GIT HUB

Contents

1. GITHUB

GitHub is a writing a wrapper of git, means Microsoft acquired GitHub, a popular code-repository service used by many developers and large companies, for $7.5 billion in stock.
Before acquiring they owned something known as Azure Repos.
Likewise, many companies own a GitHub software (Bit bucket is owned by Atlassian) (Gitlab is an open-source version) (even Amazon has something known as Amazon Code Commit [codeccommit])

2. GIT (TECHNOLOGY OR SOFTWARE)

Means it is a version control software
You can see that whatever I have done in the repository is displayed here day by day.

Difference between git and github:

The key difference between Git and GitHub is that Git is a free, open source version control tool that developers install locally on their personal computers, while GitHub is a pay-for-use online service built to run Git in the cloud.

3. CODE SYNCHRONIZATION

Means for example my colleague is a backend developer and I'm a frontend developer, he's doing some code I am writing some different code,
Only if both of our codes merged together will make the application work properly.
Now we have to collaborate, so what we will do is we upload these files to a common place. Then we will download the common code from that place and will use it.

4. REPOSITORY

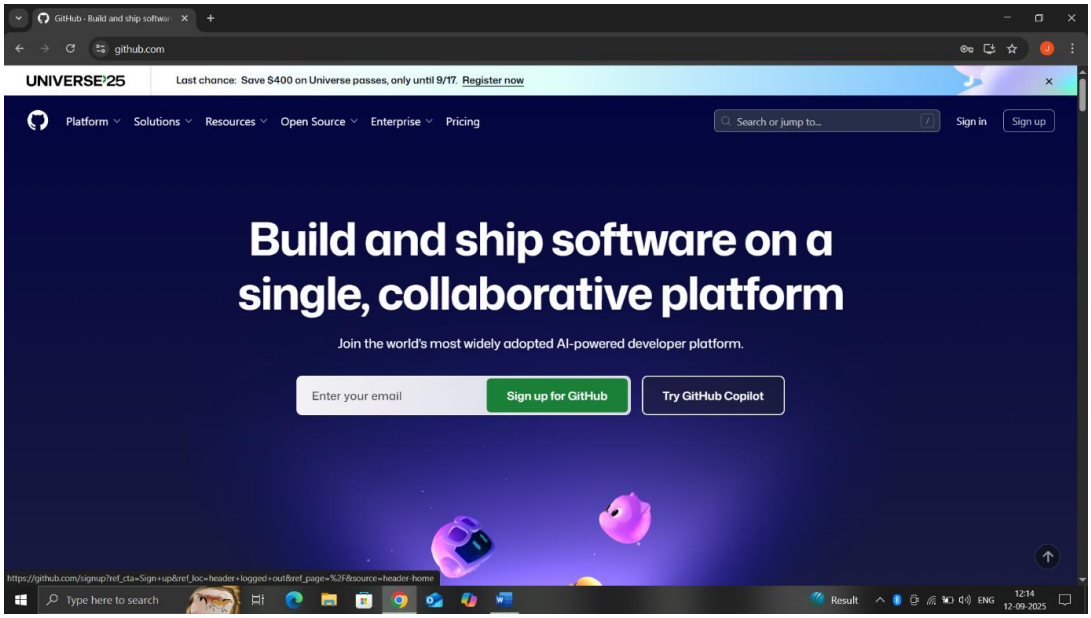
Repository means the place where data gets stored is called database and the files get stored in a file storage or folders.
Similarly, the place where code files get stored is known as the REPOSITORY. (Like folders)
There are 2 types of repos: local repository and cloud repository

1. Local repository: whatever I store inside my laptop is considered a local repository
   (Here I have created a folder name **learning html** whatever I do inside this folder is local changes because it is inside my laptop so this is a local repository)

2. I am uploading this to GitHub and it will upload to a server and the code will not be present inside our laptop. This is called as cloud repository.
   One from cloud repository everyone can collaborate if they have permission.

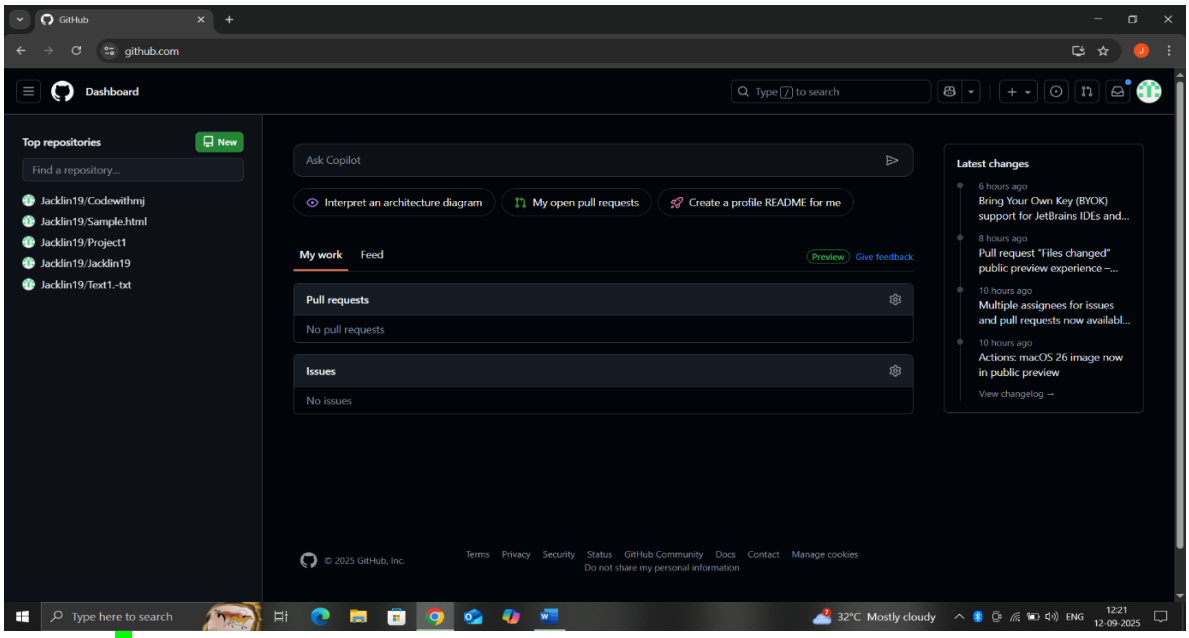
5. CREATE A GITHUB ACCOUNT

Github.com→right corner you will get signup button→click

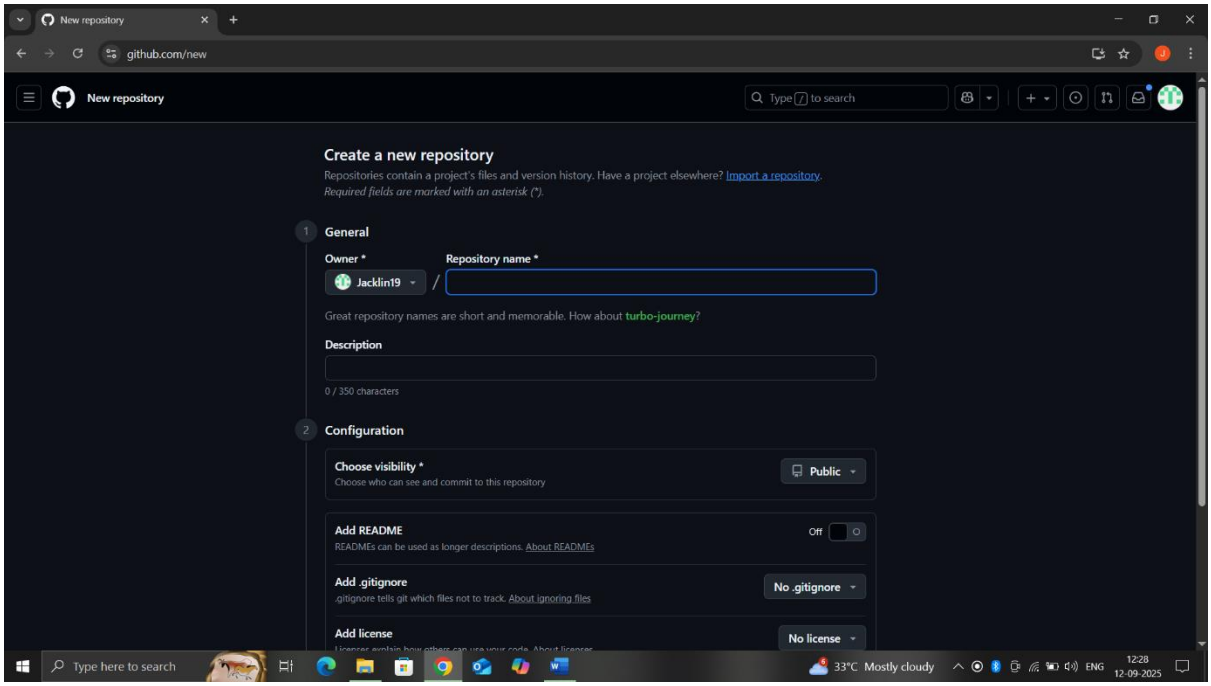


→ Show one input box →fill in the email→fill all the necessary information like phone number, otp etc. →then if you fill the captcha then the github account is ready. → If you want you can upload your profile picture. →Make sure you remember the username and password that you give.
→Now sign in you can see the home page with dashboard.

6. CREATE A NEW REPOSITORY

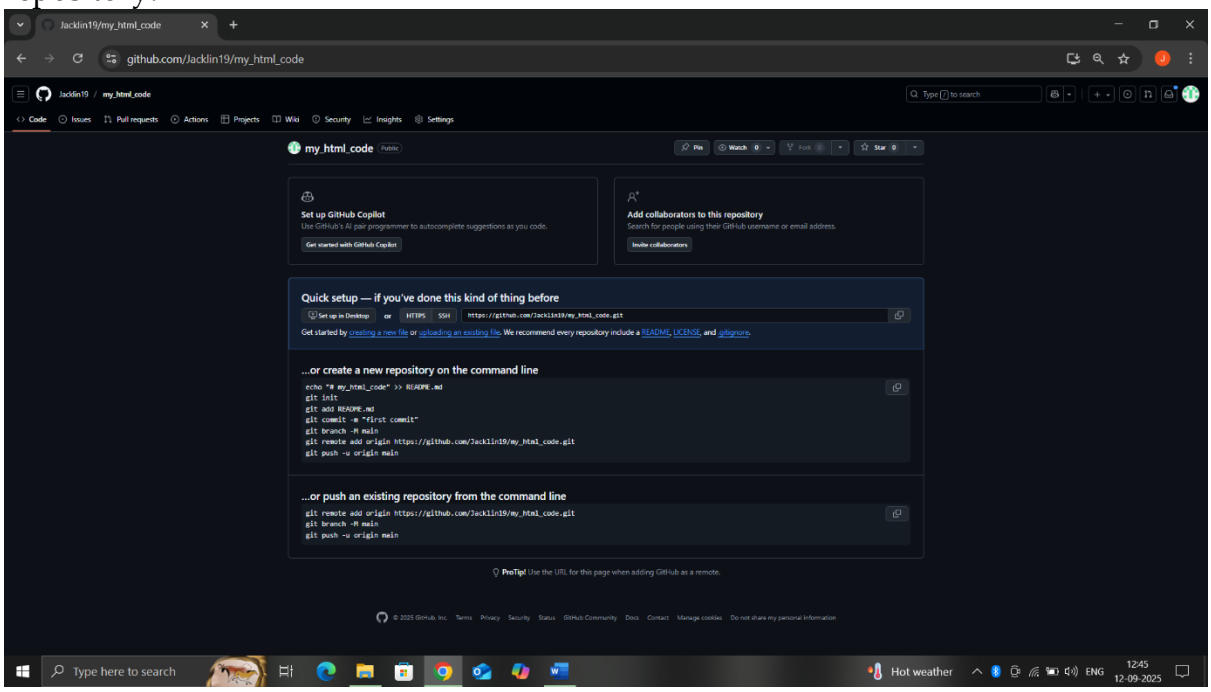


→Click + sign in the page of right side→it shows dropdown list→click new repository or click new which located in the left side.



