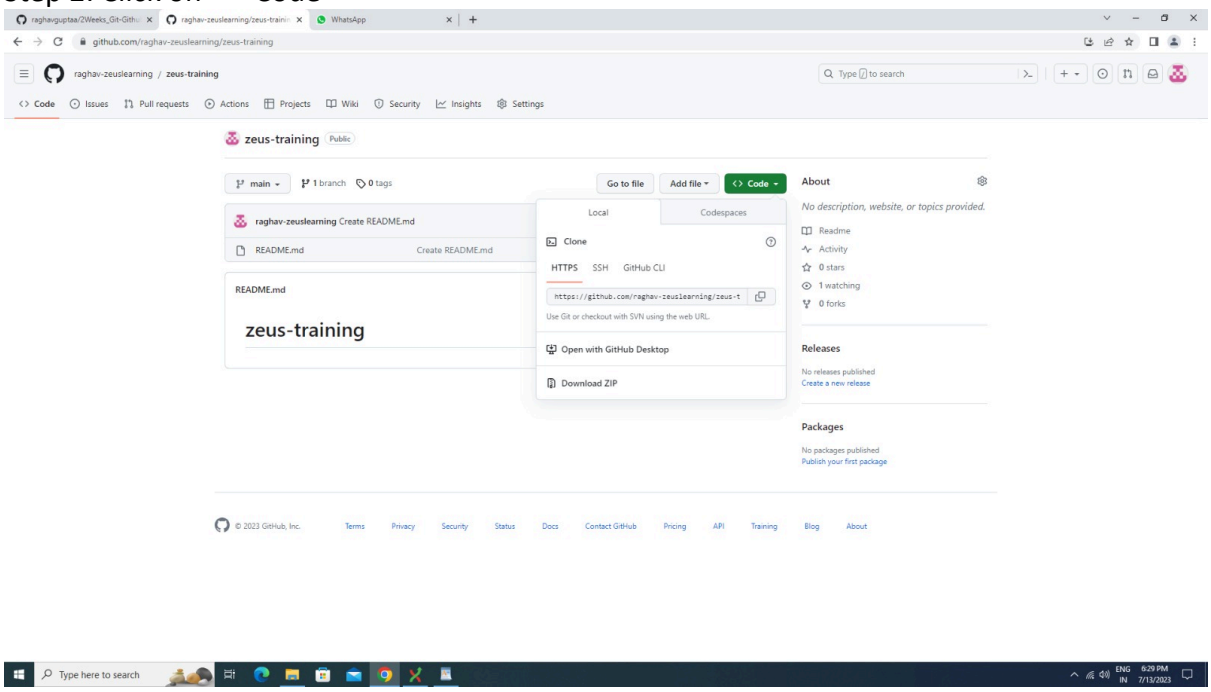
Things need to learn for Git:

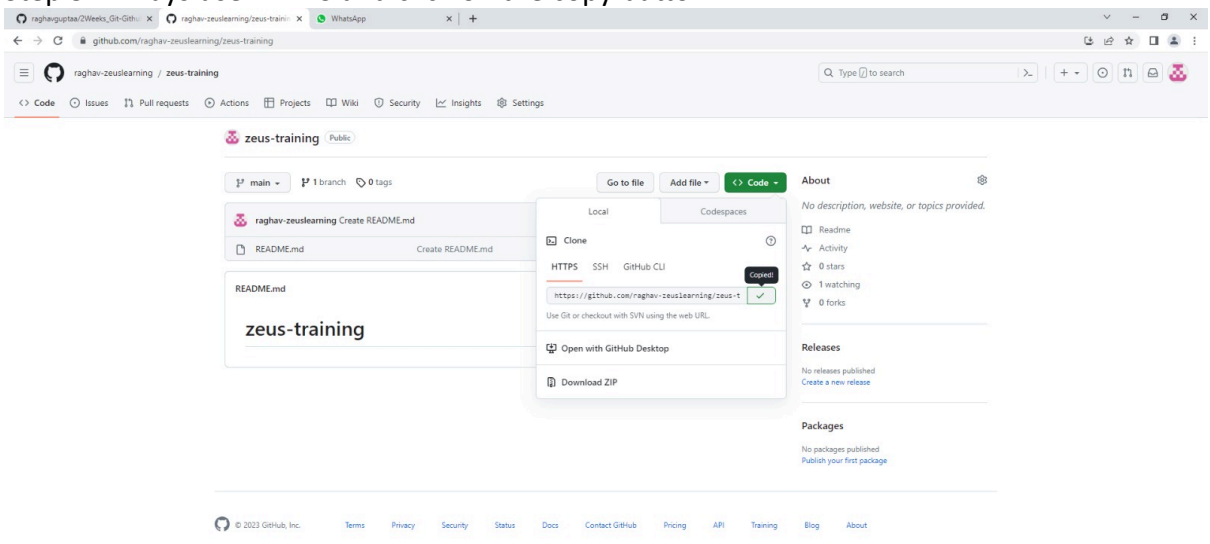
1. Cloning ✅
2. Commit & Push✅
3. Git Branches ✅
4. Git Merge ✅
5. How to invite other git users ✅
6. Git Fetch vs Pull ✅
7. Pull Request ✅
8. What is merge conflict. ✅

1. Cloning

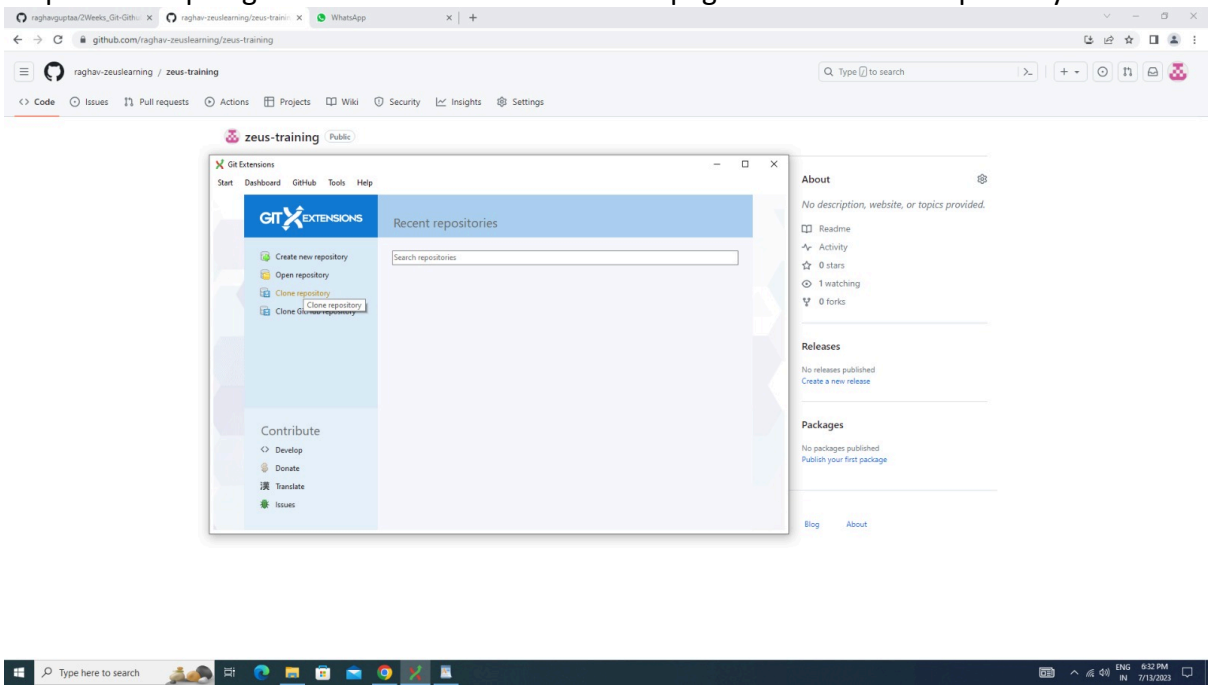
Step 1: Create a GitHub repository/or go to the repository which need to be cloned.
Step 2: Click on “<>Code”



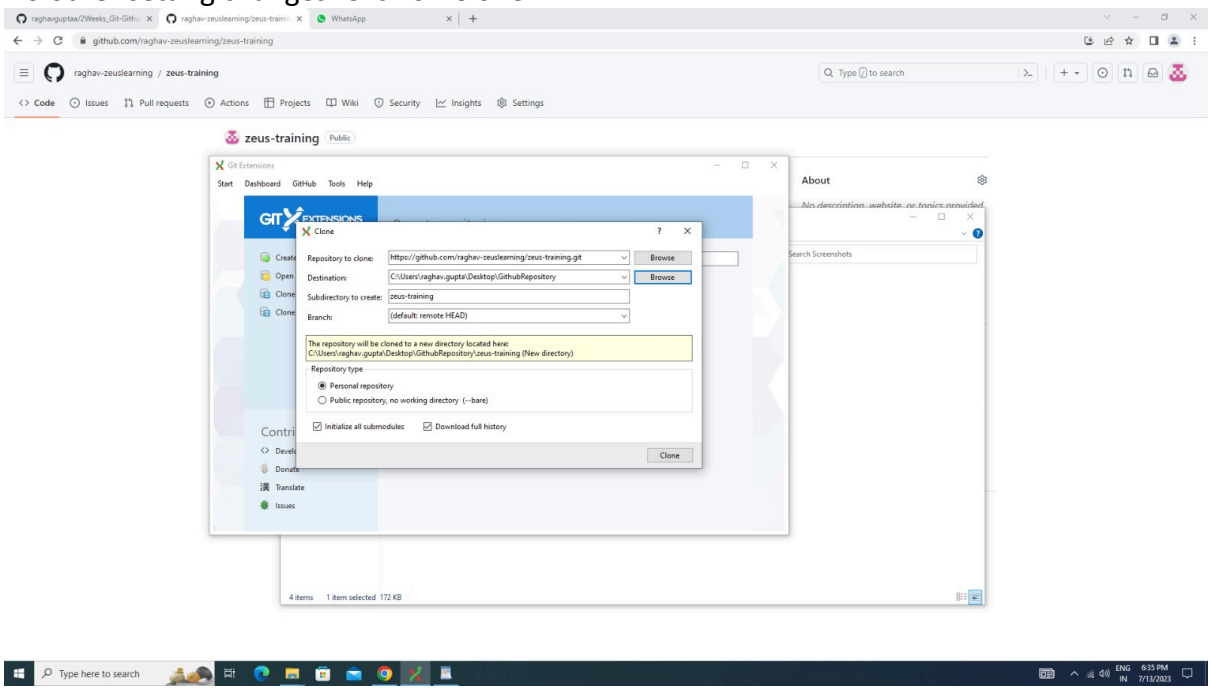
Step 3: Always use HTTPS and click on the copy button.



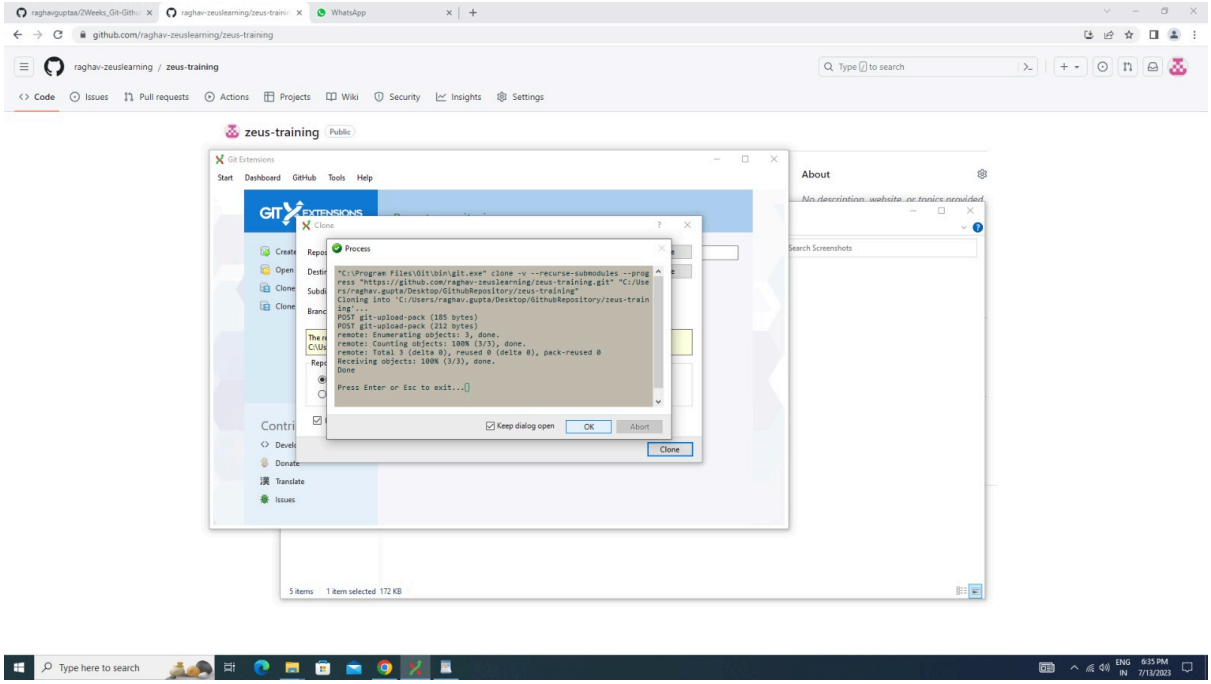
Step 4: Now open git extensions. And on the home page click on Clone Repository.



