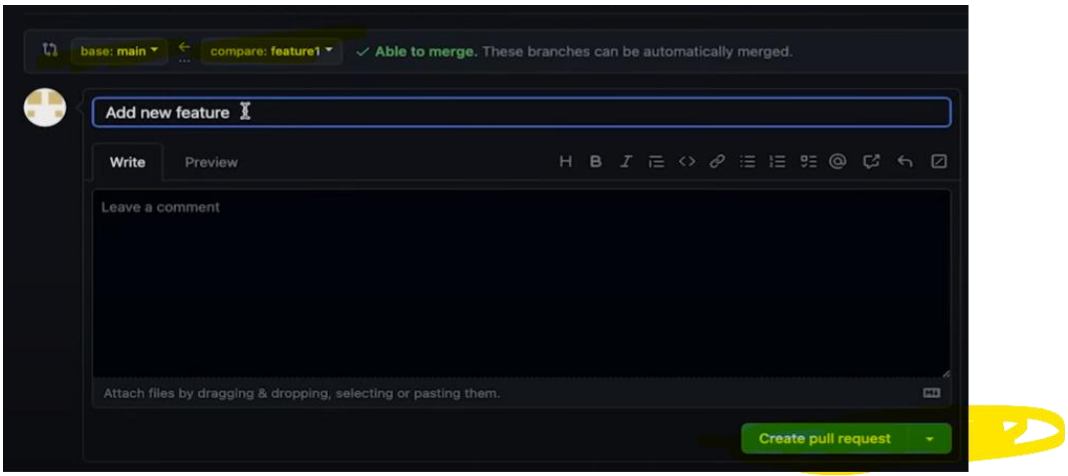
  git merge <-branch name ->
  (If you are on branch1 and want to merge with branch 2, then run “git merge branch2”).

Way2: Using Pull Request



Once we push our code from any other branch apart from main, GitHub by default shows us a pull request suggesting to merge.

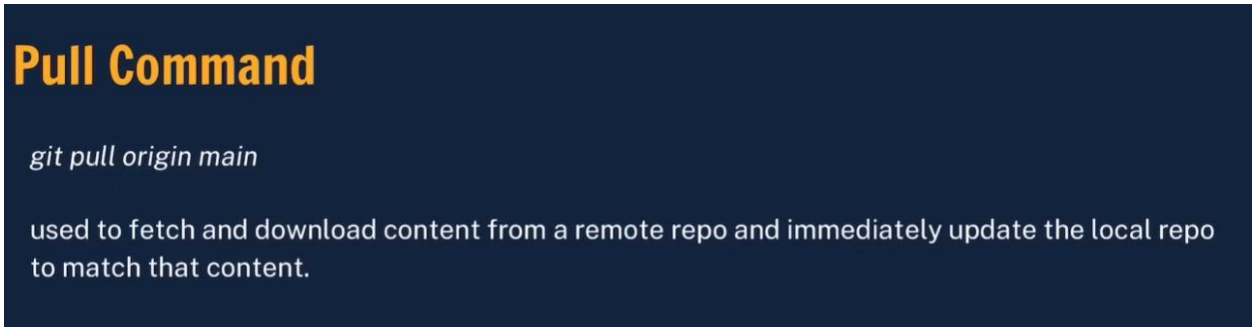




- Here, we are comparing **the feature1** branch with **main** branch to check for the compatibility to merge.

Pull Command:

- git pull origin main.
- Used to fetch and download content from a remote repo and immediately update local repo to match that content.



Resolving Merge Conflicts:

An event that takes place when git is unable to automatically resolve differences in code between the two commits.

Undoing Changes:

Case 1: staged changes (changes which were added, but not committed)

git reset <- file name -> (for few files, we can give file name manually)