→This is our repository creation page→give repository name→click public→do not click on **add readme** because it will give you instruction only if you don't click it → click create repository.

→Now you can see empty repository. It's all done through a browser called cloud repository.



7. CREATING NEW FILE

When you created a new repository, you can see **creating a new file or uploading an existing file** in that home page.

Click that option to create new file or if you want to upload already existing file can also upload.

→Name the file→commit changes (means save).

8. GIT INIT

Git init means you're initialising a local repository inside your laptop or creating a git environment.
If you want to create one inside our laptop then we have to use the **git clone command.**
There is a lot of commands but are you thinking where can I use this? To do this you must download git first.
The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository.

9. DOWNLOAD

Website→git download →click git scm downloads→click window→ click (**click here to download)** on first line →give ok to licence then click next then install button.
Go and check in command prompt→git→you get all details that git downloaded.

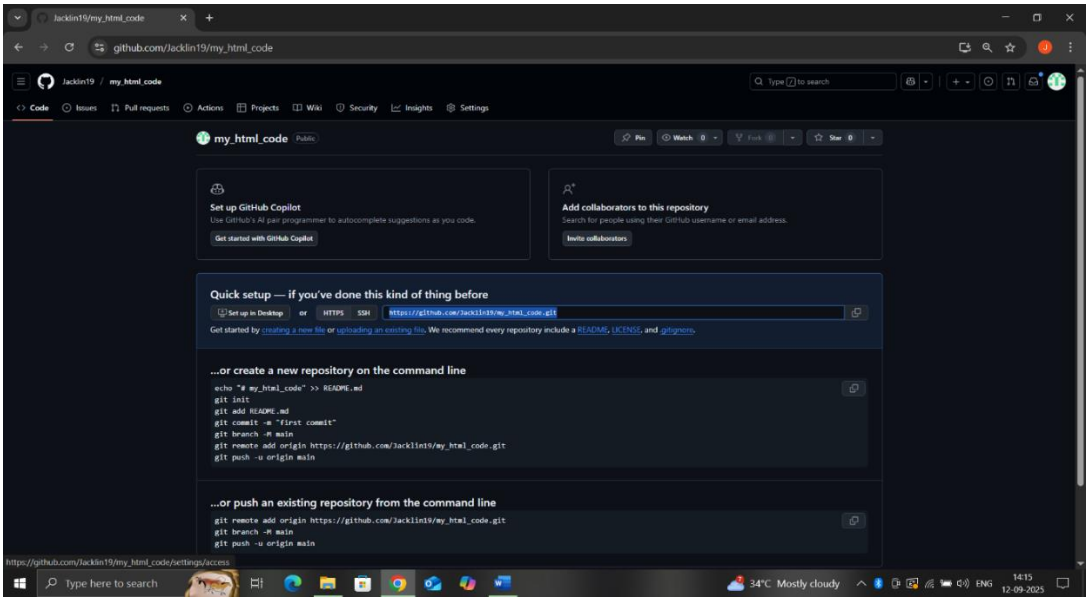
10. GIT CLONE

Already we have created repo is residing somewhere alone, the folder is residing somewhere else, means local repo and cloud repo. Now we have to somehow create a link between these two means now we have to clone the repository to our laptop. Clone means we are creating a copy of something.
A quick note: git init and git clone can be easily confused. At a high level, they can both be used to "initialize a new git repository." However, git clone is dependent on git init. Git clone is used to create a copy of an existing repository. Internally, git clone first calls git init to create a new repository.

11. GITHUB CONFIGURE

To configure github you have to use this following commend

- select folder you created in laptop
- copy path go to cmd
- type→git config –global user.name "Jacklin19" next
- type→git config –global user.email jack12345@gmail.com next
- type→git clone (copy path from git repository)

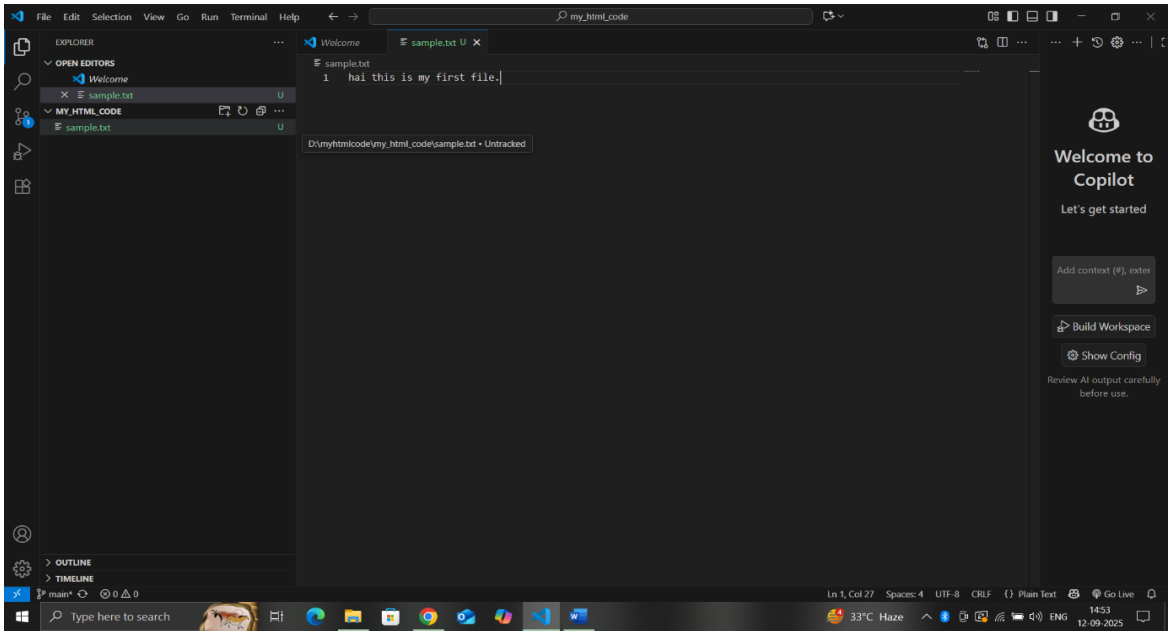


- After this we get git repository in local folder→close cmd→go and check local folder. Now everything has been configured perfectly.

→Now go and check in the folder which you stored in the laptop, inside you have clone repository which you created in git.
And also, you can see "**.git**" folder **it is the connection between your local repo and your cloud repo.** This folder will take care of creating communication, creating hash and doing commits etc.
And this is your link if you delete this you cannot push or pull.
By default, this ".git" folder will not be visible. To see this click **view** and then **show** and then **hidden items** so that the hidden files will also be visible.

12. CREATE FILE IN LOCAL REPOSITORY

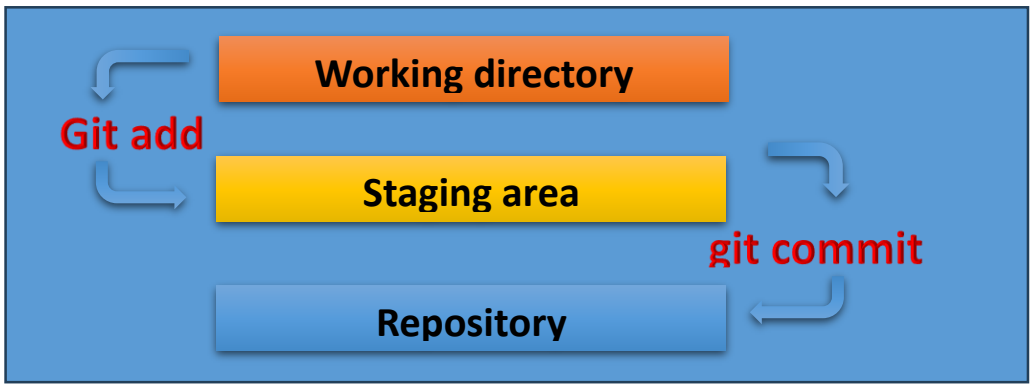
After clone go to folder you will get clone repository →select→go to path→copy type cmd it comes to cmd page→ then type code .--> you will go to vs code editor with this folder.
Create new text file (which you want to show in github) → you can see it is mentioned as untracked "U" in green colour.



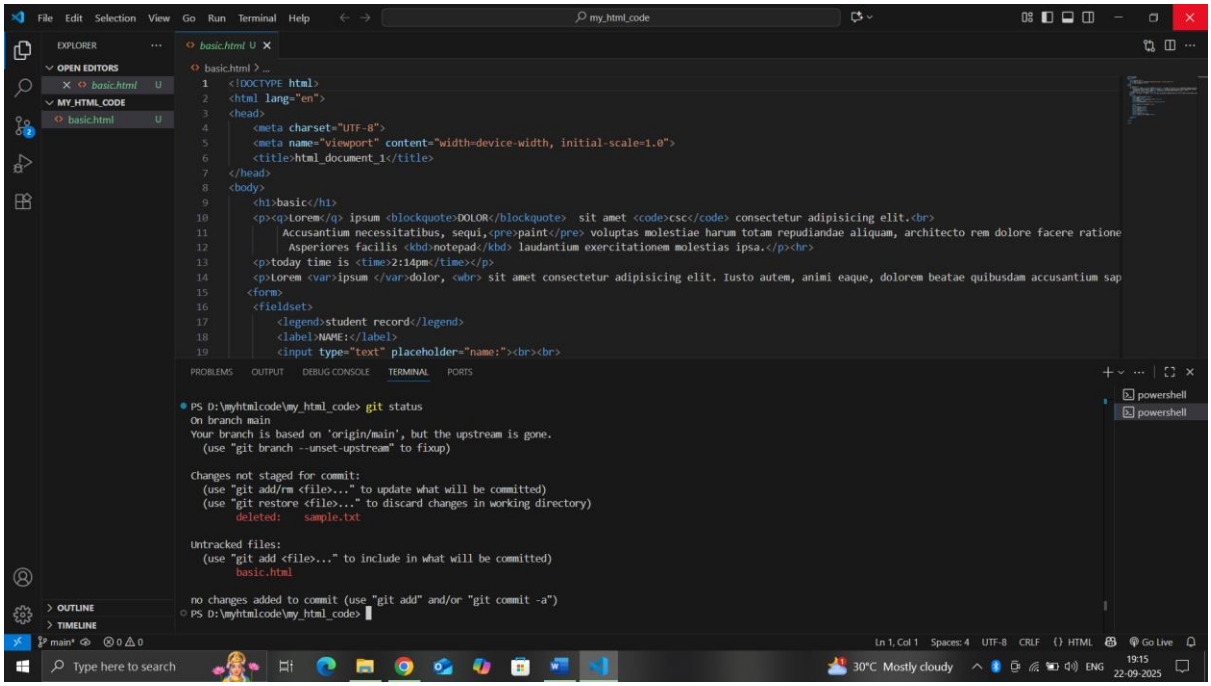
Untracked file means unstaged.

Staging and unstaging

- When you create an untracked file go to terminal type **git status** (it will check and tell you everything if you made any changes).
- If you use **git add** commend in git repository, it will make your files ready for **staging.**
- If you **committed** these will be staged.
- If you **push** it the staged files will be uploaded to the cloud.