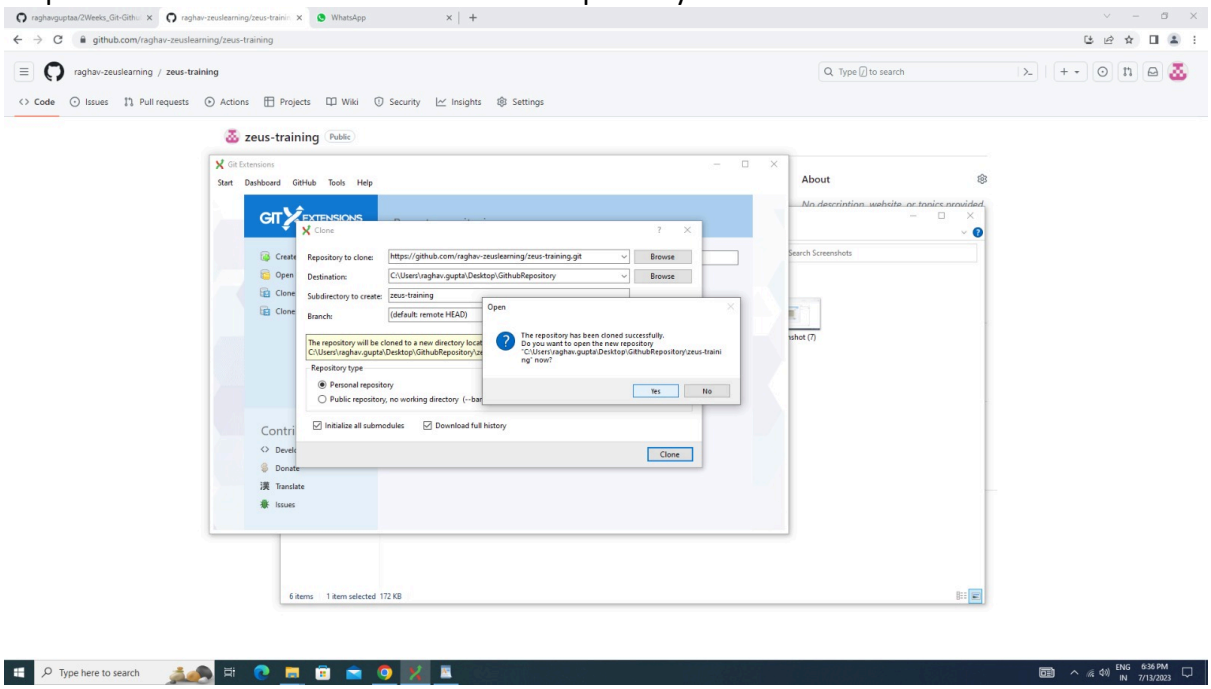
Step 5: Now,
Paste the copied link in the “Repository to Clone”.
Destination: Where you want to locally clone the repository and work on it.
Its better to make folder in the Desktop>New Folder>GithubRepository(No Spaces)
Subdirectory: Name of the Github Repository. (It will automatically come)
No other setting changes. Click on Clone.



Step 6: This should be the next popup. If everything goes right.



Step 7: Click on Yes to work on the GitHub repository now.

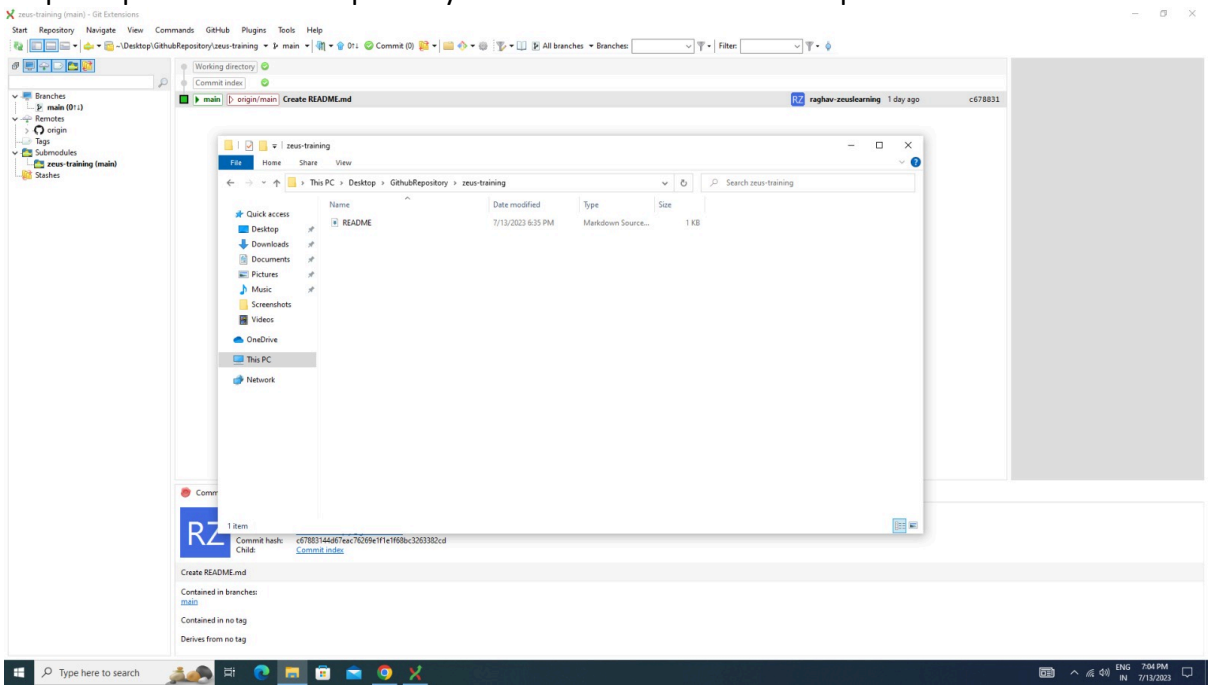


Cloning is Completed.

2. Commit & Push

Commit means Updating the code and then sending the new code to the GitHub server.
Push means whatever is committed on the local machine, put it on GitHub.

Step 1: Open the GithubRepository folder which is on the desktop.

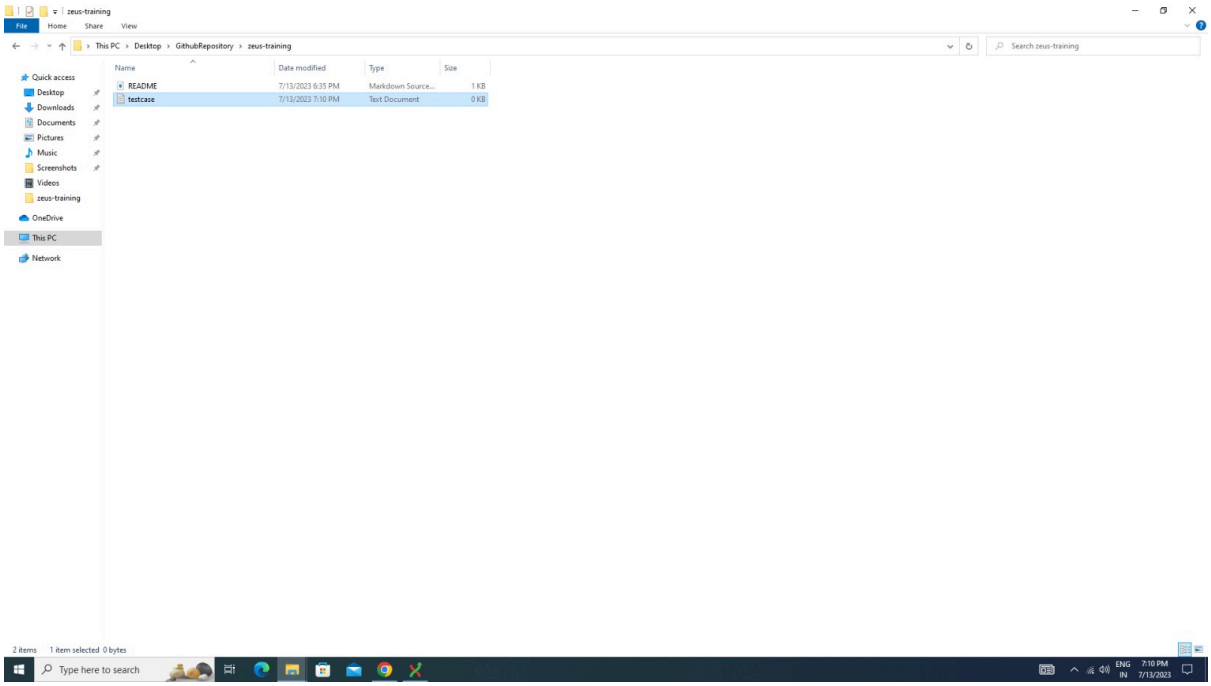


I only had README file. So, it's only showing README. Whatever you had it will show the same.

STEP 2:

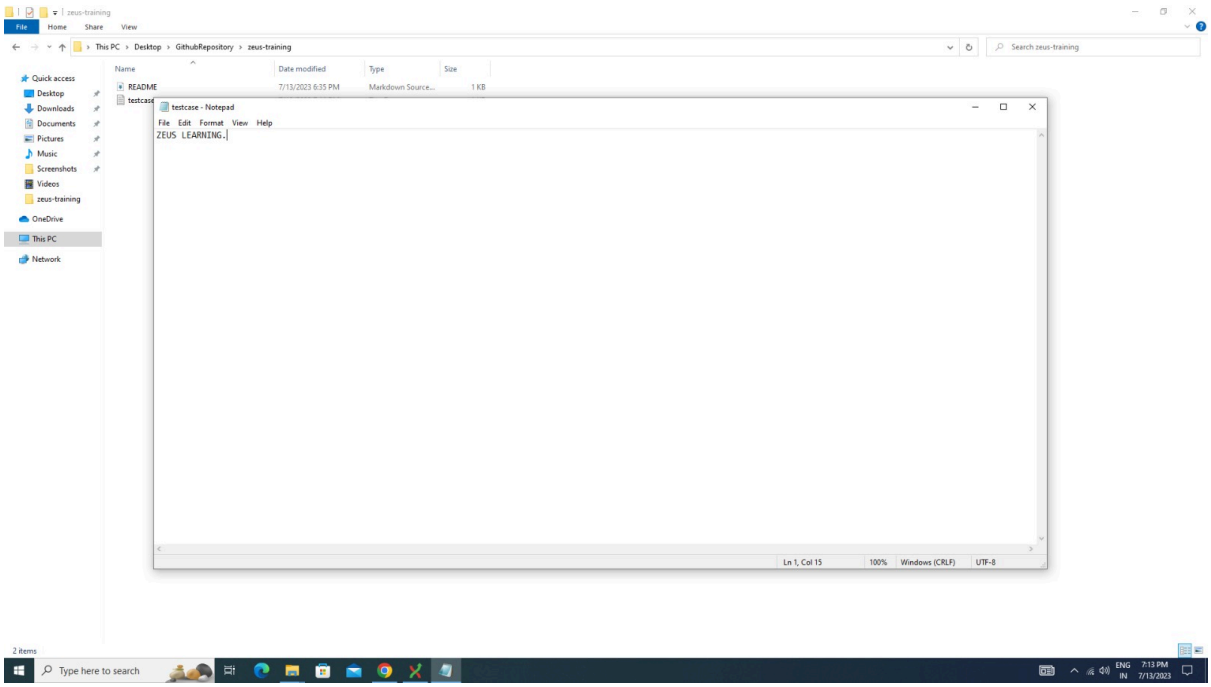
Now we will add a sample test file and then commit it.

I added a New>Text Document then name it testcases.



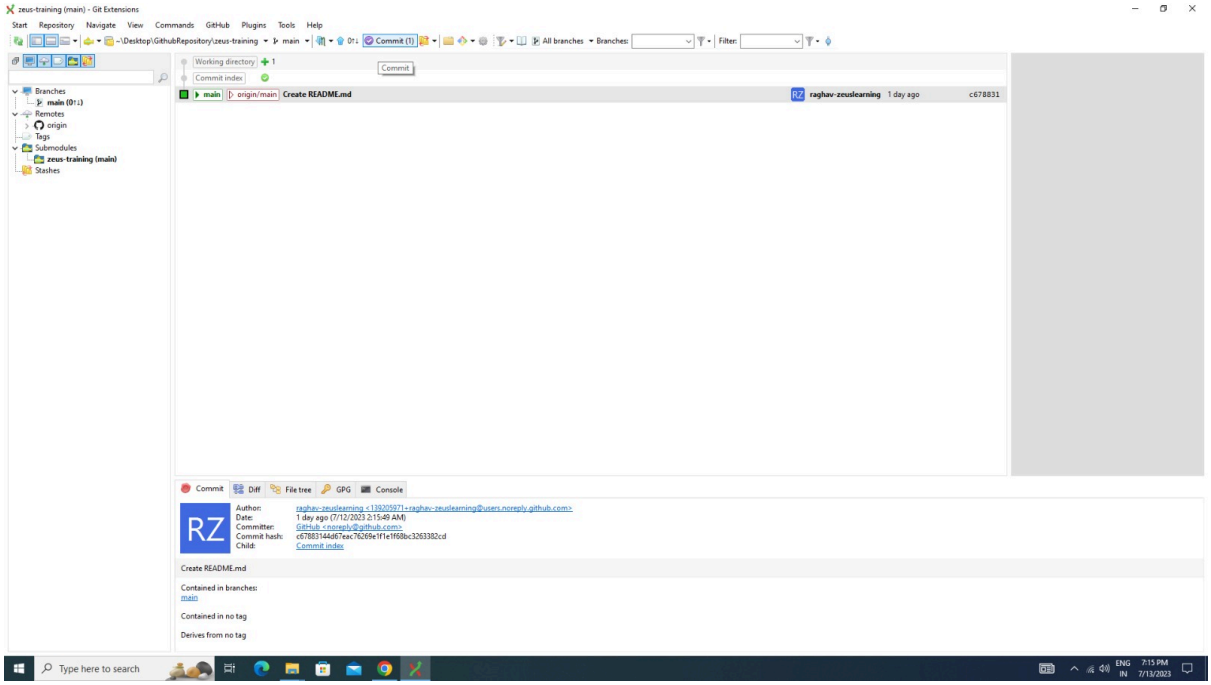
STEP 3:

In that file I wrote ZEUS LEARNING and then saved it.