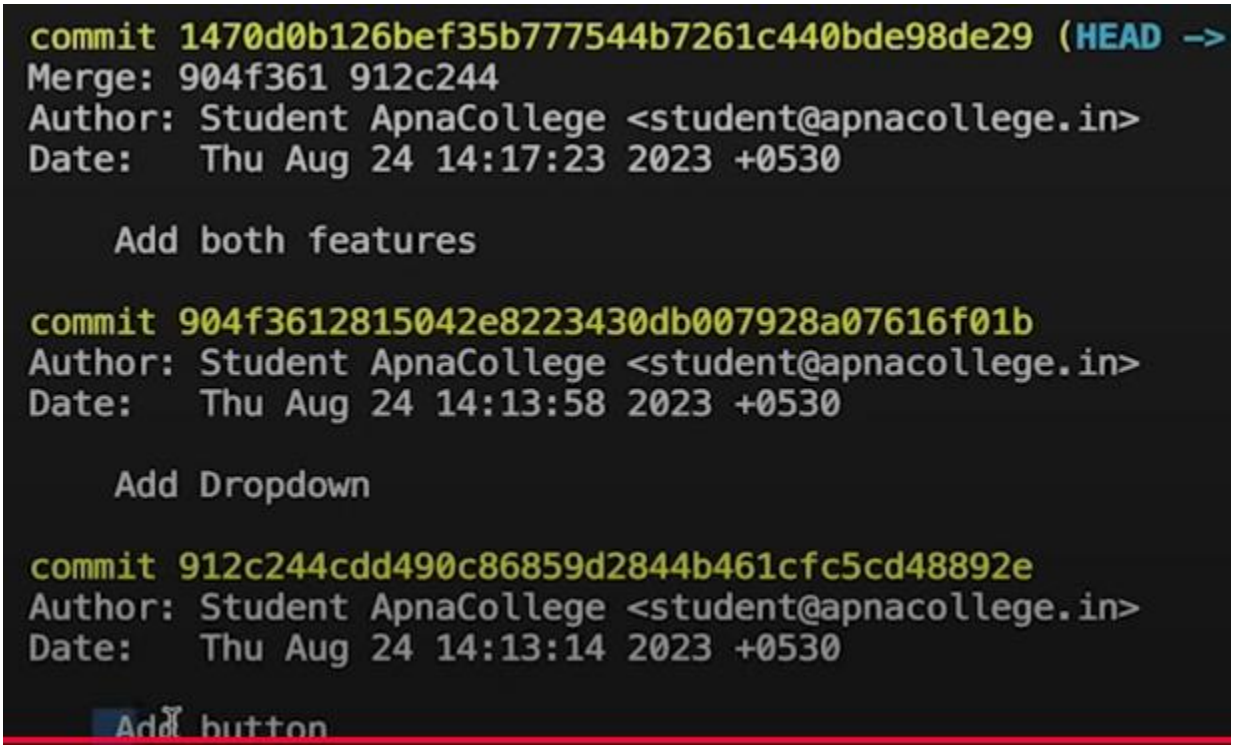
git reset (for all files added, we can undo changes by resetting it)

Case 2: committed changes (for one commit)

git reset HEAD~1 (goes back to one previous commit)

The last commit by default is called **Head.** (After running this command, changes become unstaged. That is, now we need to add the modified file again).

Git log: By running this command, we get the list of all the commits that have been done so far. Also, we get commit hash along with it.



To quit from the current git logs and get to the git terminal, we can press the **“q”** button.