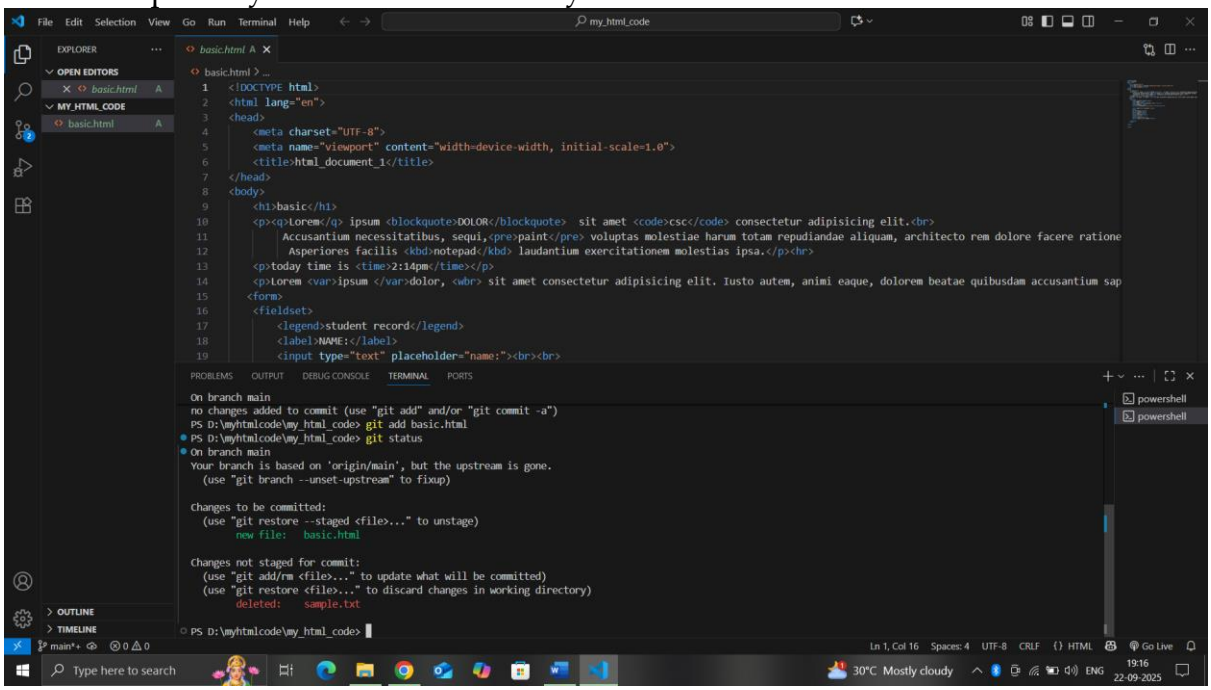
Git status:



13. GIT ADD AND GIT STATUS

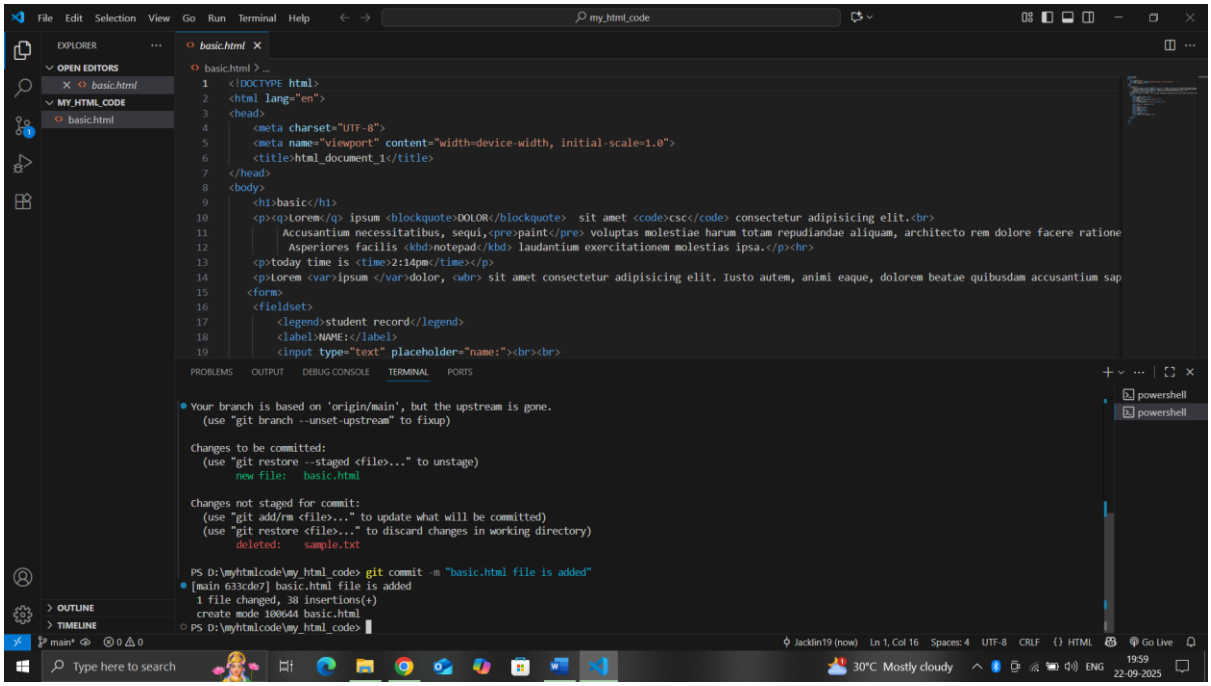
→**git add basic.html**
→**git status**

Git add means you're getting it ready for staging which means you have put it into the envelope but you have not sealed it yet.



14. COMMIT

→**git commit -m "added some content in basic.html"**



Now changed from untracked to tracked one.

Commit means you have sealed it and you have written the message on it called commit message.

Now you are placing your envelope on someplace, basically your files are on some stage.

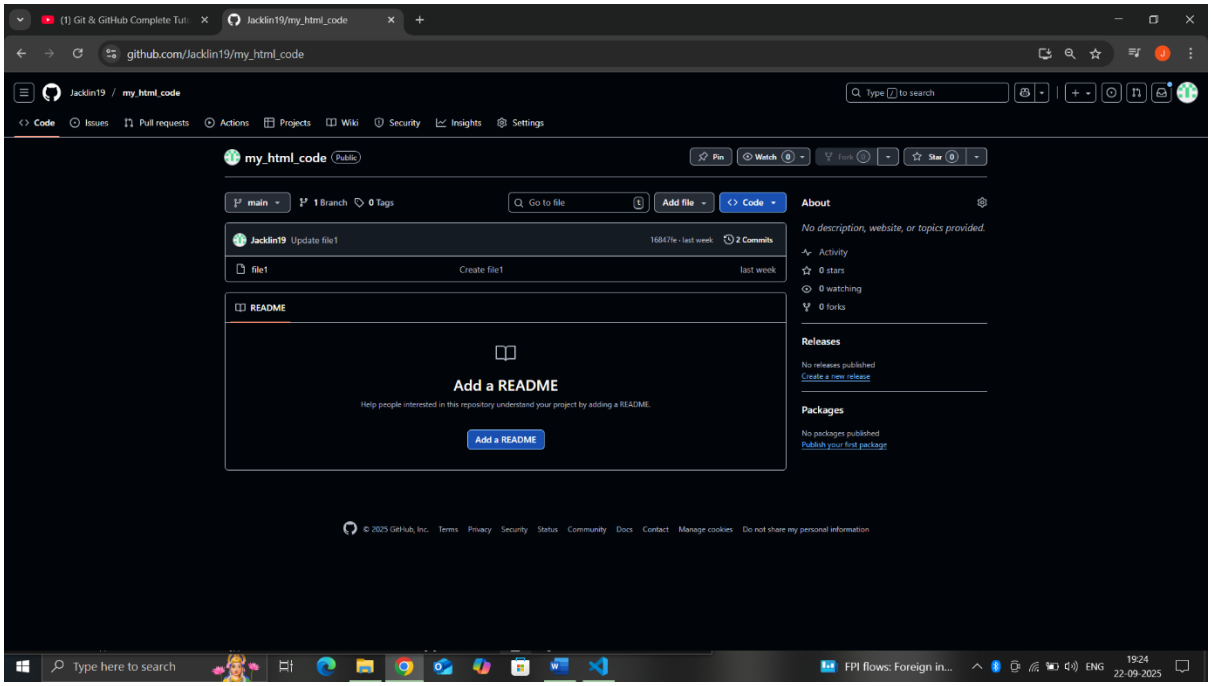
Whenever the Postman is ready, he can take your envelope it is very much ready to be posted.

Here -m refers to message and in " " have some message.

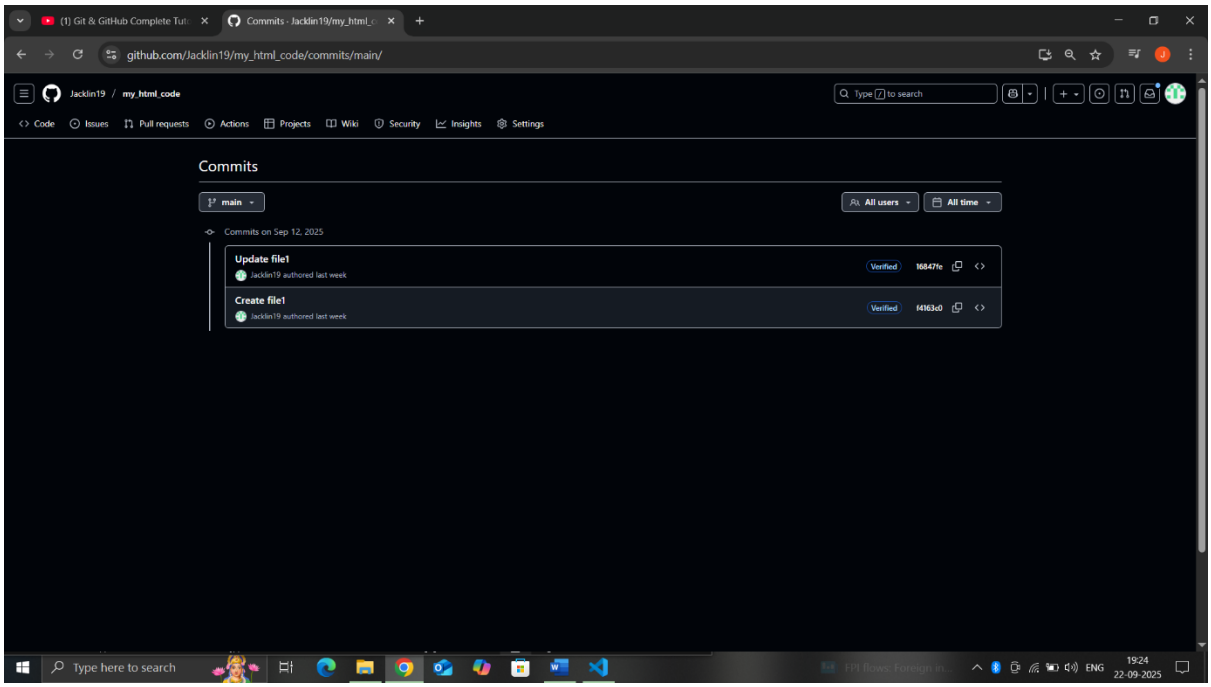
- Check conventional commits in website (conventionalcommits.org) for to write a commit message.
- Example:
    - If you're fixing something then use "fix: message"
    - if you're creating a feature then use "feat: message"
    - If you're refactoring something like changing a valid variable name you can use "refactor: message".
    - If you're writing some documentation then you can mention it as "doc: message".
    - If you're doing some unimportant work then you can mention it as "chore: message".
    - If you're making some changes that will affect other people then mention it with an exclamatory mark "! fix: breakage message".