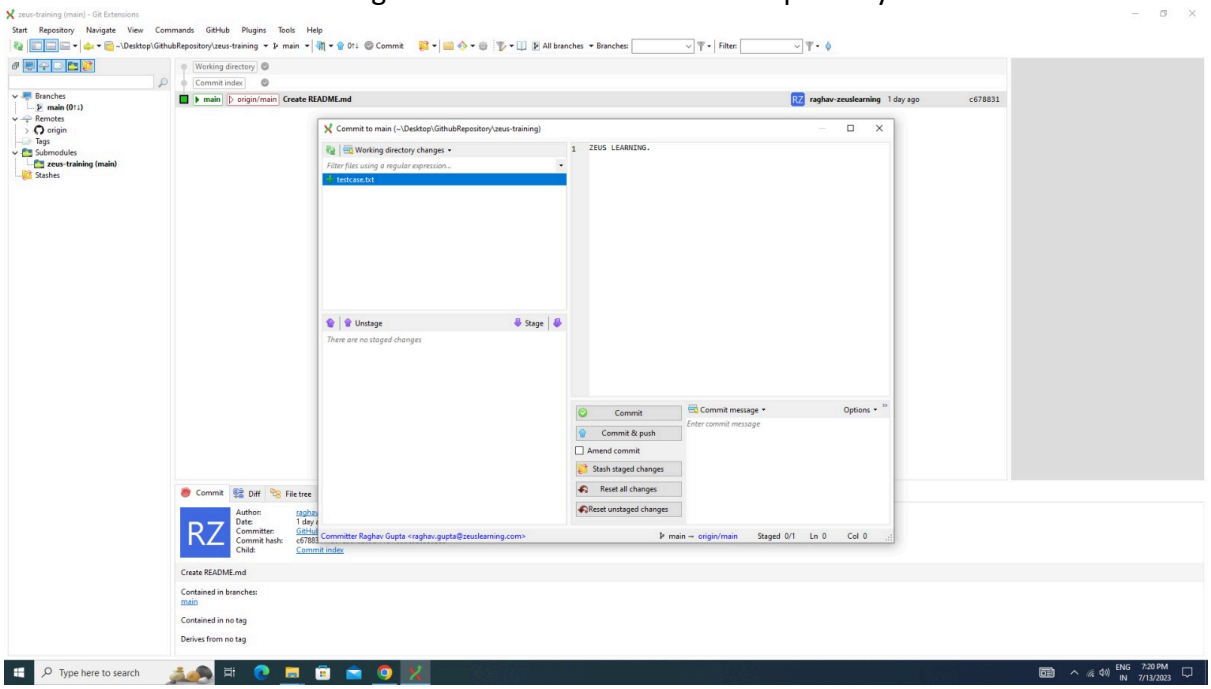
STEP 4:

Since I added a new file. I can see there is 1 thing to commit in the git extension application.

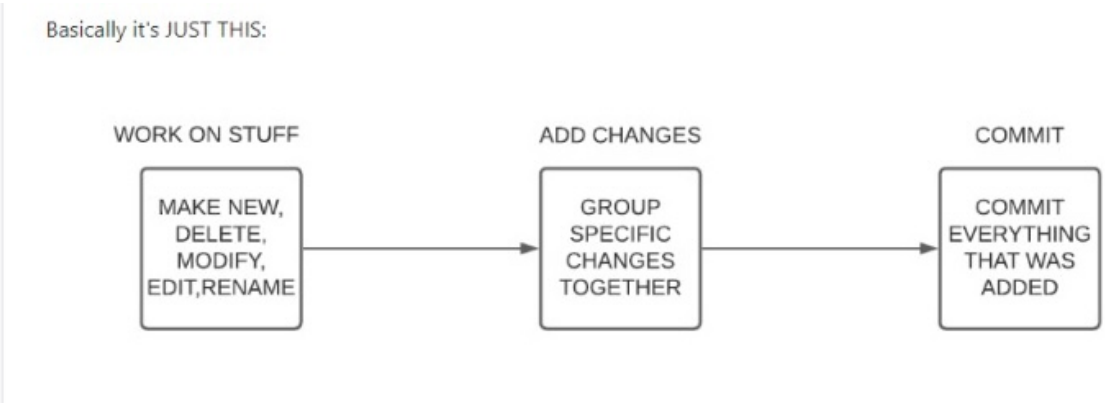


Step 5:
Click on commit.

Now we will see all the changes we had done on the local repository.



Now we need to understand the committing workflow.

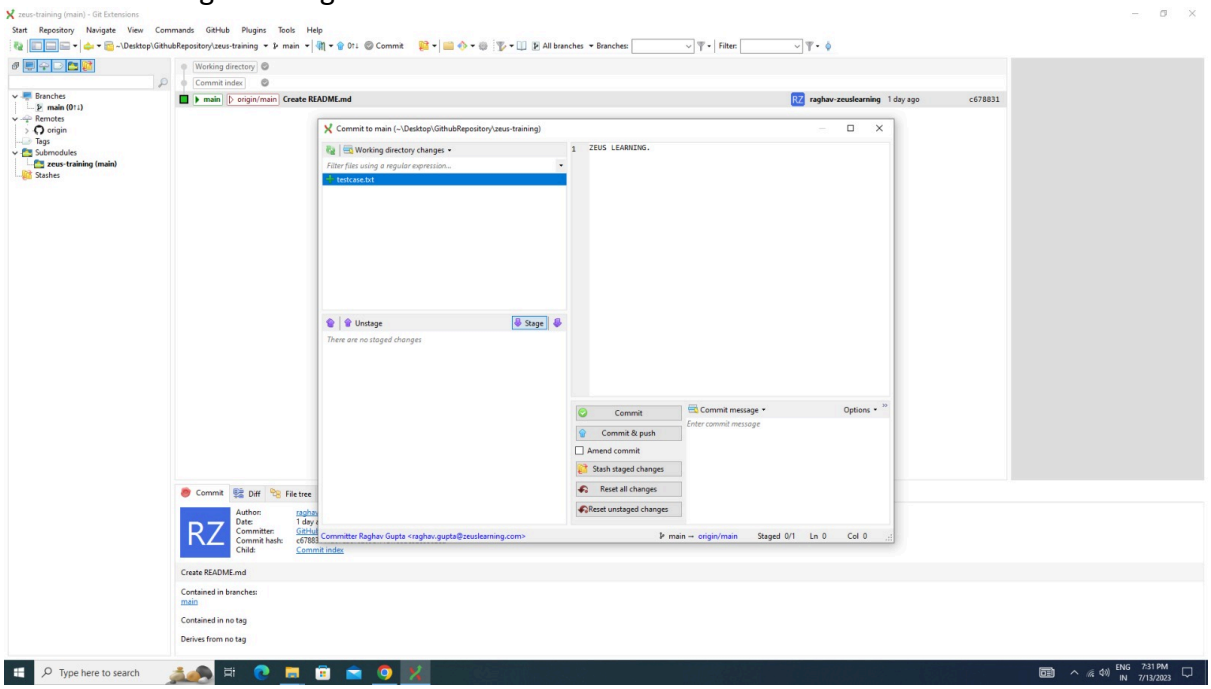


We work on some stuff like Create, Delete, Modify, Edit or Rename.
Then we add changes which means we “STAGE THE FILES” then we commit.

Committing is a 3 Way Process:
First: Work on some stuff.
Second: Add the stuff to the staging area. (LOCAL)
Third: Commit the Staged Changes to the repository. (Not visible on GitHub)
NOTE: Staged Changes and Committing are local to the machine, you need to PUSH then only you can see the changes on the GitHub.

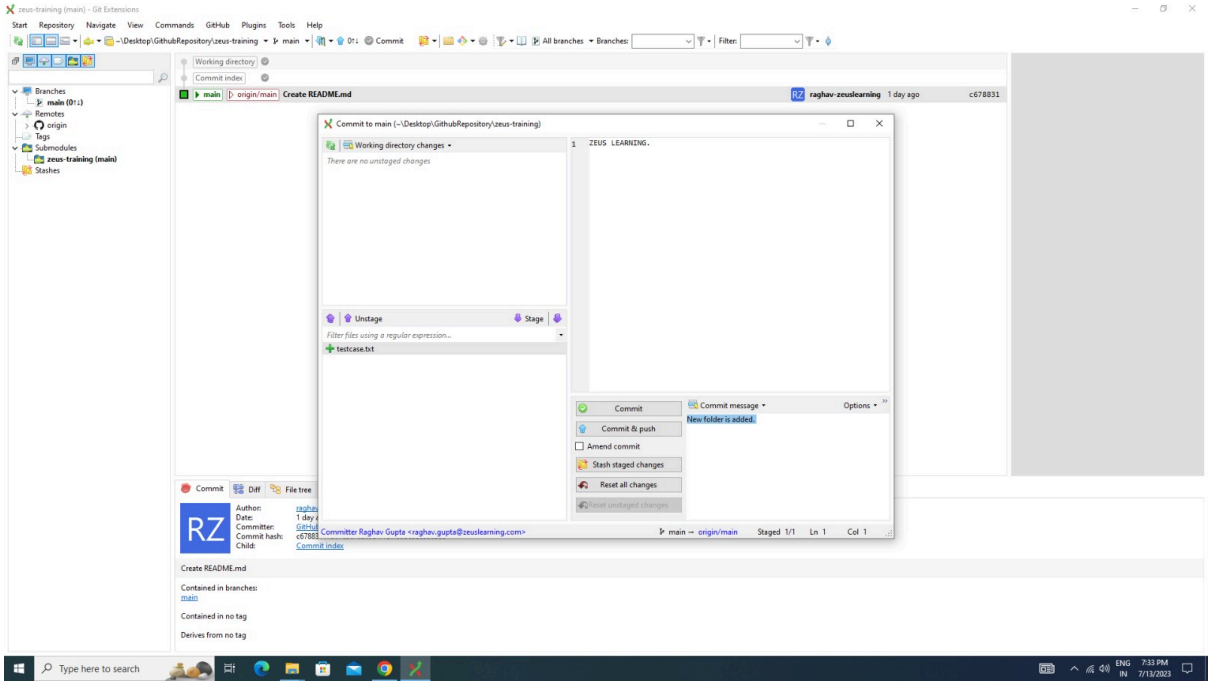
STEP 6:

We click on Stage. To stage the files.

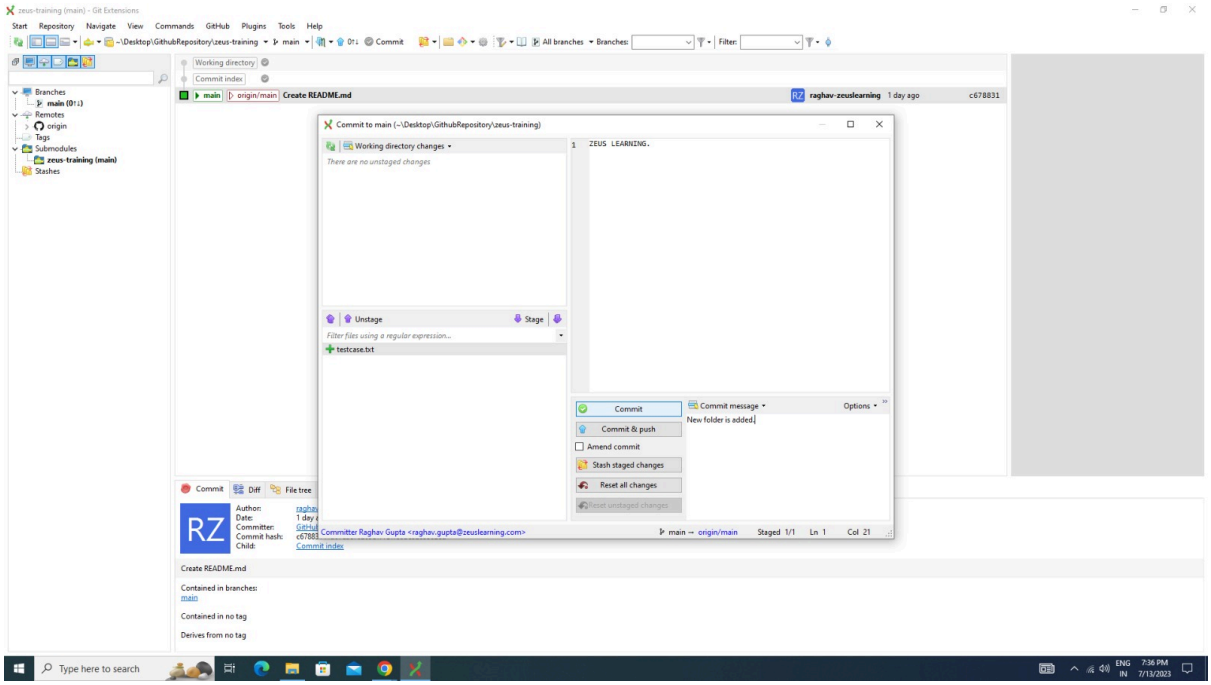


STEP 7:

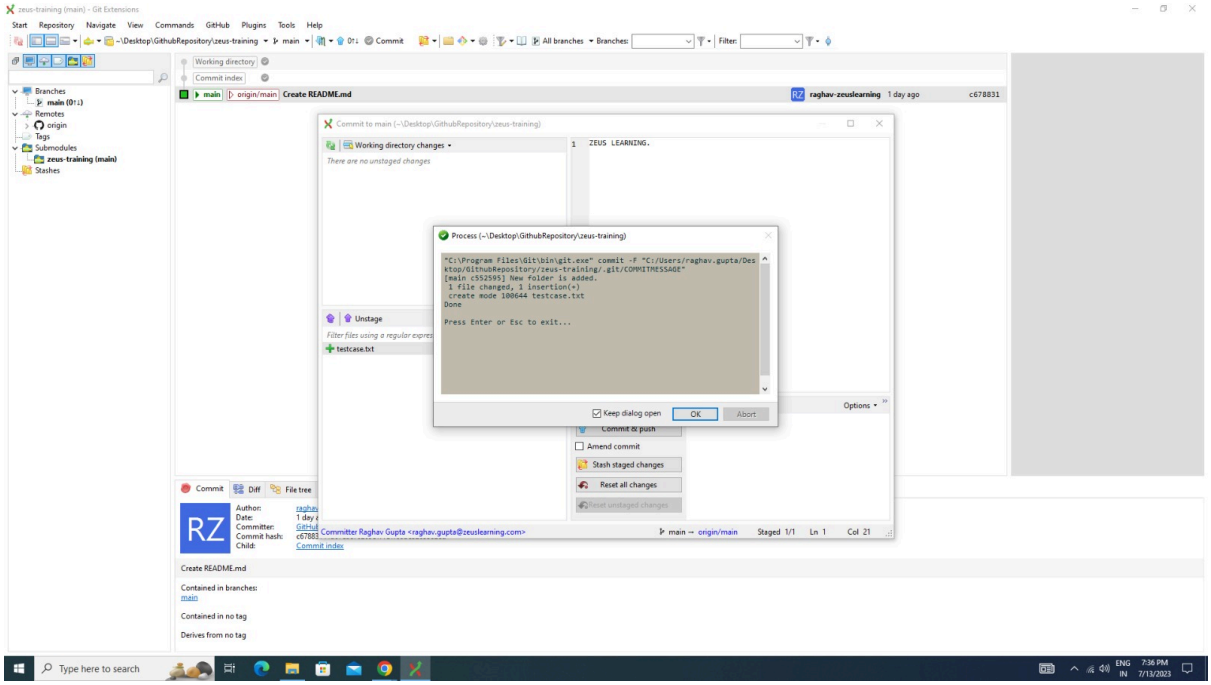
Then we write the “Commit Message”



STEP 8:
Click on commit.