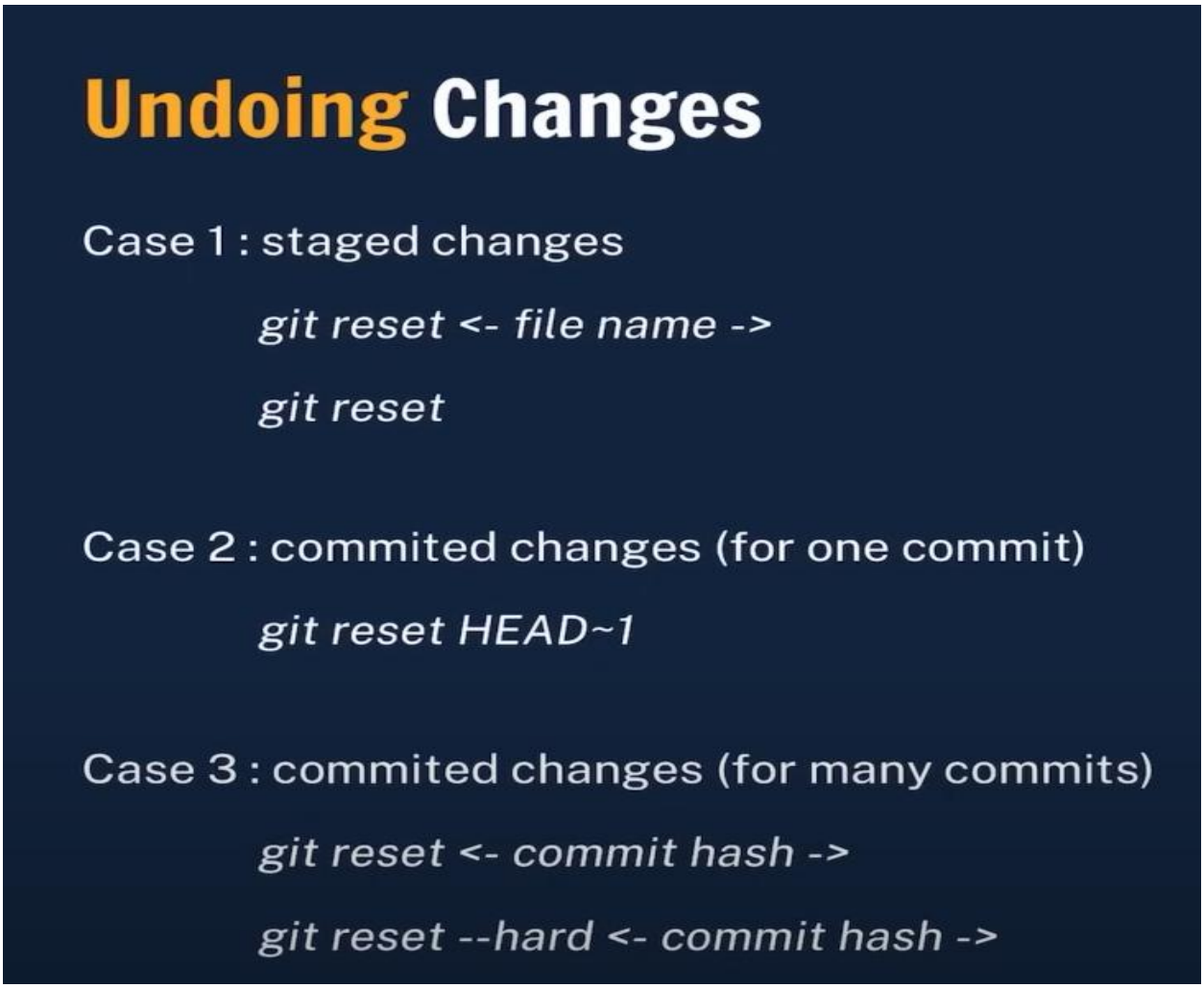
Case 3: committed changes (for many commits)

- git reset <-commit hash->

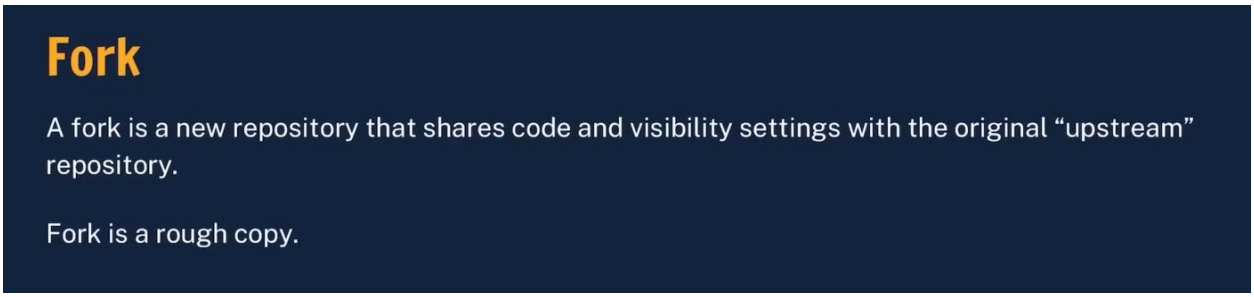
  Directly jump onto that hash, but the code changes are still there on the VS code (not matching with the committed code).

- git reset -- hard <-commit hash->

  Jump onto the hash code and the code gets changed on VS code as well.



Fork:



Create a rough copy of another’s project onto our GitHub repository.