This basically means conventional commit this is especially used when more than 100 developers are working. This will be helpful to track who made a fix and who created a feature and it will be useful to track yourself also.

Go to cloud repository to see commit message and commit hash:



→I have 2 commits click here.



→ Commit hash will be used in many places when we click on **commits** in our repo after **git add**, we can see all the commits have a **message** and also can you see the random numbers on the side. These numbers are called as **commit hash** it will be unique.

If any error while:

I have some mistakes in that now I am thinking of reverting some changes what I can do?

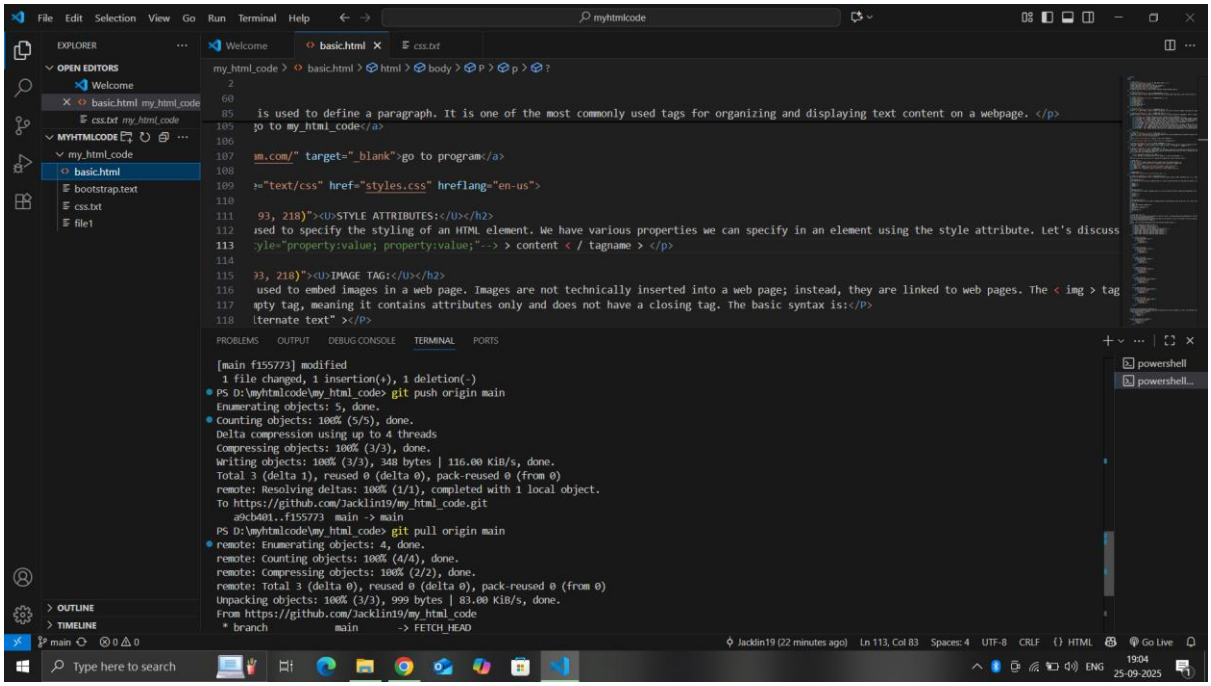
You can use the git reset command. Here if you have made some mistake and those mistakes are also staged and if you want to unstage those files you can use git reset to revert it to its original state stage.

15. GIT PUSH

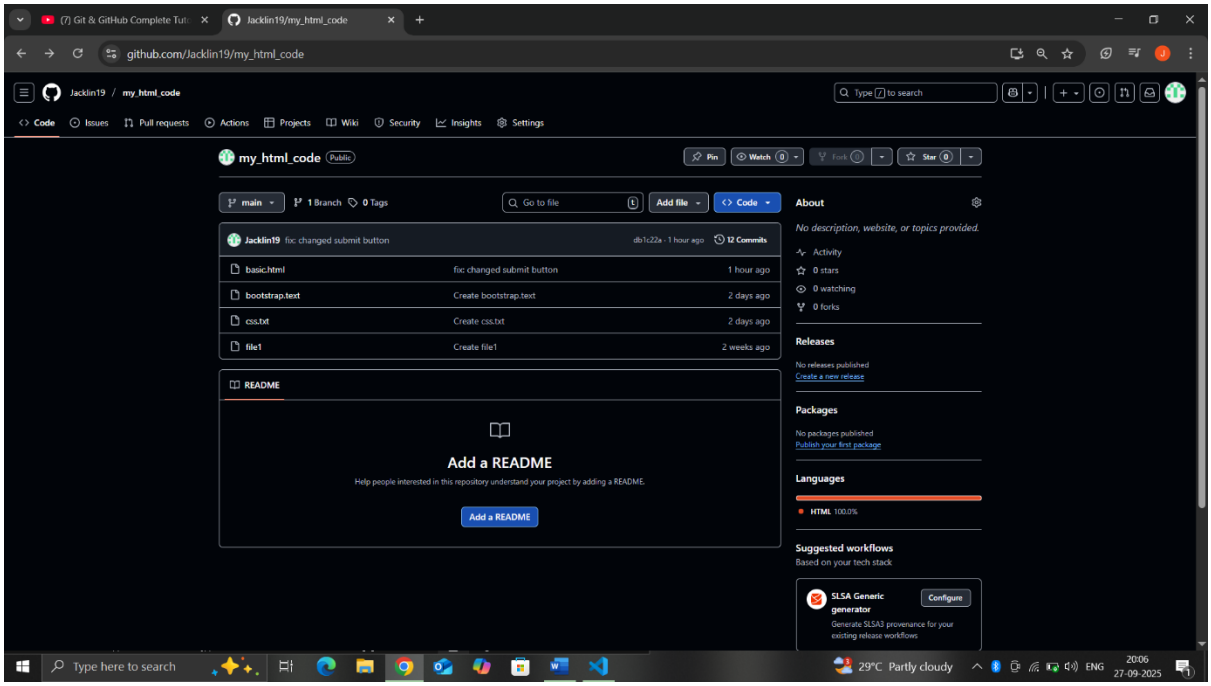
At this moment if you type **git push** after commit everything will be uploaded. You would have created many files but not all the files will be uploaded whatever you have added and whatever you have committed only those files will be uploaded to the cloud repo.

→git push origin remote or
→git push origin main / master.

- Because we have given a command git clone and some url on the starting. That url is basically the origin. Git knows already mentioned it. So, you don't have to mention the origin now externally.
- If you want you can mention, if you type out origin that specific url will be replaced here.
- Then what branch you are pushing for -is important. Now I am mentioning it as main branch or I will mention it as master branch.
- Commonly the most important branches are main or master where all the developers will be collaborating.



→Now it got posted →to check: go and refresh the browser you can see the file has been uploaded now.



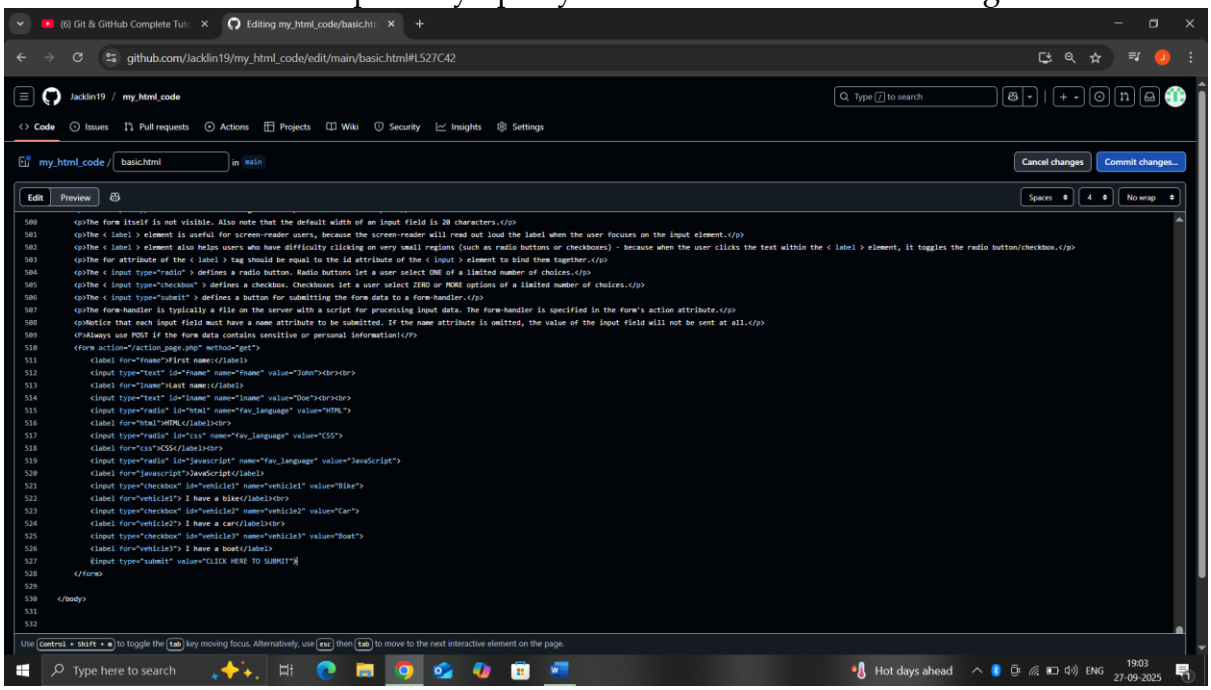
→You don't have to drag and drop and upload multiple files you can just use git add, git commit and git push.

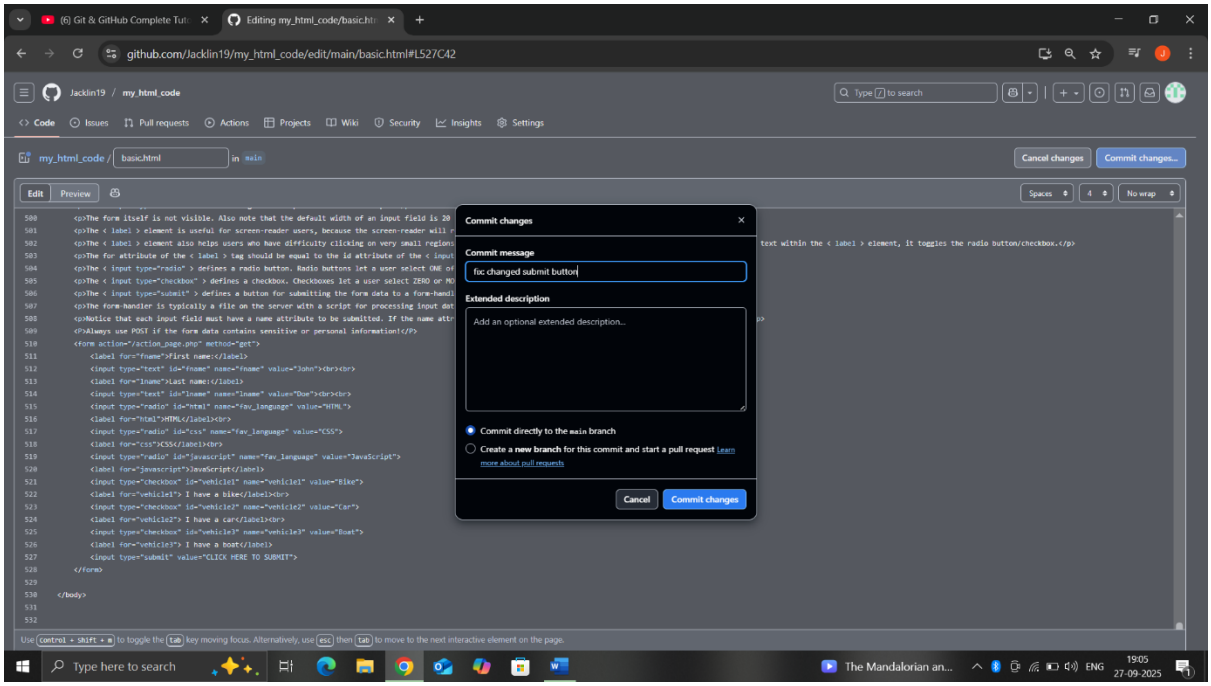
→Also you can check all the commit messages you have given in the time of commit (above the files of your right corner.)