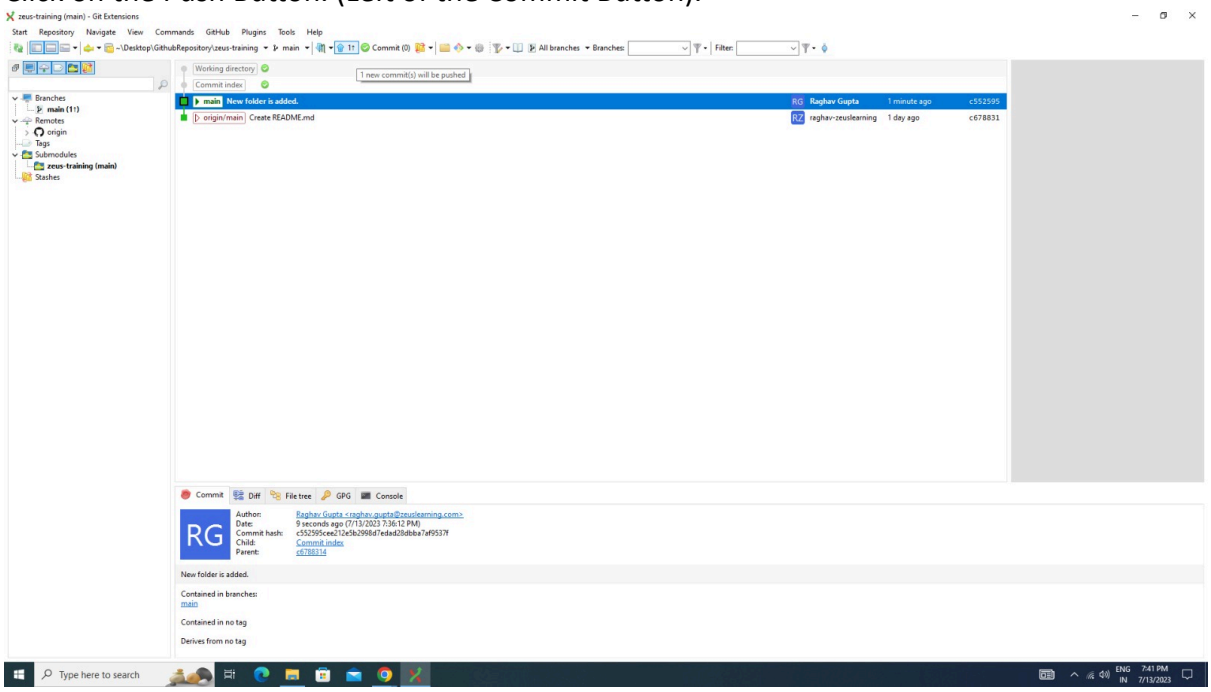


STEP 9: NOW COMMIT IS DONE.

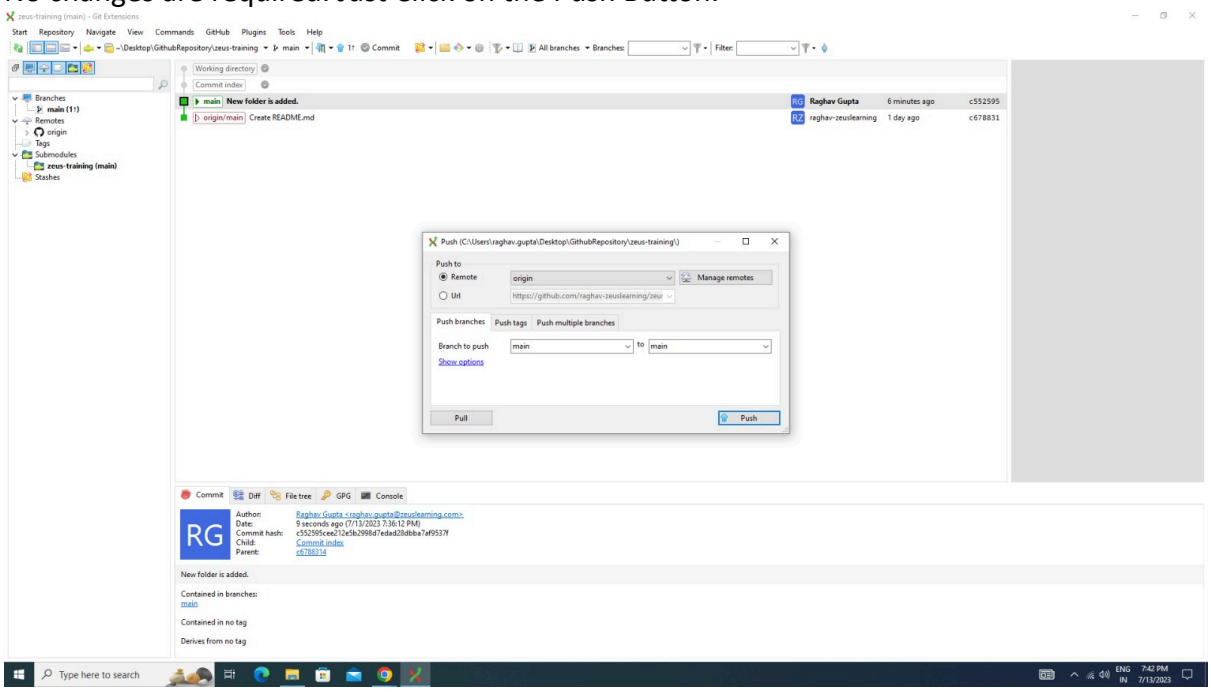


STEP 10:
Now we will understand how to Push.

Click on the Push Button. (Left of the Commit Button).



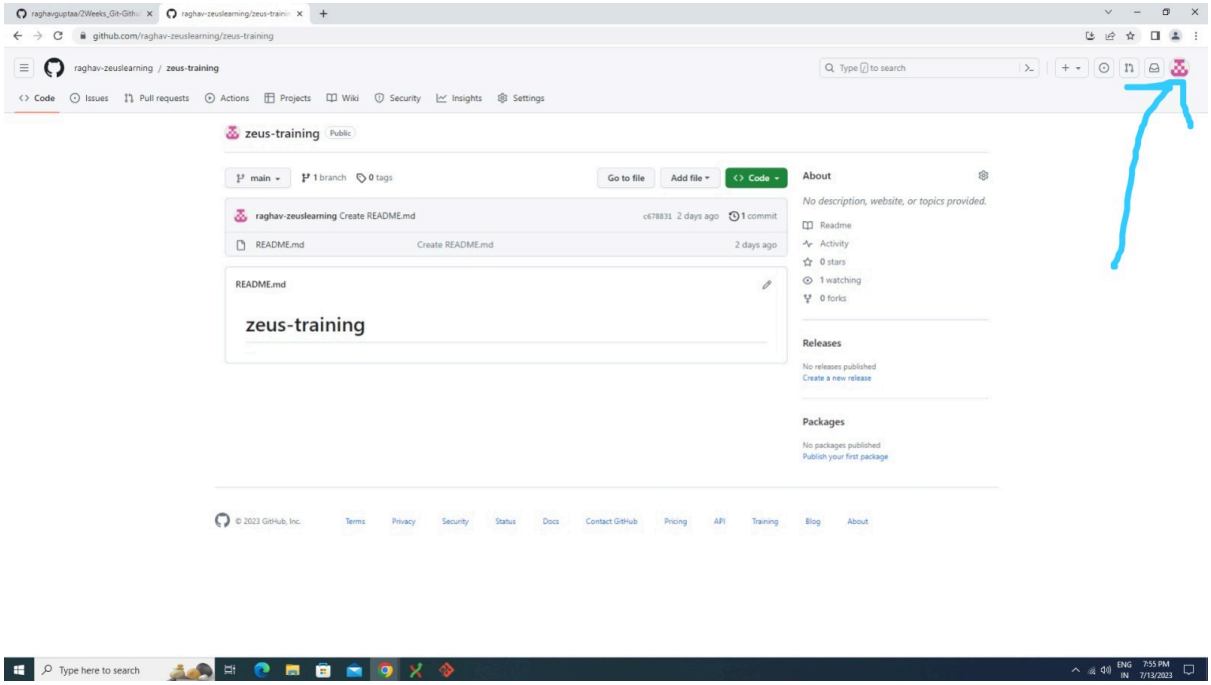
STEP 11:
No changes are required. Just Click on the Push Button.



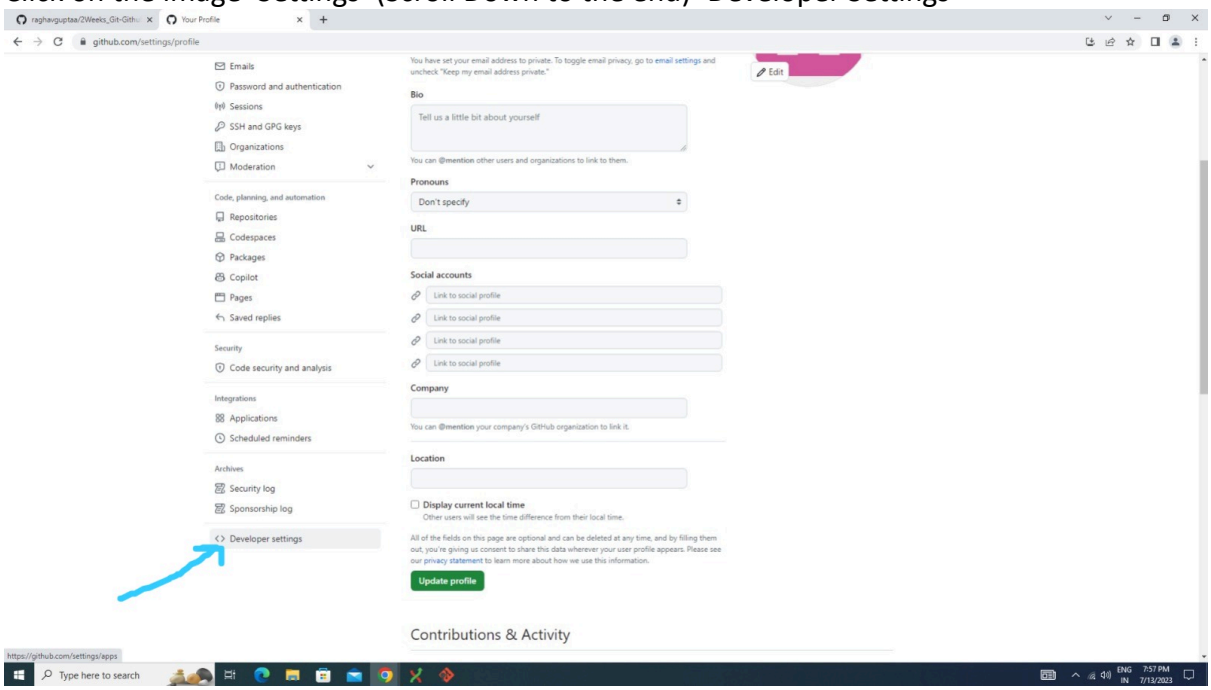
Now Github will ask us to login to the account in 2 ways.
1. Browser
2. Token

If we use Browser, then we must again and again log-in to the GitHub for pushing and pulling. And it's not the safest option to give our id and password to the machine. So, iit's better to use Token.

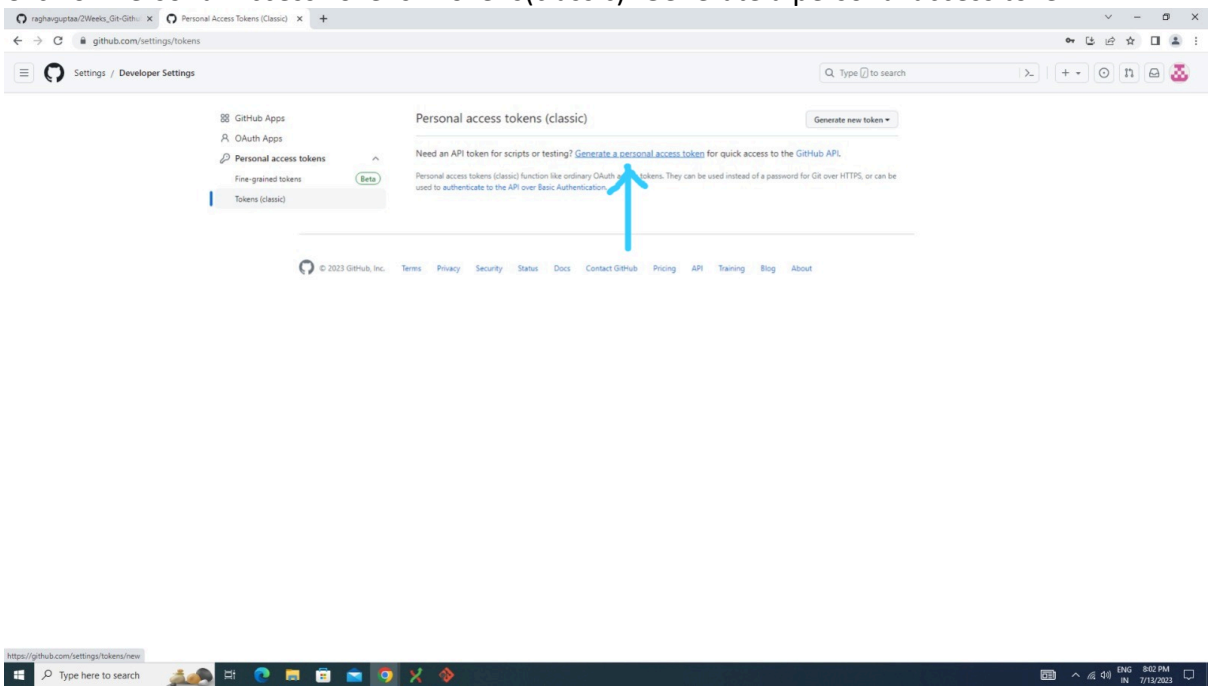
For getting token we need to go to the GitHub home page:



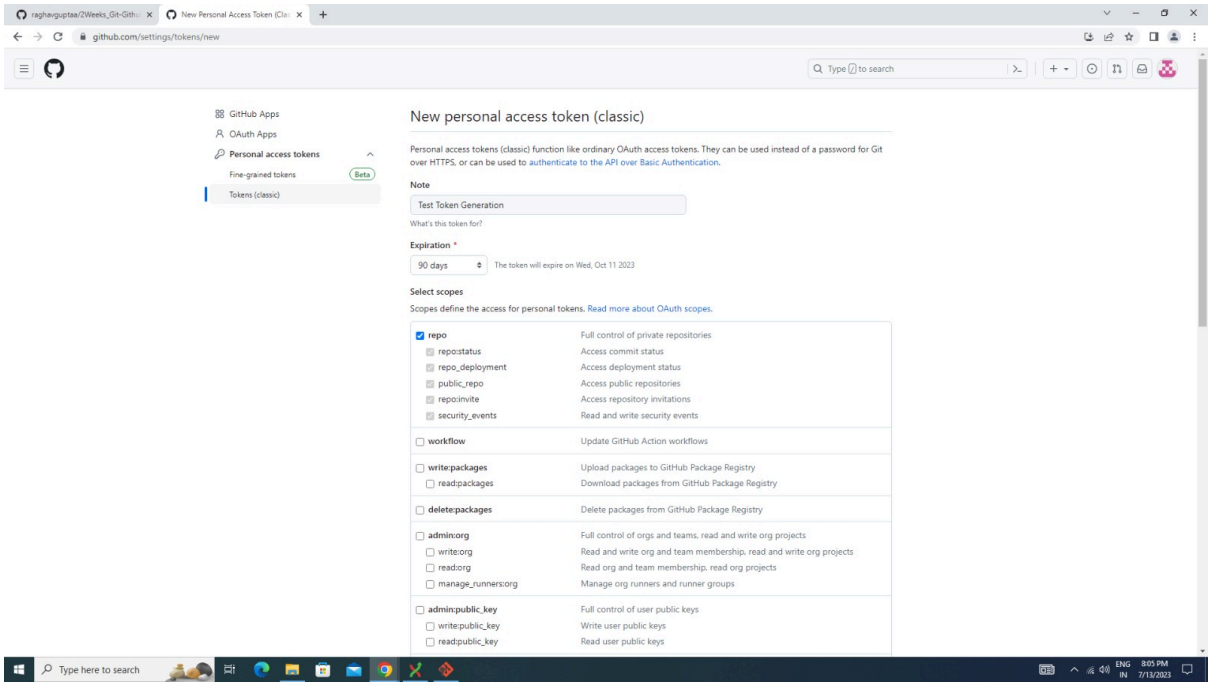
Click on the image>Settings>(Scroll Down to the end)>Developer Settings>



Click on Personal Access Tokens> Tokens(classic)>Generate a personal access token.



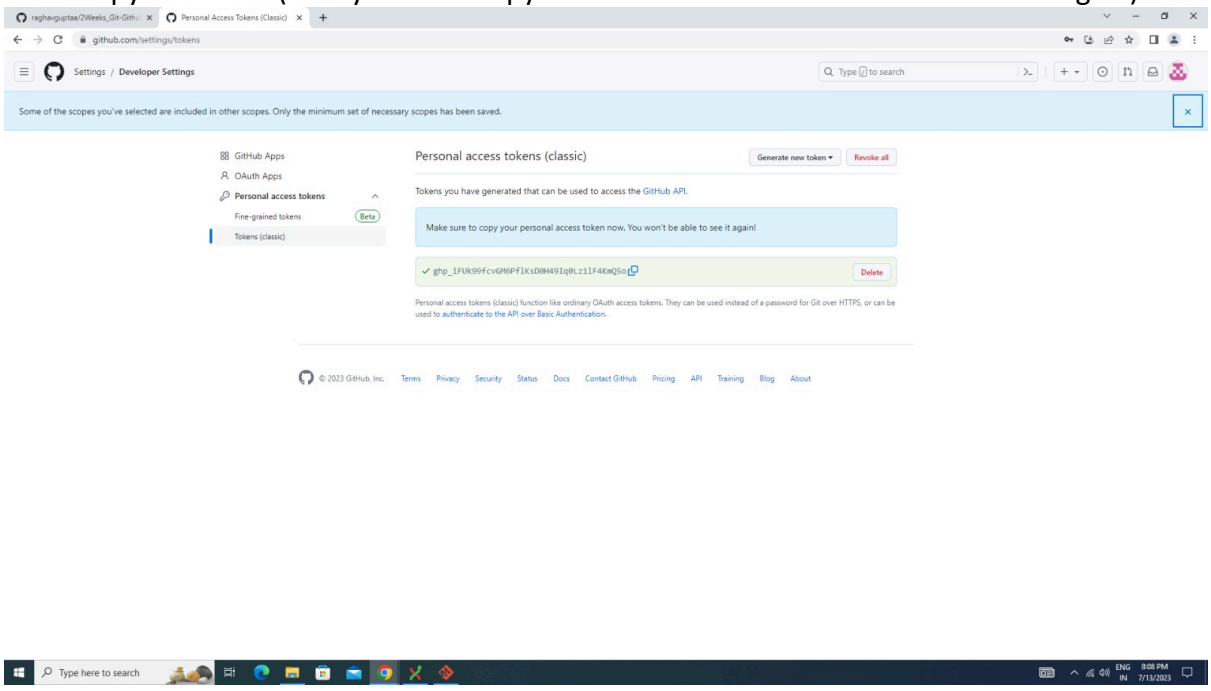
It might ask you the GitHub password. Just Provide it.



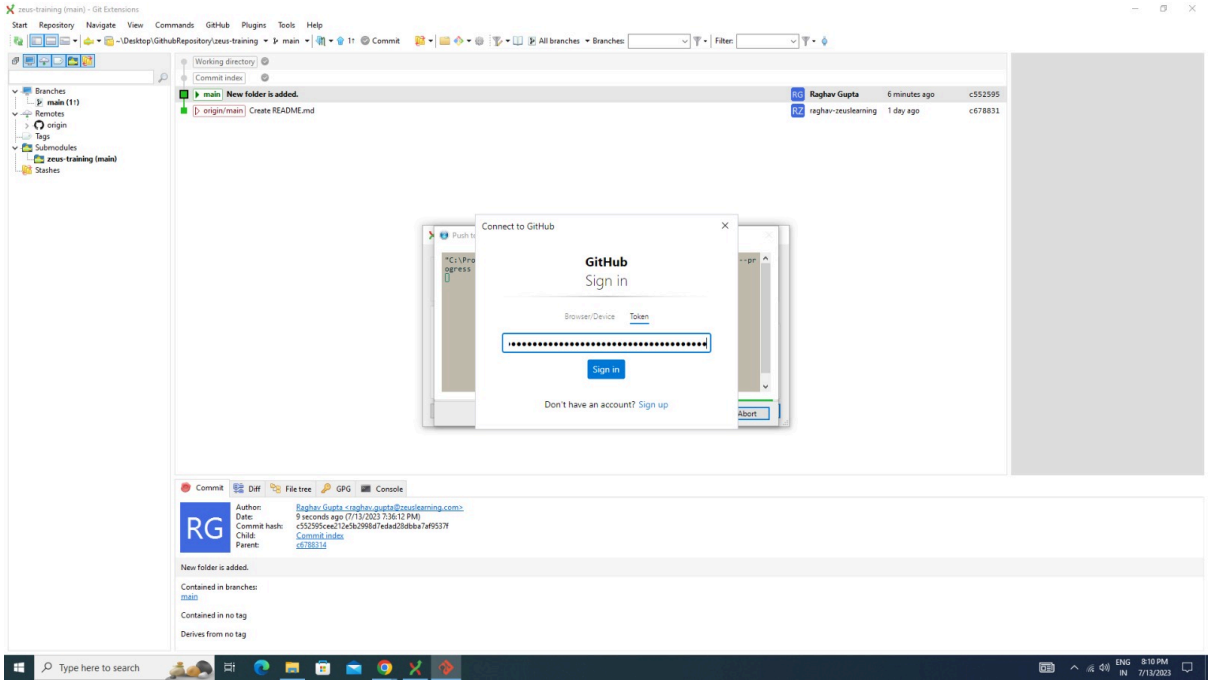
Provide a note: Anything can be on the note.
Select 90 Days as expiration.
Then Click on repo.

Scroll Down to Generate Tokens.

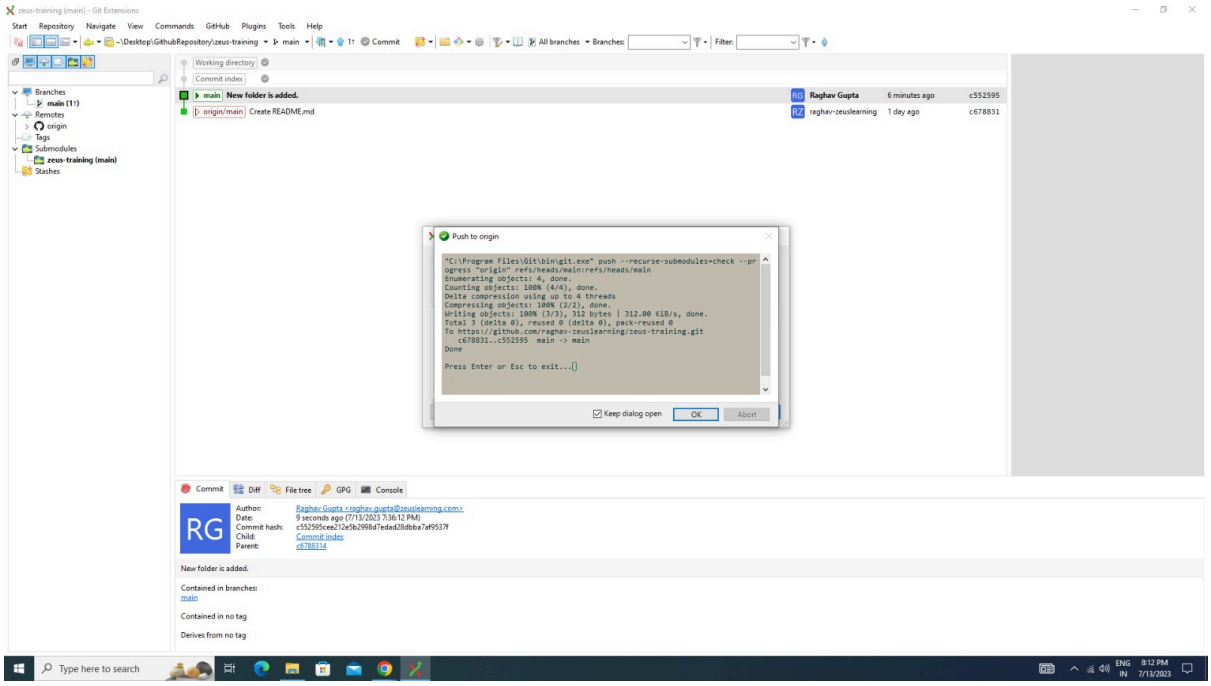
Then Copy the code. (Always have a copy of this code. Once lost can't be retrieve again)



Paste it in the token section and click on sign-in.



Now we can see the code is Push-ed to the GitHub.



To confirm – Go to GitHub account and check the repo and you will see the code is pushed.

3. Branches.

Theory!!

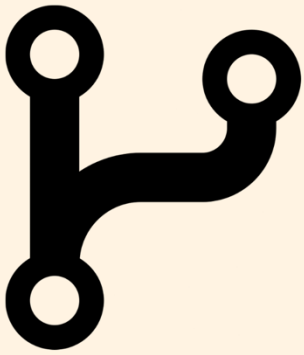
Branches

Branches are an essential part of Git!