16. GIT PULL

(Now I will make some changes in the cloud repository (just do some changes in content) and we will see how we can download it to our local repository)

→Go to GitHub cloud repository open your file and make some changes.



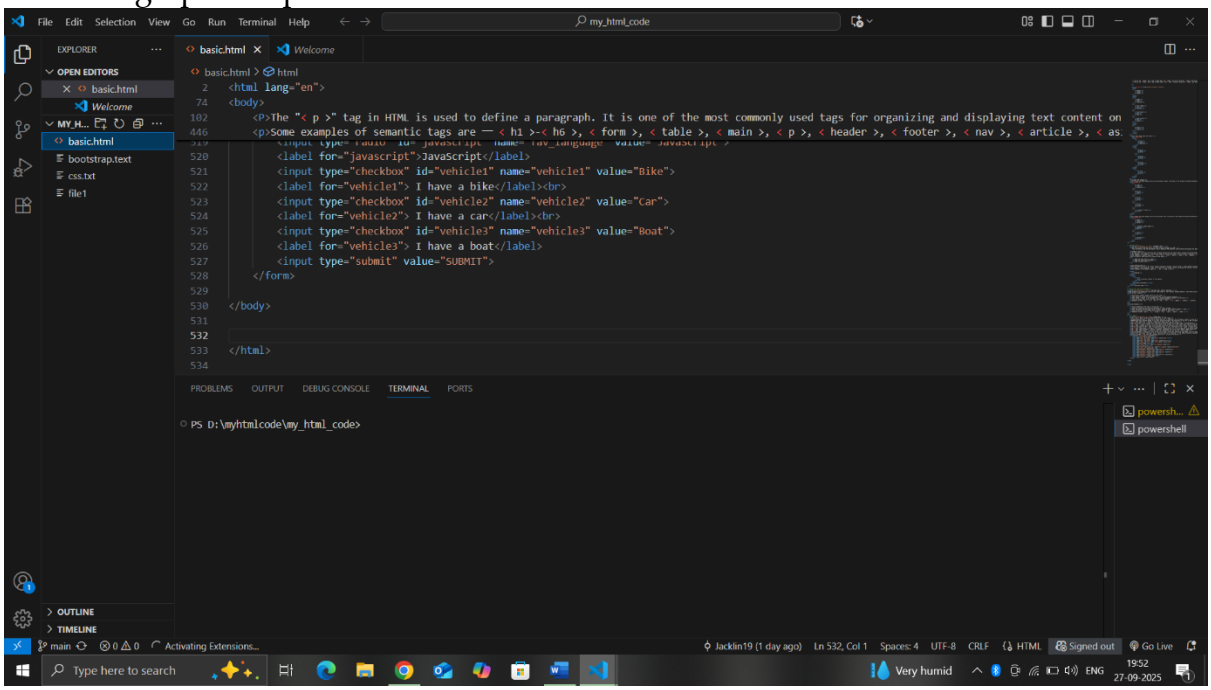


→Click commit changes→shows small dialog box→give commit message then click commit changes.

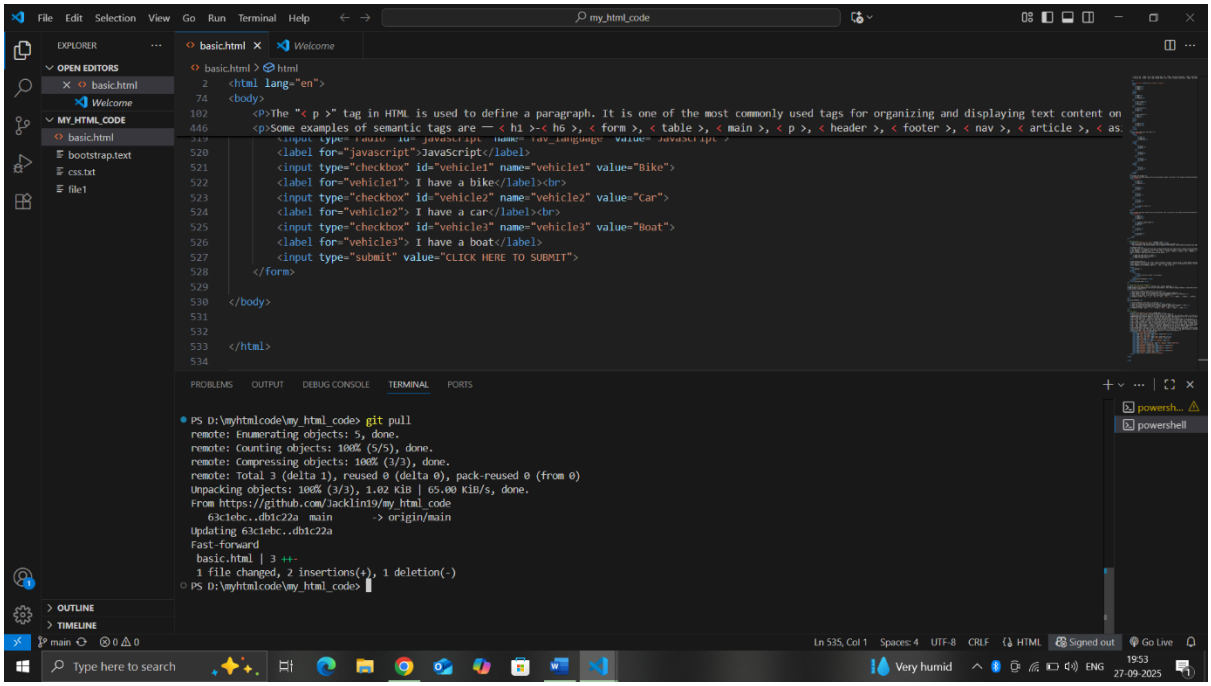
→Go to local repository there is nothing has changed. Now I want to download whatever that is present in the cloud repository.

→To upload from local to cloud repository we used git push to upload obviously to download we will use the git pull command in local repository.

→Before git pull request:



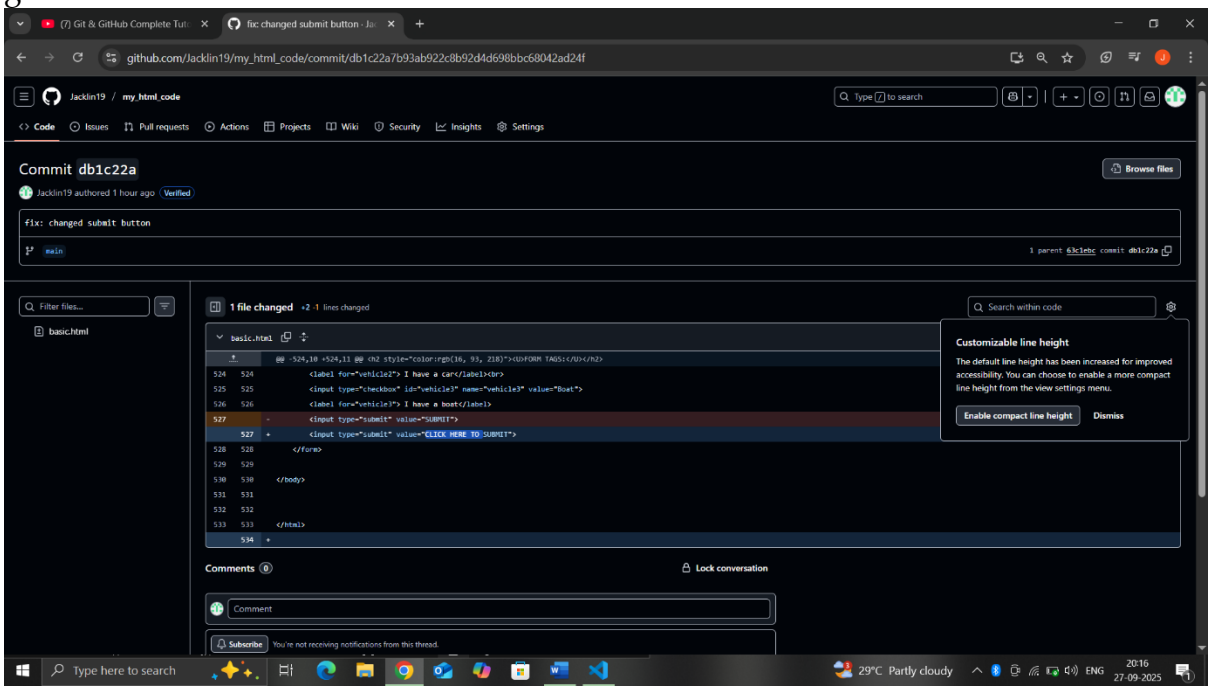
→After git pull request:



Now it will be downloaded.

17. GIT DIFF

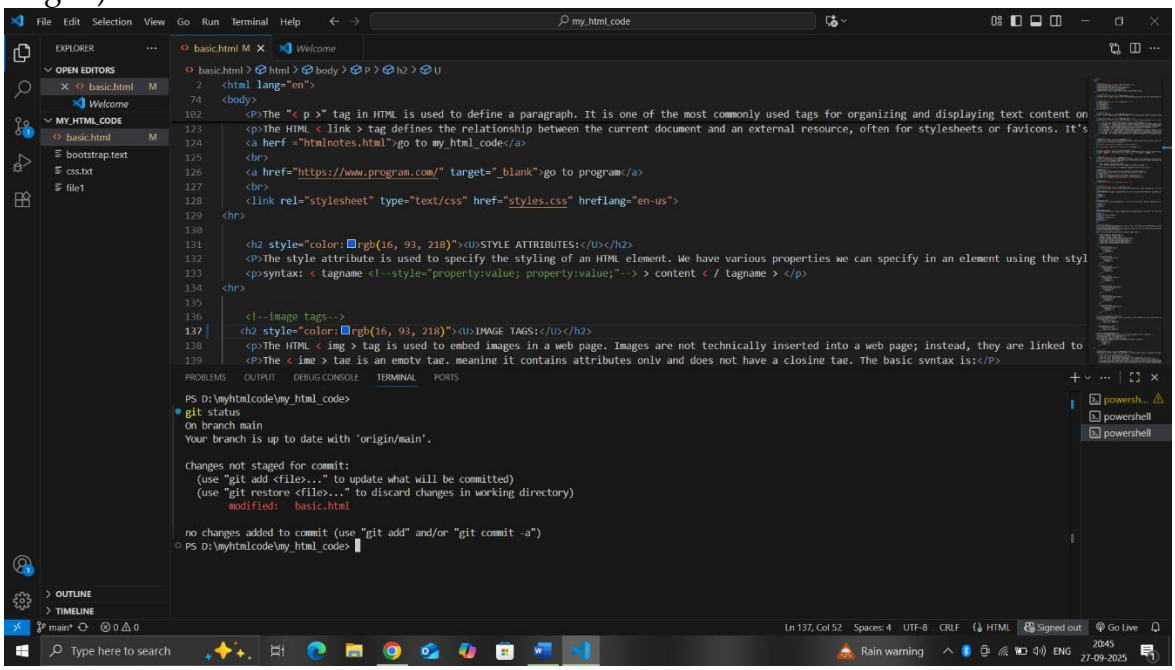
When we open the commits in cloud repository will get all the commit messages. It shows all the changes that we had made so far. Just open one commit message it shows us that what we have given before and what we have changed after this is called as git diff.