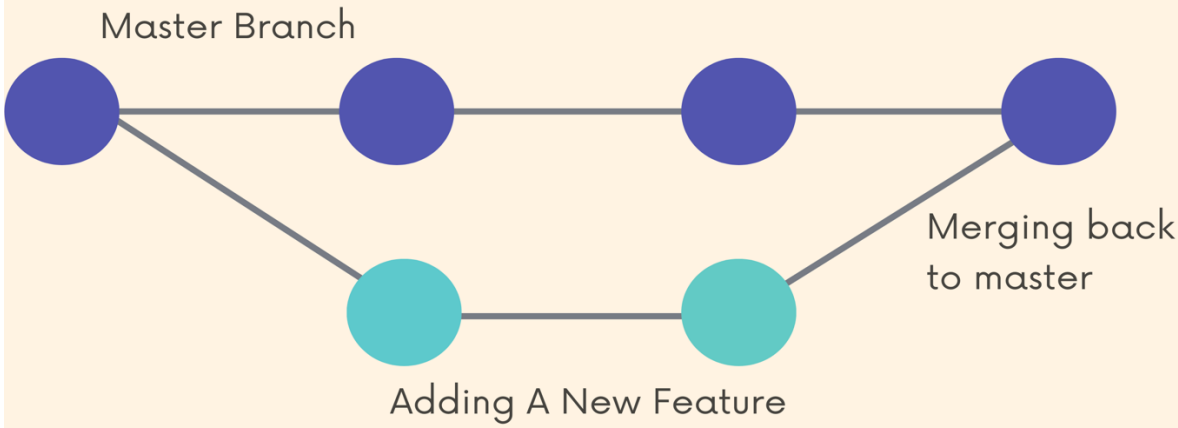
Think of branches as alternative timelines for a project.

They enable us to create separate contexts where we can try new things, or even work on multiple ideas in parallel.

If we make changes on one branch, they do not impact the other branches (unless we merge the changes)



A Common Workflow



.

In git we are always working on a branch. The default branch name is master.
(It doesn't do anything special or have any special powers. It's just like any other branch)

Master

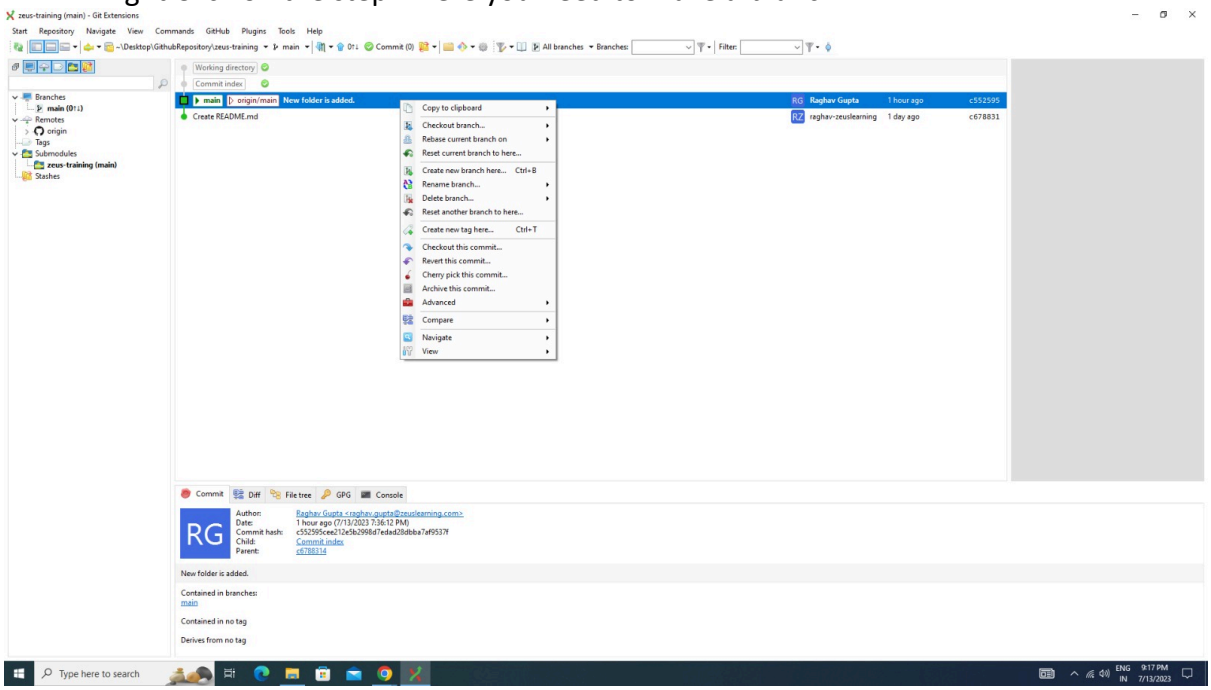
Many people designate the master branch as their "source of truth" or the "official branch" for their codebase, but that is left to you to decide.

From Git's perspective, the master branch is just like any other branch. It does not have to hold the "master copy" of your project.

Practical:

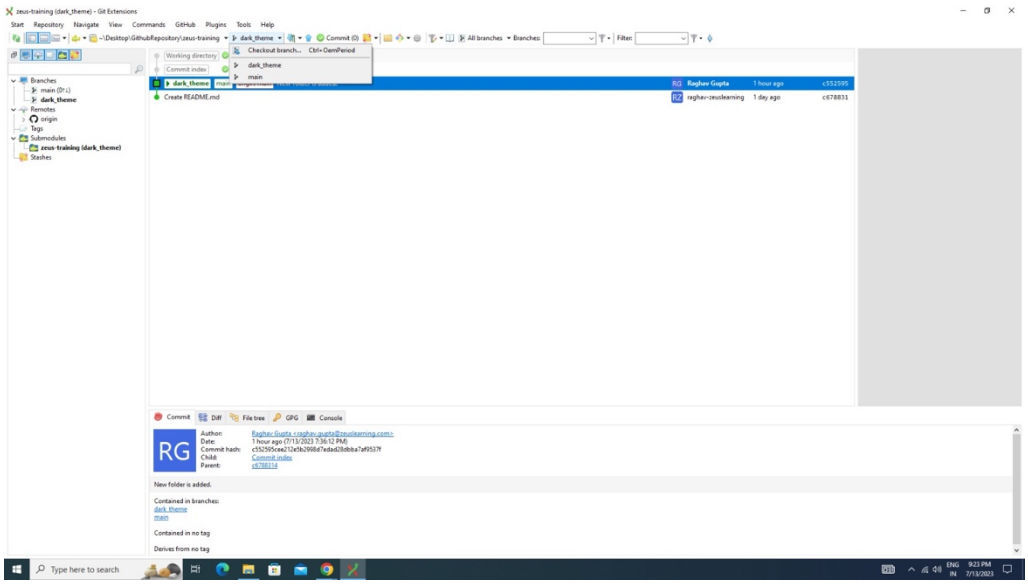
How to create a branch?

- Right Click on the step where you need to make a branch.



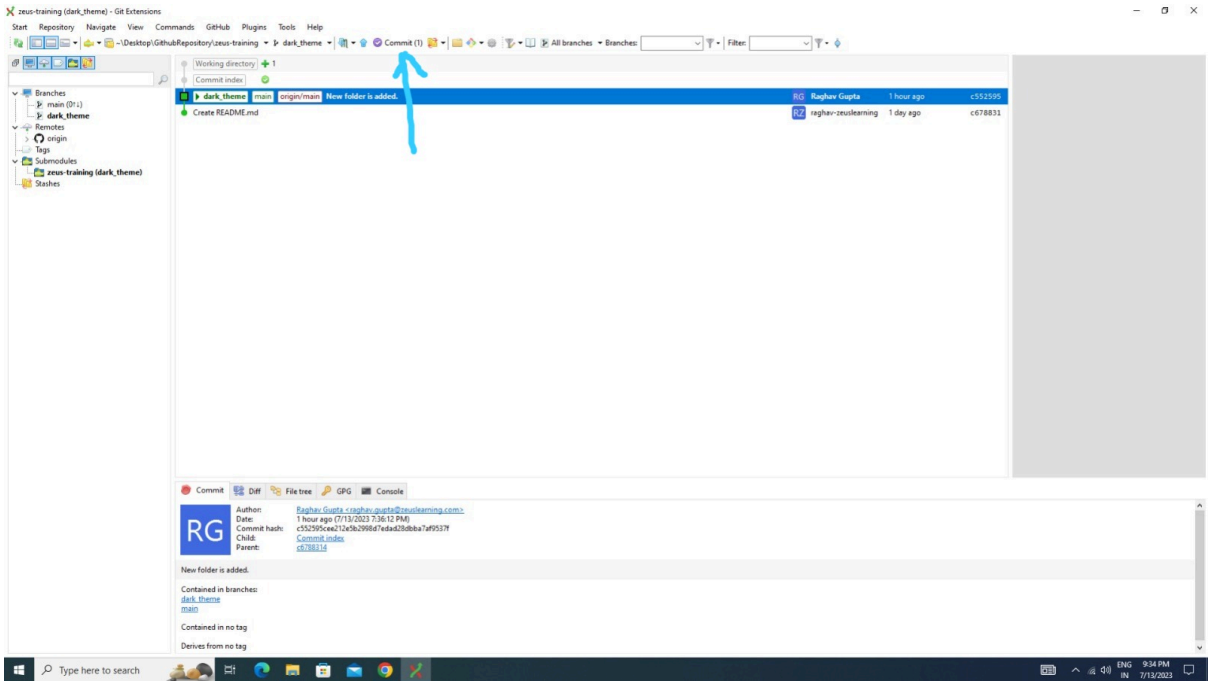
- Select “Create a New Branch here...”
- Rename the branch name according to the update.
    - Example: Dark Theme, New Feature etc. etc. (Avoid Using Spaces)
- Click on create branch. And done!

Now you would have 2 branches. Check them from here.

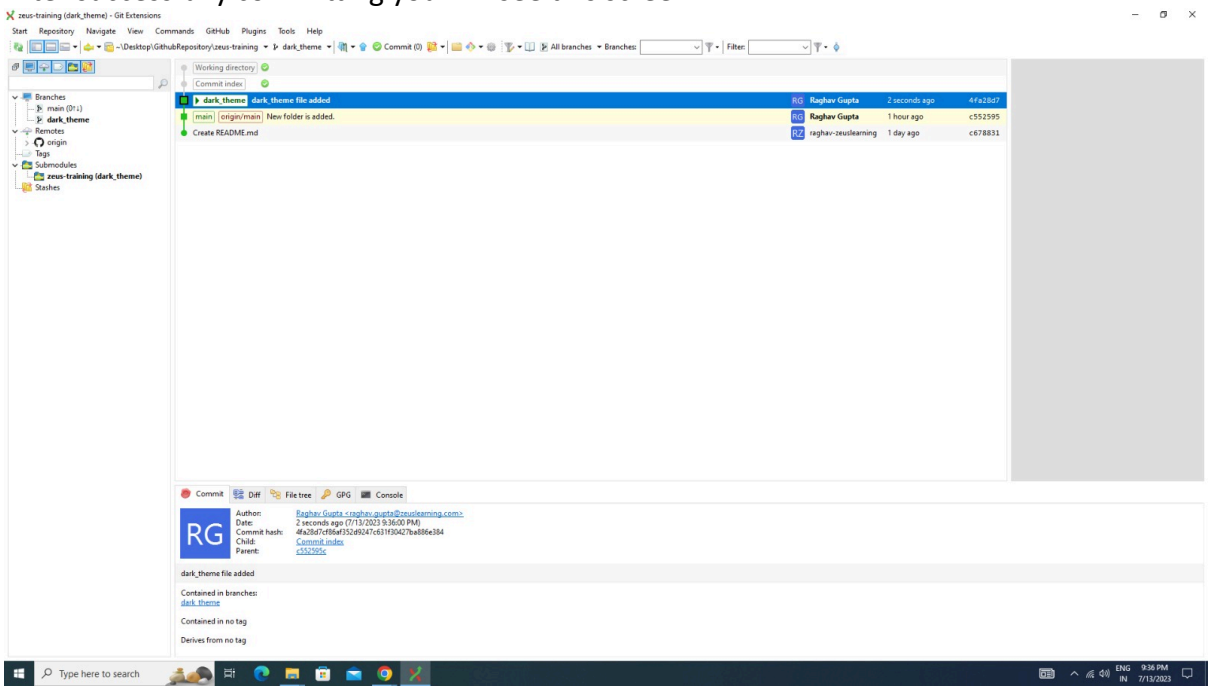


Whichever branch you need to work select and work accordingly.

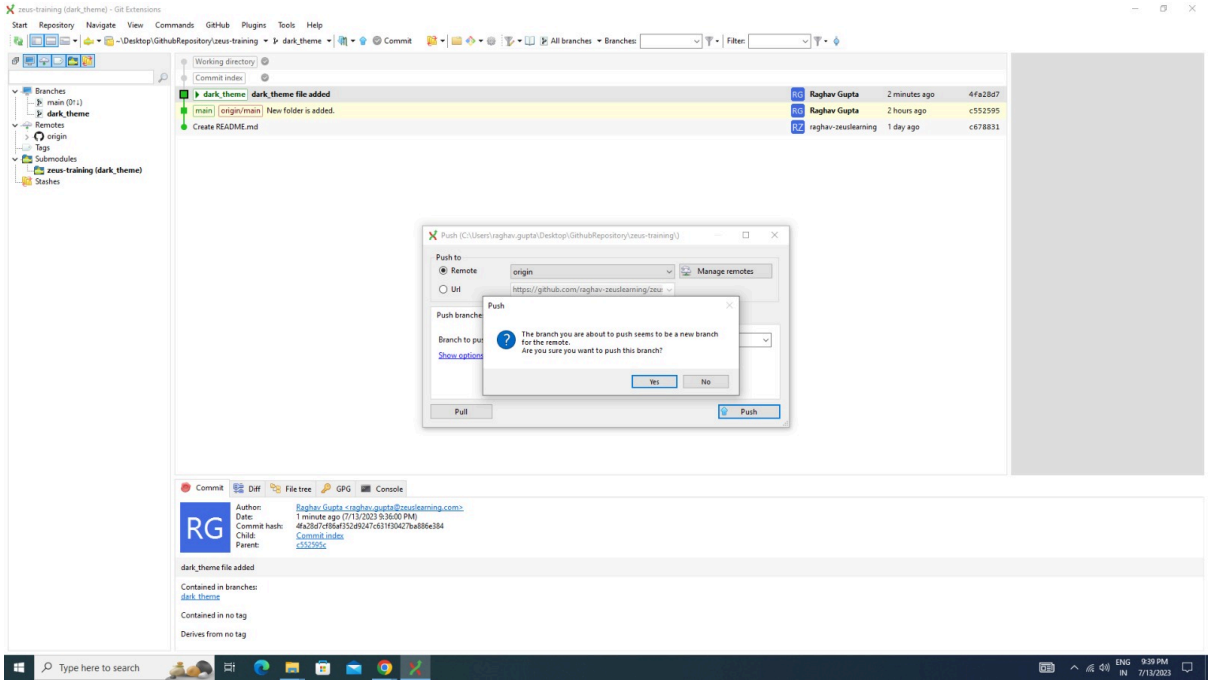
Now for better understanding,
1. Select the dark_theme branch.
2. On the desktop>githubRepository>-repositoryName->add new text file.
3. Name text file dark_theme and write anything in the file and save it.
4. After Saving you will find Commit(1)



5. Commit the file as per instructions given in previous section.
After successfully committing you will see this screen.

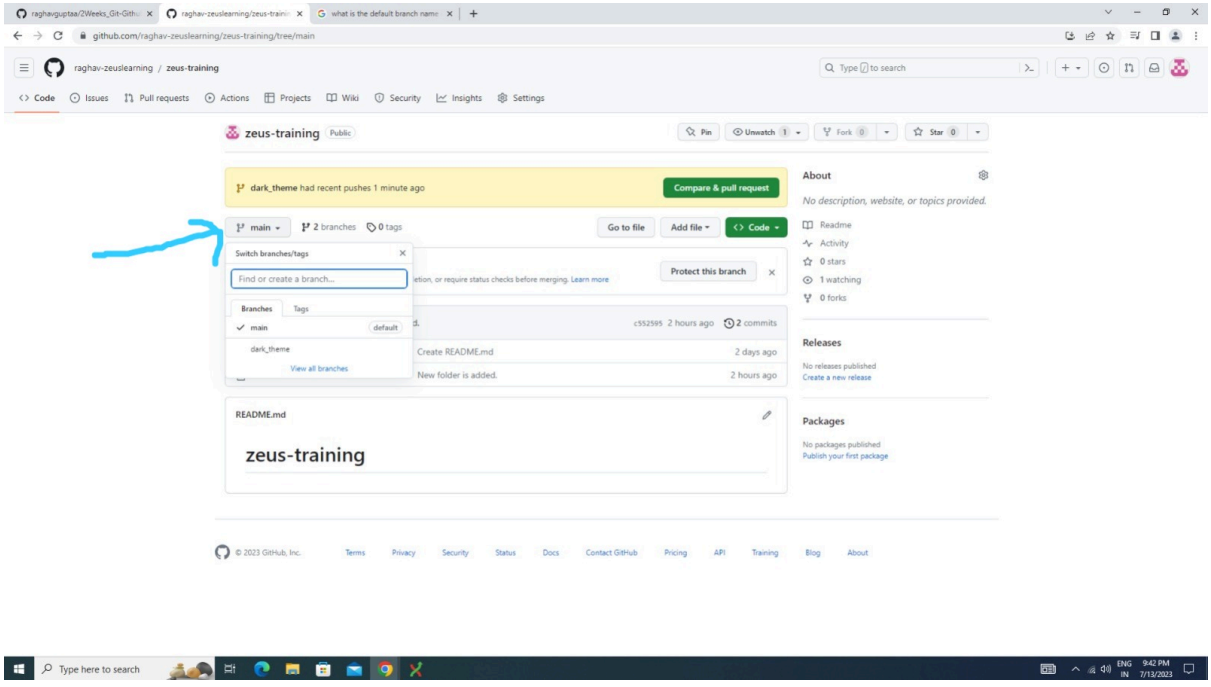


Now, push the branch to the GitHub repository.



Say yes for tracking.

Now, on there would be 2 branches on the GitHub Repository.
1. Main
2. dark_theme



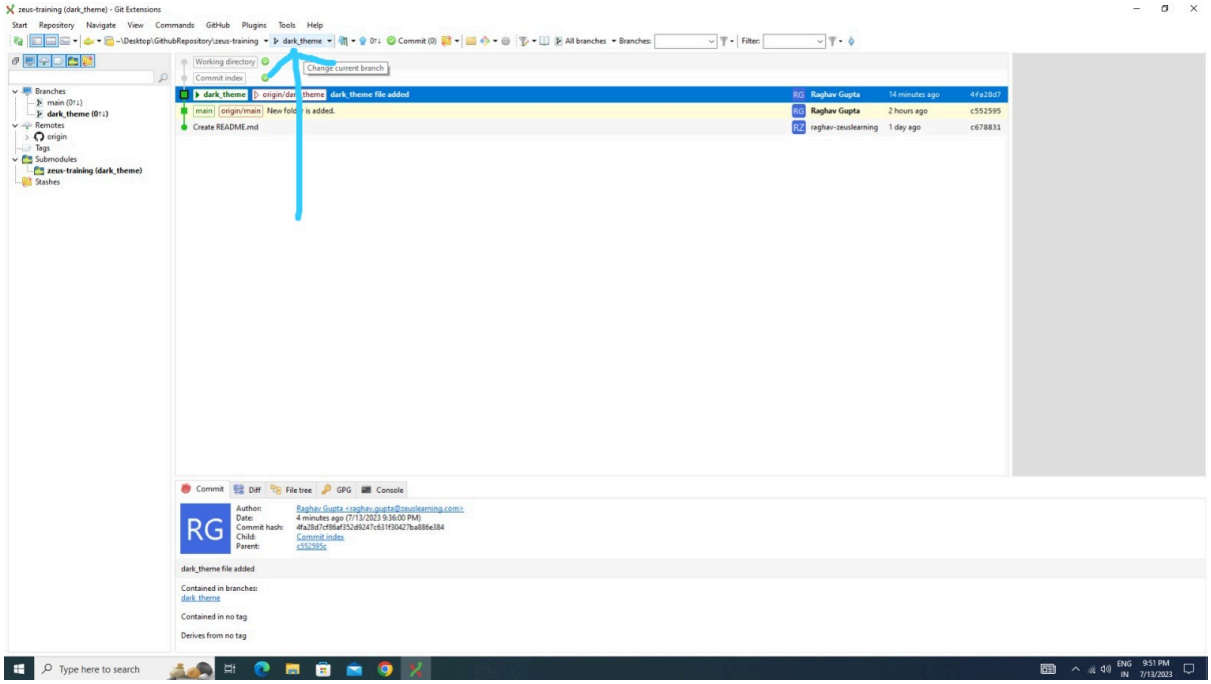
And done. Branching is Completed.

4. Git Merge

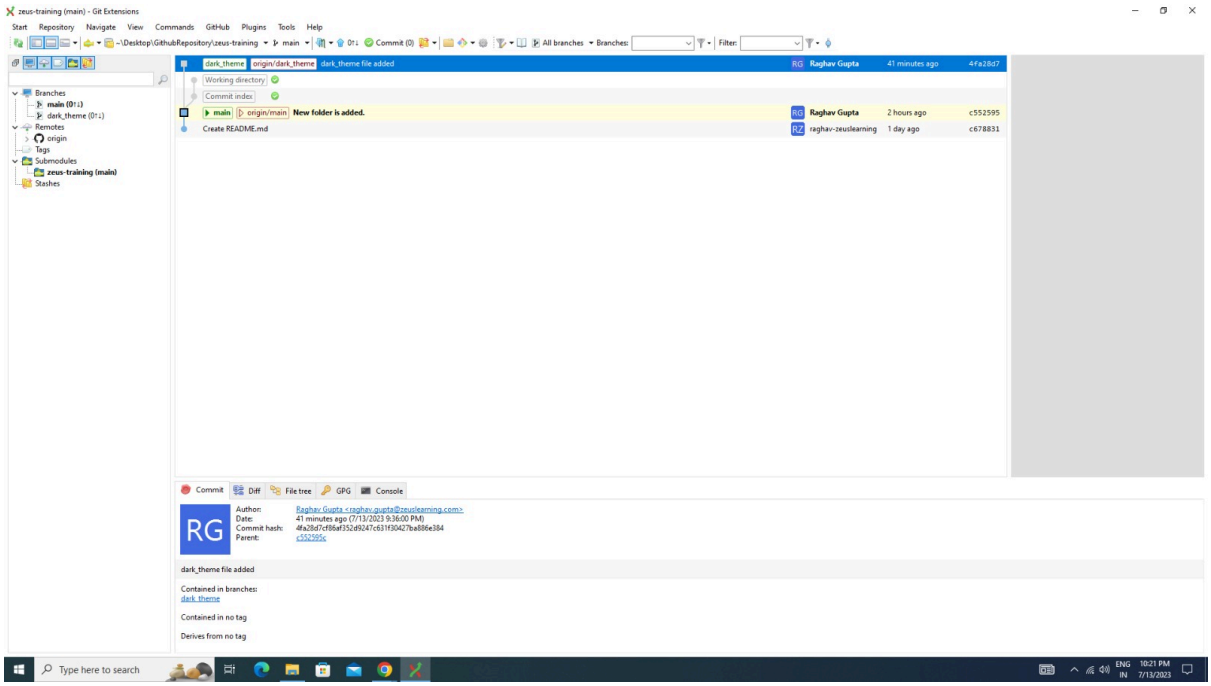
Git merge will merge the current working branch to the main/master branch or any other branch.

Now, we have created a dark_theme file and need to merge it to the main branch.

1. Firstly we need to check on which branch we are. The Current branch is dark_theme as we can see below.



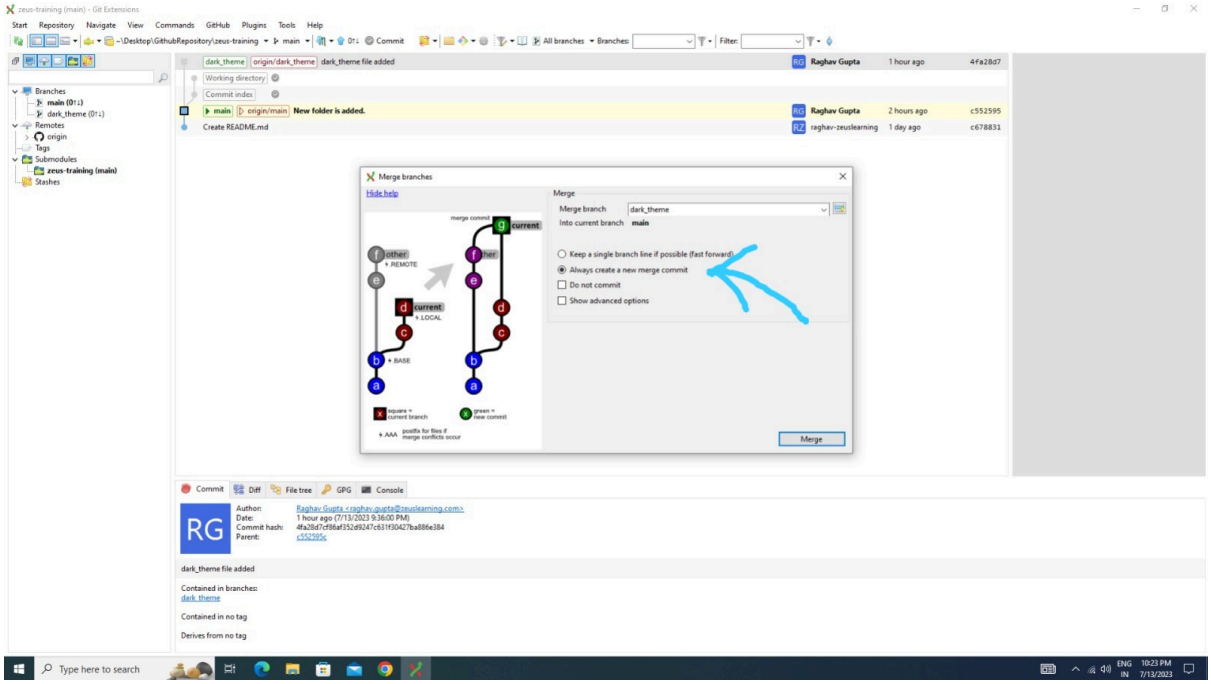
2. Now, we will select main branch. As we need to merge the dark_theme to the main branch.