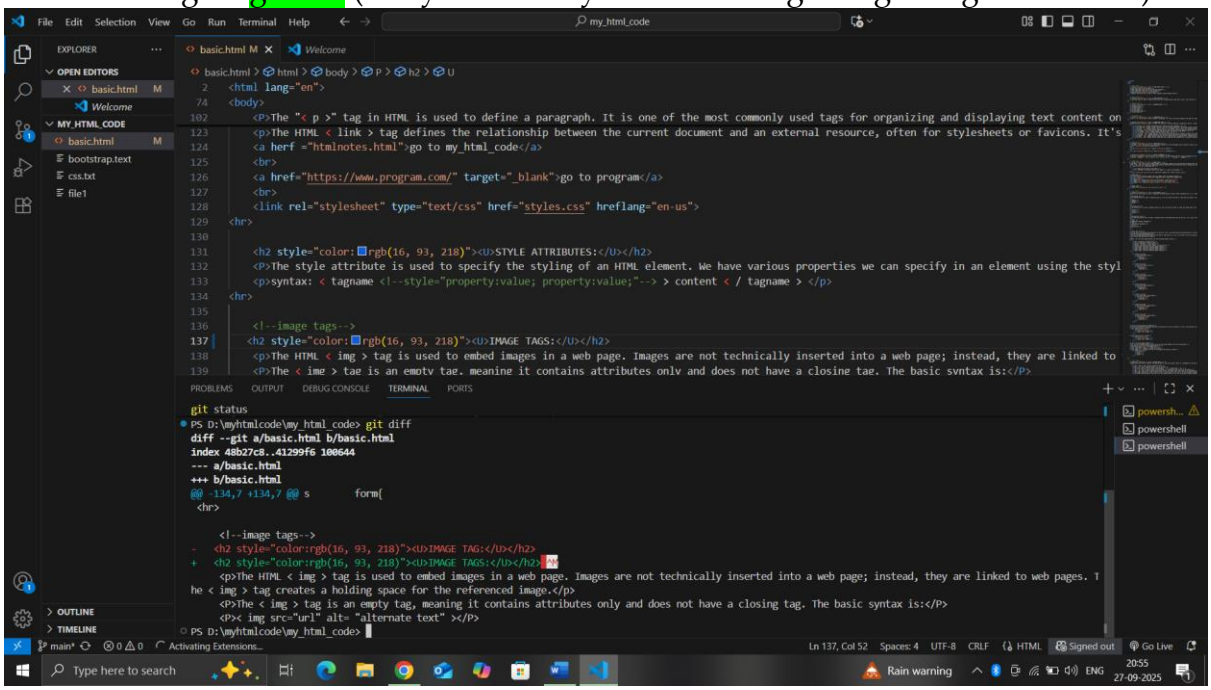
Let's see about git diff command:

➔Just do some changes in local repository and save it its shows modified file symbol M.

→Go to terminal →enter git status (its shows this file has been modified but not yet staged).



→After that give git diff (it says that you have changed tag to tags in this file).



This is basically git diff command.

→Still it is modified file we need to change it as a tracked file→ just enter git add . without mentioning file name.

When to use Git add.

- You know whatever changes you have made and you also want to push all the file then you can use this command git add.
- "." means your current folder and "./" also means current folder.
- You're saying that take everything inside this folder.

- If you give git add every file will be added, then give commit and git push.

18. BRANCH (WORKING WITH OTHER DEVELOPERS)

(Let's assume one person working as a backend developer name "ajay" and another person working as a frontend developer name "abi". Both of them have to work alone. Why? Because, his code must not affect to her and her code must not affect to him.
When he started coding he would have different user interface. And when she was started she will have different backend. Both of them will have perfectly working code.
She know what mistakes she will do in her code and also he will know his code.
What both of them will do is push code to a common place.
Now he will pull the latest code from there as well her code also. He will have her faulty code and she will also download his faulty code. Both of their code will not work. Both of them will get confused.)
To avoid this there is a concept known as branching.
She will create front end branch and work on that, He will create a branch called backend and he will work on that.
So that his faulty code will not affect her and her code will not affect him.
After merging both of the branches both of them code will work properly.
At that time they will clone from their main branch and they will use their code.
We will try to simulate this one:

- Git branch will show you the current branch you are in.
- We can use git checkout -b "branch name" command to create a new branch.
  Ex: git checkout -b "frontend", now you're in frontend branch.
- If I want to switch back to main branch, I will give git checkout main. If again I want to back to frontend branch, then will give git checkout frontend.

→Now we have main branch, frontend branch and backend branch. Some person will maintain main /master branch like manager or maintainer.
→Frontend branch will maintain by one developer and backend branch will maintain by another developer. After whatever changes made by developer they have to add, commit and push into their branch.
→ Frontend branch contains some file changes and back branch will contains some backend file changes. All the person can work together in main branch but in the main branch both the backend and frontend changes are not present.
→Here is the concept called as compare and pull request means my boss/ manager will usually maintain the main and master branch. So, you will make a request to those persons, you will tell them you made a frontend change and you want them to accept it same as another person also do. When both are pushing, that person will review our code and accept it accordingly.

19. MERGE/ PULL REQUEST:

- Now I am clicking on compare and pull request in GitHub.it is automatically comes in your GitHub page when you make changes in your file within your branch.
- You will get add title and add a description (write some message) →then click create pull request.
- Now the boss will work in his repo →click the pull request → just click on the latest pull request. Here the boss can see what the developer is done and they made changes by clicking on files changed.
- If it is all ok then they click merge pull request and then confirm merge or if it is not ok then write some message in add a commend and then click on close pull request.

20. MERGE CONFLICT:

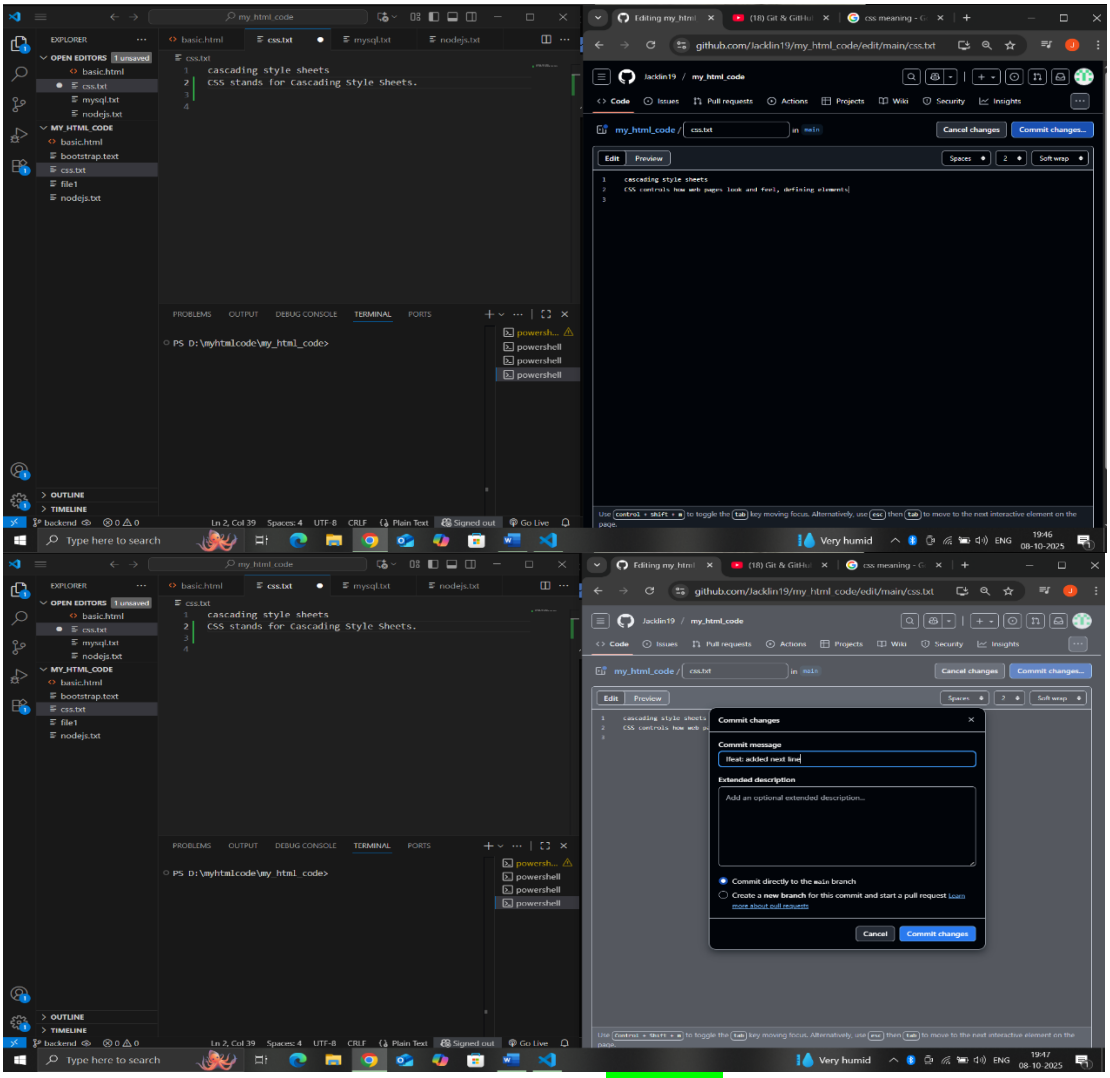
A merge conflict occurs in Git when a merge or rebase operation fails because the same lines of a file were changed differently in the branches being combined. Git cannot automatically decide which changes to keep, so it pauses the process and requires a human to manually resolve the conflict.

Common causes of merge conflicts

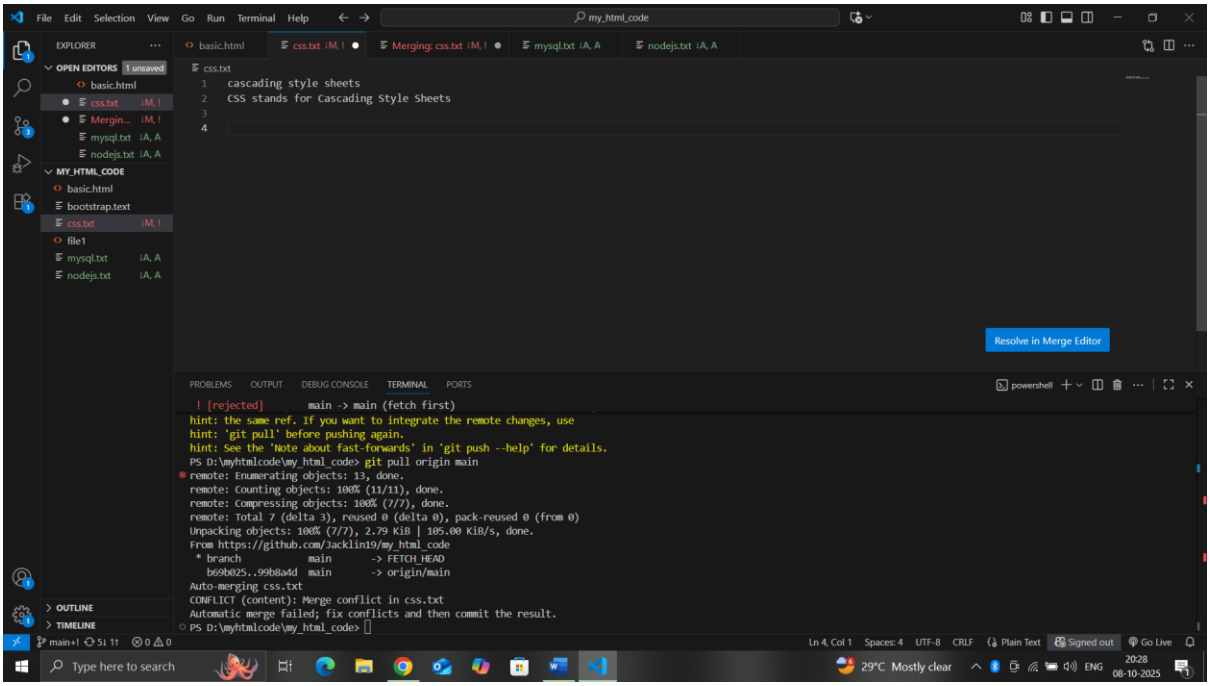
- **Competing line changes:** Two branches modify the exact same lines in the same file with different content. Git, unable to reconcile the competing changes, marks the file as conflicted.
- **File modified and deleted:** One branch edits a file while another branch deletes the same file. Git does not know whether to keep the edited file or honour the deletion.
- **Same-file renaming or deletion:** A file is renamed in one branch and modified or deleted in another, making it difficult for Git to track.

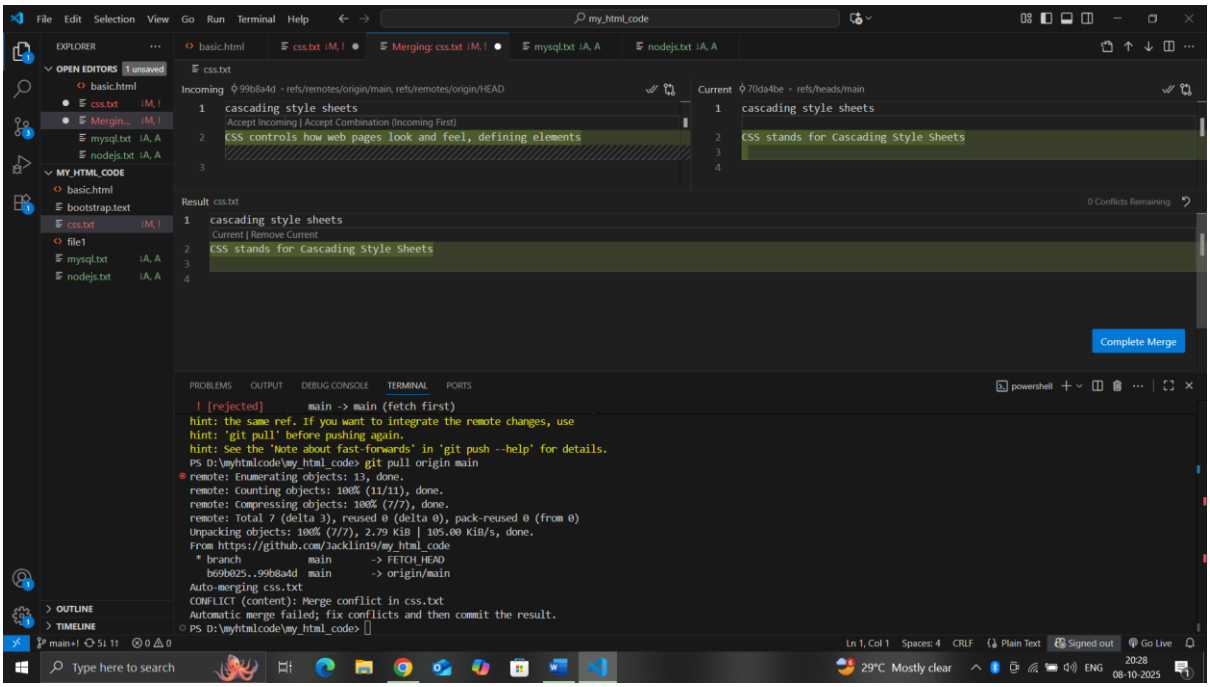
Example:

- Now one person going to make a changes in cloud repository file and made commit, meanwhile another person made changes in same file in local repository.



- Go to local repository now give command git push after add and commit.
- You will get some conflict error message because some person made changes in cloud repository same file you had made some changes so you need to pull it from cloud repo.
- Now both of them need to discuss which one need to change or select important code click resolve in merge editor and click complete merge.
- Then do all the commends which need to push it in cloud repo and go and check it in cloud repository.





Resolve conflicts on the GitHub website

This method is suitable for simple merge conflicts caused by competing line changes in a file.

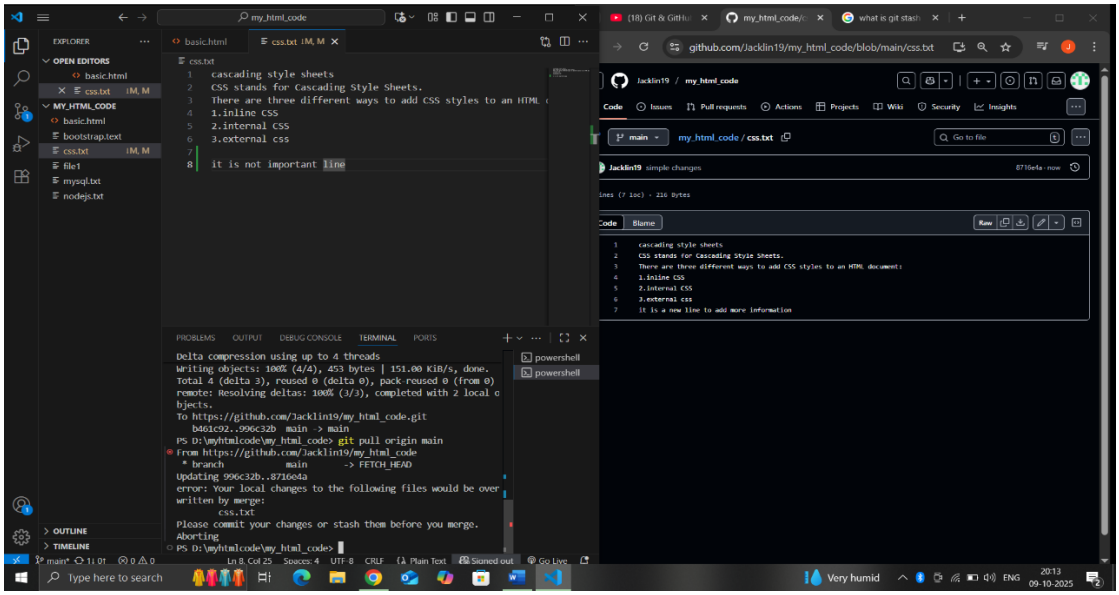
1. **Open the pull request:** Navigate to the pull request with a merge conflict. You will see a "Resolve conflicts" button near the bottom of the page. If this button is disabled, the conflict is too complex to resolve in the web editor, and you must use the command line instead.

2. **Edit the file:** Click **Resolve conflicts** to open the web editor. Inside the editor, you will see conflict markers (<<<<<<<, =======, >>>>>>>) that highlight the conflicting sections of code.
3. **Choose the changes:** Decide which version of the code you want to keep.
    - **Keep your changes:** Select **Accept current changes**.
    - **Keep the other branch's changes:** Select **Accept incoming changes**.
    - **Keep both:** Manually edit the code between the conflict markers to merge the desired changes.
4. **Mark as resolved:** After resolving the conflict for a file, click **Mark as resolved**.
5. **Commit the merge:** Once all conflicts are resolved, click **Commit merge** to finalize the process. If needed, you can create a new branch to contain the changes.

21. GIT STASH:

git stash is a command that temporarily saves, or "stashes," your uncommitted changes in Git. It is used when you want to switch context and work on something else, but your current changes are not ready to be committed. It's like a recycle bin it will save all the stash one by one like stack method. If it is any emergency, you can restore it.

→Make some changes in both the repository in the same file don't commit the local repo. Only give commit changes to cloud repo and then pull it. You will get a message aborting.
(Means its given warning to you that "you didn't commit the file it's going to stash just save it as soon as possible".

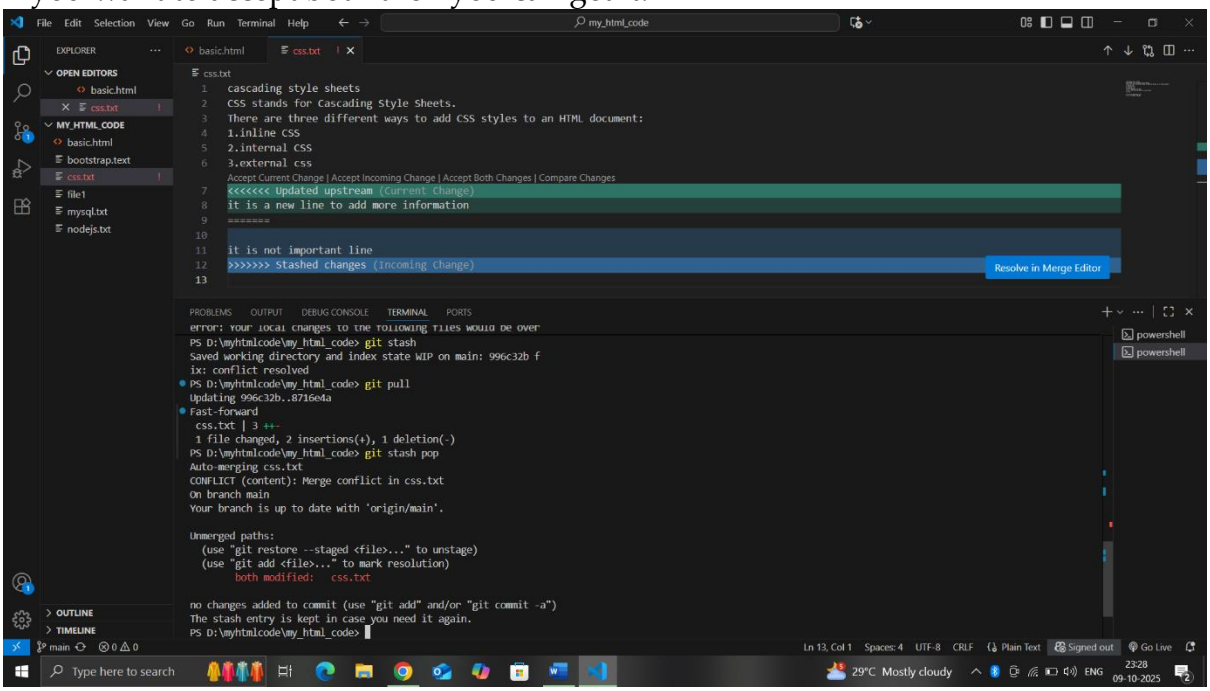


Then give this command:

- Git stash 'file name' (if you want to through in the recycle bin whatever you changed in your file if it is not important)

- Git pull (it will give code which is created by another person)

- If you want to retrieve your stash which is through in the dustbin by giving this commend:
  Git stash pop
- Now you have already changed code as well as your retrieved code both are in the same line. Here also you will get conflict. It will ask which change you need, if you want to accept both then you can get it.



- After accepting the complete merge as usual you give add, commit and push command.

22. HOW TO CHANGE THE BRANCH NAME FROM TERMINAL

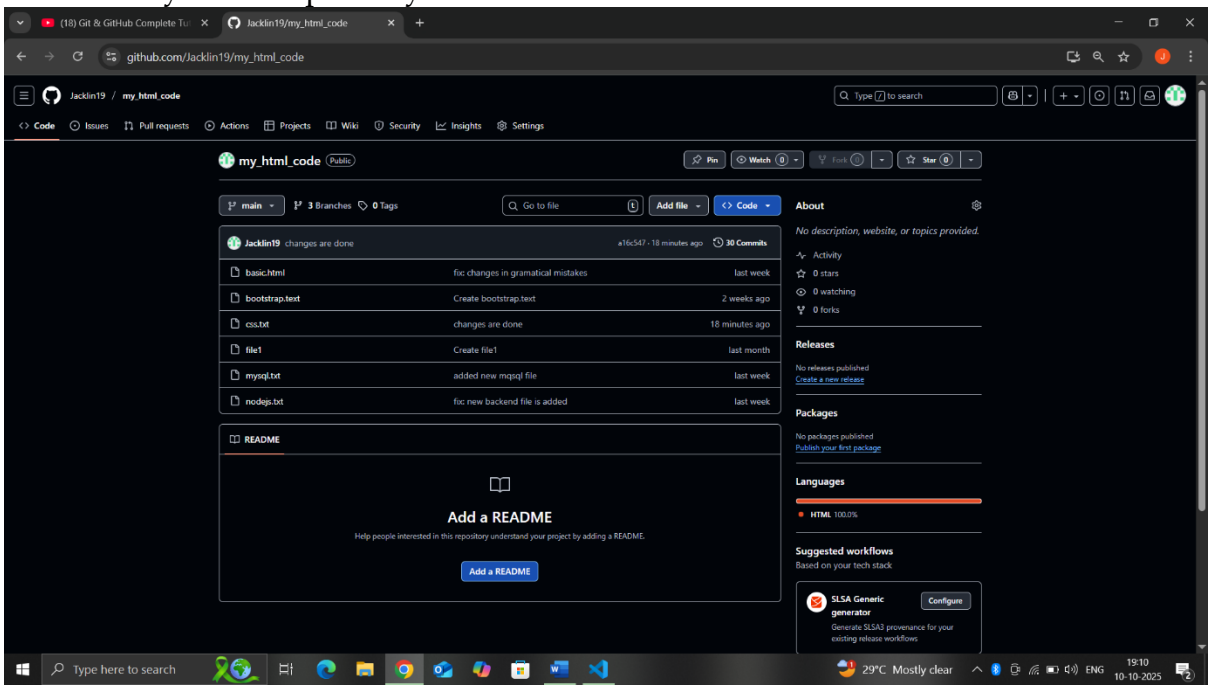
In local repository:
→git checkout -b branch1(creating new branch)
→git branch (it shows all the branches with green colour of current branch)
Ex: I have 3 branches name branch1, branch2 and main
Branch1
Branch2
main
→git branch -M "frontendbranch" (rename the current branch)

These commends are only for local repository it shows whatever branches we have, if we want global or cloud repo branches, then follow next commend.
→git branch -a (shows all the branches whatever there in local and cloud)

Note: if you had made any changes in cloud or local repo, you have to use pull and push commends even though for adding new branch.

In cloud repository:

→This is my cloud repository I have 3 branches here.

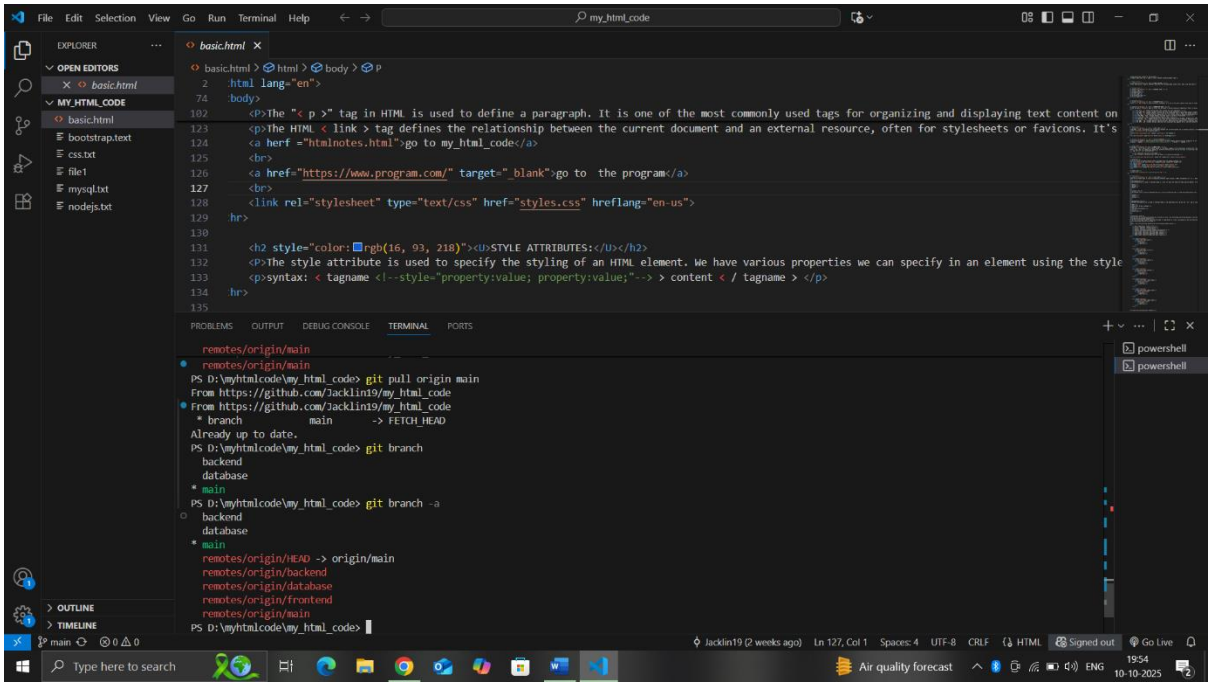


→Click 3branches below in your repository name →click new branch →give new branch name → click create new branch.

Or

→Click main dropdown button → enter new branch name in find or create a branch (ex: frontend)→ click create branch frontend from main (now you will get 4 branches in cloud repository).

→Then git pull after creating new branch in cloud repo and check out in terminal.



23. GIT LOG COMMAND

The git log command displays the commit history of a Git repository. By default, it lists commits in reverse chronological order, with the most recent commit appearing first.

Each commit entry in the default git log output typically includes:

- **Commit SHA-1 checksum:** A unique identifier for the commit.
- **Author:** The name and email of the person who authored the changes.
- **Date:** The date and time the commit was made.
- **Commit message:** A description of the changes introduced in the commit.

How to navigate the log:

- The output of git log can be long, so Git uses a terminal pager, usually less, to display it.
- Scroll down line by line: Press j or the down arrow key.
- Scroll up line by line: Press k or the up-arrow key.
- Scroll down page by page: Press the Spacebar or Page Down.
- Scroll up page by page: Press b or Page Up.
- Exit the log: Press q.
- You can also search within the log by typing / followed by your search term and pressing Enter, then pressing n to jump to the next match.

24. OPEN SOURCE

(We can say community driven software

Ex: 1. apple is not open source, only they can use their code in their system. It is called preparatory code or closed source.

2. Git lab is a clone of git hub it has same function what git hub using. It is open-source software.
3. Android is also open-source software. Samsung, Google also using this software which is created by Google for free anyone can modify this code. It is called open source.
4. This open-source code can be published in git hub we can download it in git repository.

Ex: git hub→search android (we can choose which is best for android) and use sort by: most stars for best one and copy or download it. This is not only for window can we take for Linux also.)

Open source on GitHub refers to a public-facing project where the source code is freely available for anyone to view, use, modify, and distribute. GitHub is the most popular platform for hosting open-source software, which allows developers and organizations to collaborate on projects transparently and securely.

What defines an open-source project?

For a GitHub repository to be truly "open source," it needs to be made public and include a compatible open-source license.

- **Public Visibility**: The repository must be accessible to anyone on the internet, not just collaborators invited by the project owner.
- **Open Source License**: A license file, such as MIT or Apache 2.0, grants users legal permission to use, modify, and redistribute the code. Without an explicit license, a project is typically copyrighted and cannot be used by others without permission.

How to get started with open source on GitHub

For new projects

If you are creating a new open-source project from scratch:

1. **Create a public repository**: When creating a new repository on GitHub, make sure the "Public" option is selected.
2. **Add a license**: GitHub offers the option to choose a license (like MIT, Apache 2.0, or GPLv3) when you initialize a new repository. This step is crucial for defining the terms of use.

3. **Include essential documentation**: In addition to a license, a complete open-source project should have:

1. **README.md**: Explains what the project does, how to use it, and how to get started.

2. **CONTRIBUTING.md**: Provides guidelines for how other developers can contribute to the project.

3. **CODE_OF_CONDUCT.md**: Establishes behavioural standards for the community.

For contributors

If you want to contribute to an existing open-source project:

1. **Find a project**: You can find projects by using the GitHub Explore feature or searching for topics like #good-first-issue to find beginner-friendly tasks.
2. **Fork the repository**: Create your own personal copy of the project by clicking the "Fork" button on the project's page. This is where you will make your changes.
3. **Clone the project**: Copy your forked repository to your local machine using the command git clone.
4. **Create a new branch**: A new branch keeps your changes isolated from the main project. Use git checkout -b <branch-name>.
5. **Submit a pull request (PR)**: When your changes are ready, push them to your GitHub fork. Then, open a pull request from your fork to the original repository. The project maintainers will review your work before merging it.

Why open source on GitHub?

For maintainers

- **Encourage collaboration**: GitHub provides the tools (issues, pull requests, project boards) to manage contributions from a global community.
- **Improve project quality**: Public scrutiny and peer review from many developers can lead to more robust, secure, and innovative code.
- **Showcase work**: Hosting a project on GitHub provides maximum visibility and can attract new users and contributors.

For contributors

- **Learn and build skills**: Contributing is one of the best ways for developers to learn from experts and gain real-world experience.

- **Build a portfolio**: A public GitHub profile showing open-source contributions serves as a powerful portfolio to attract potential employers.
- **Help others**: You can contribute to projects you care about, fix bugs, and improve tools that benefit the entire developer community.

25. README IN GITHUB AND LEARN MARKDOWN SYNTAX

(If I just want to explain about my repo to my colleague then I can use .md file (markdown file).

Just create a file name like readme.md then we have to use markdown format (to learn markdown cheat sheet))

A README is a documentation file included in a GitHub repository that provides an overview of a project. Written in Markdown, a simple mark-up language, it is the first file visitors see on the repository's main page and is crucial for explaining the project to users and potential collaborators.

A README file is a text document that contains essential information about a program, utility, or game. It typically includes instructions, additional help, and details regarding patches or updates.

Key purposes of a README

- **Introduces the project:** Tells visitors what the project is, what it does, and why it is useful.
- **Provides instructions:** Includes details on how to install, set up, and use the project, often with code examples.
- **Outlines contribution guidelines:** Informs other developers on how they can contribute to the project, such as by submitting issues or pull requests.
- **Specifies licensing:** Declares the license under which the project is distributed, which tells others what they can and cannot do with the code.
- **Enhances user experience:** A clear, well-structured README makes a project more accessible, understandable, and appealing to a wider audience, including those viewing your work for a resume.

Profile README

In addition to project-specific README files, GitHub also allows users to create a special profile README.

- This is a public repository with the same name as your GitHub username.

- The contents of its README.md file are displayed prominently at the top of your public profile page.
- This allows you to create a personalized, dynamic resume or portfolio to introduce yourself, highlight your skills, and showcase your best work.