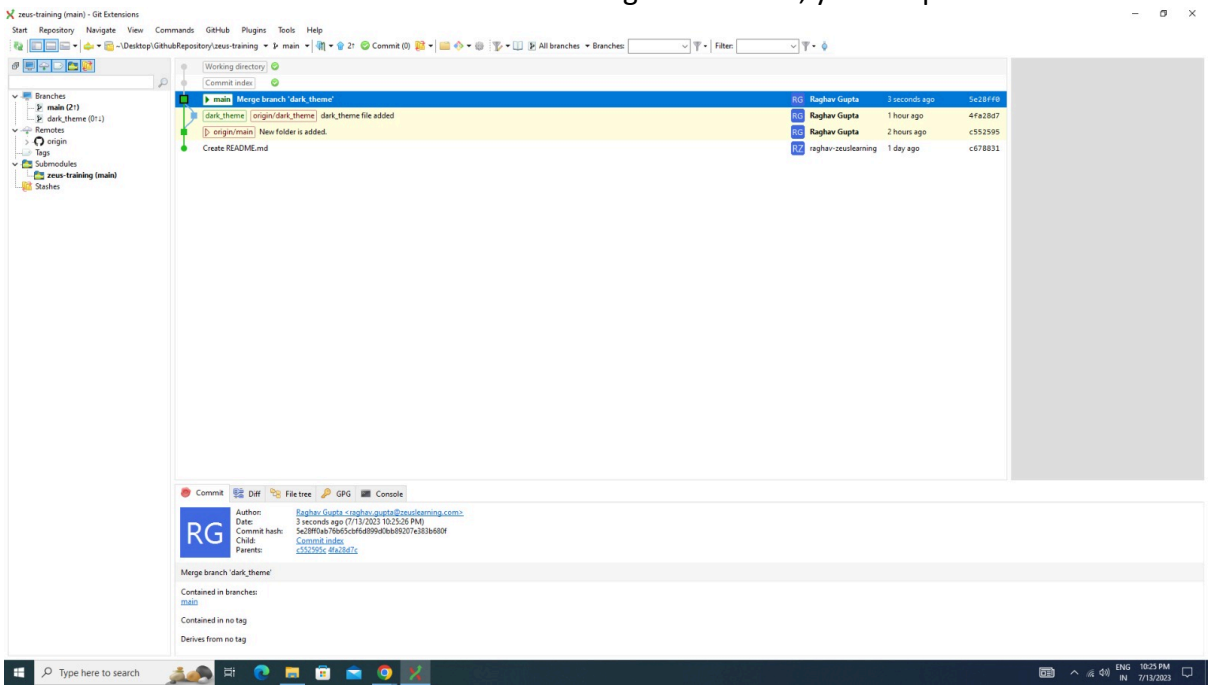
NOTE: If we need to merge to the main branch, we need to be on the main branch. And then select the other branch and merge it to the main branch.

3. Right click on the branch which we need to merge and then click on merge into current branch.

4. After clicking, always select the “Always create a new merge commit.”



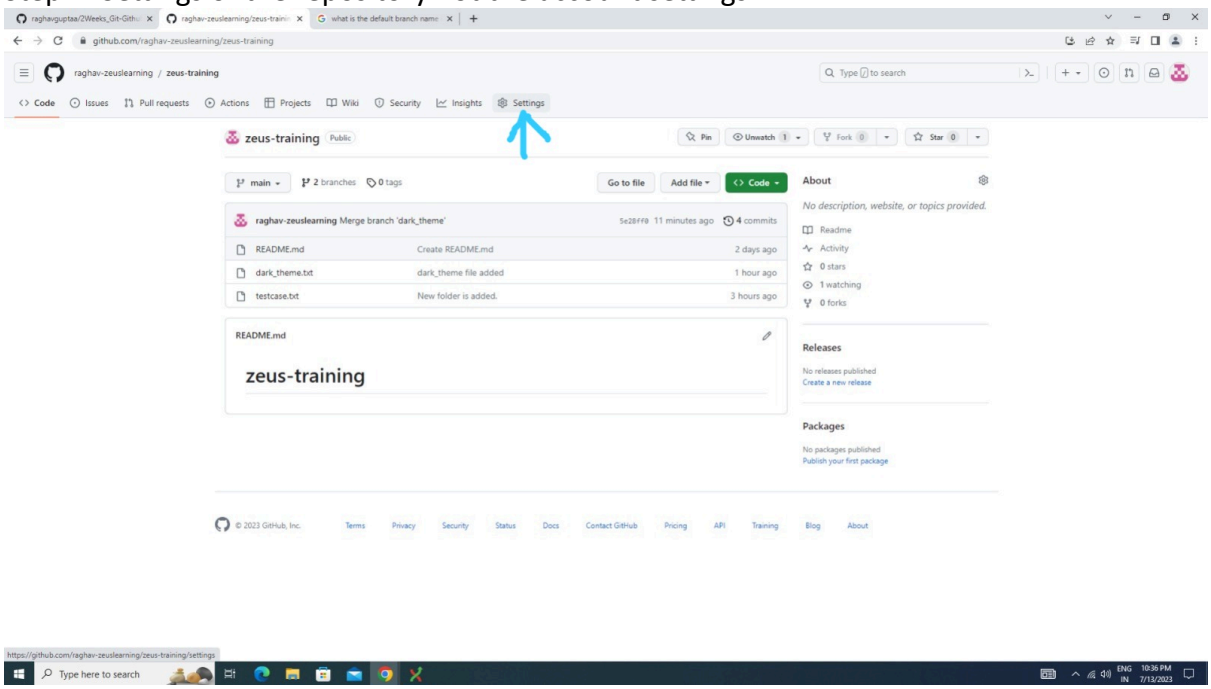
5. Now we can see the branches have been merged. And now, you can push it to GitHub.



5. How to invite other git users

Step 1: Go to the repository on which you want to add a collaborator.

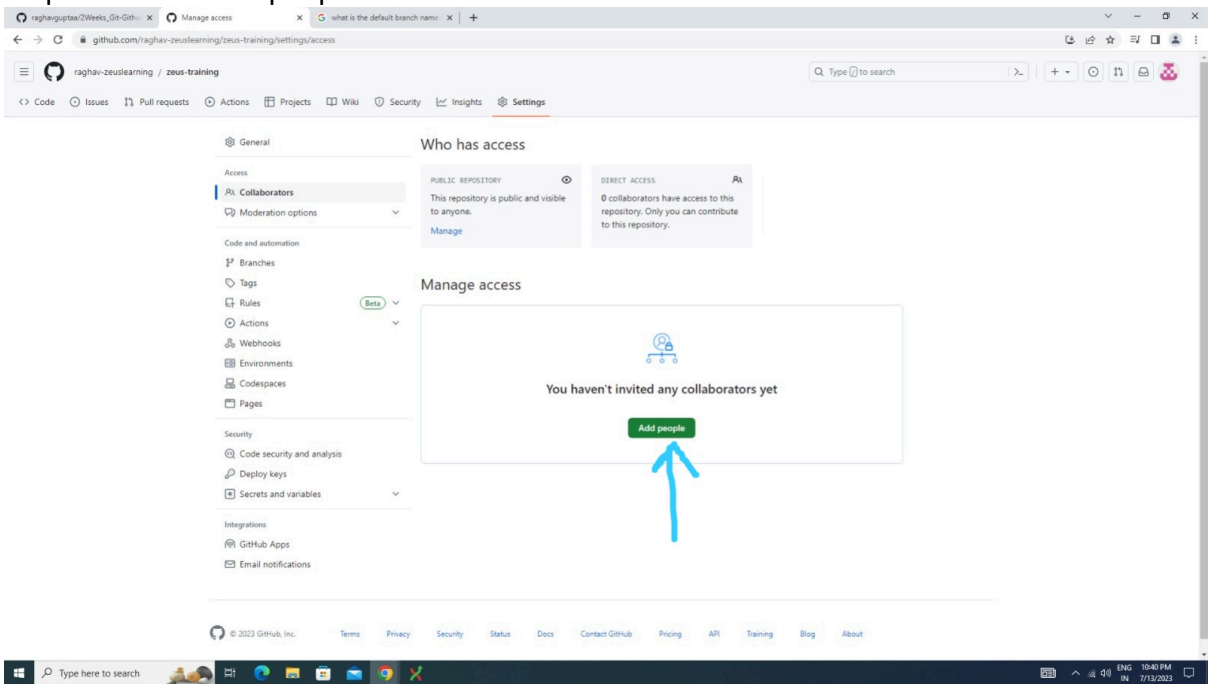
Step 2: Settings of the repository not the account settings.



Step 3: Click on collaborators.

Step 4: Password will be asked. Provide the password.

Step 5: Click on Add people.



Then add them with email/username.

6. Git Fetch vs Pull

Git Fetch	Git Pull
Gives the information of a new change from a remote repository without merging into the current branch	Brings the copy of all the changes from a remote repository and merges them into the current branch
Repository data is updated in the .git directory	The local repository is updated directly
Review of commits and changes can be done	Updates the changes to the local repository immediately.
No possibility of merge conflicts.	Merge conflicts are possible if the remote and the local repositories have done changes at the same place.
Command for Git fetch is git fetch<remote>	Command for Git Pull is git pull<remote><branch>
Git fetch basically imports the commits to local branches so as to keep up-to-date that what everybody is working on.	Git Pull basically brings the local branch up-to-date with the remote copy that will also updates the other remote tracking branches.

Simply, we majority of the times use only git pull.

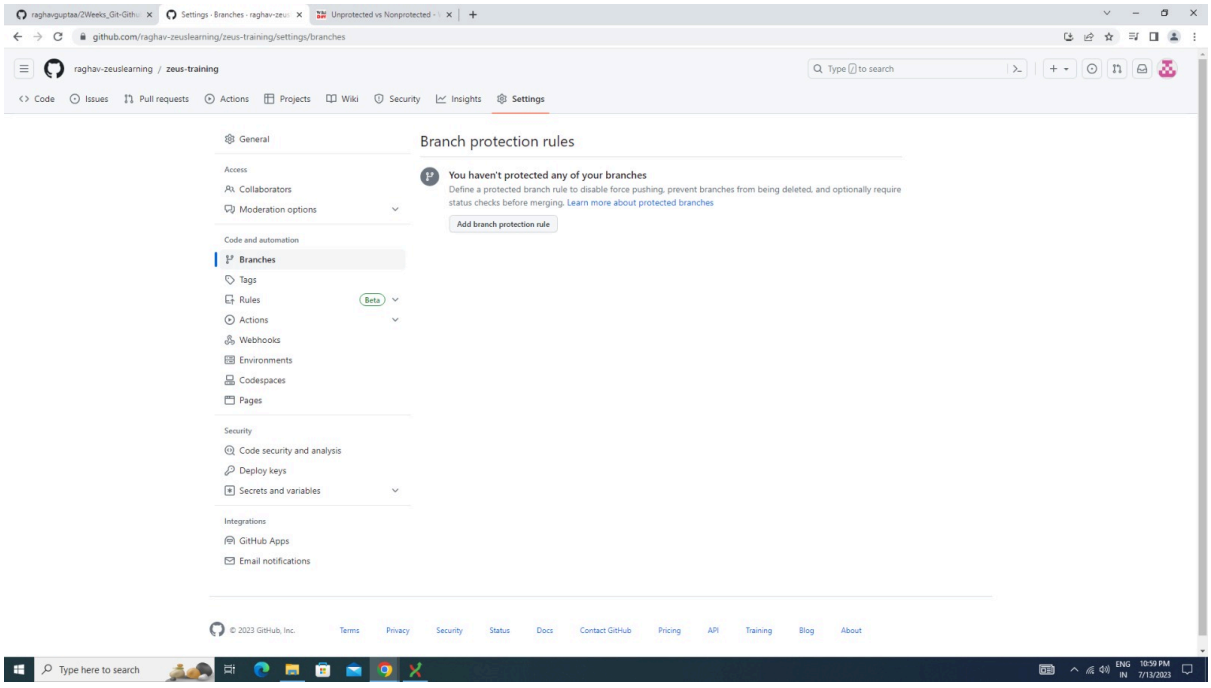
7. Pull Request.
There are two types of branches.

1. un-protected.
2. Protected

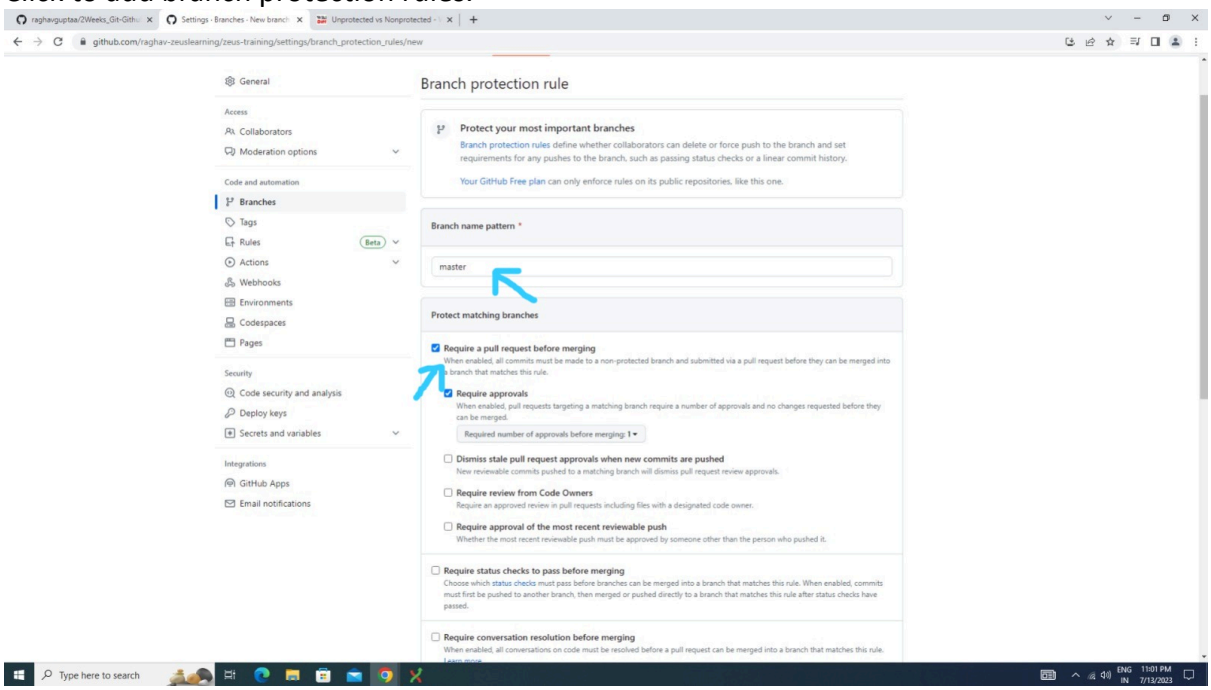
We were using un-protected branch until now. So, anybody can merge branches without any reviews from the owner. Which is not good. Right?

So, we need protection from unwanted merging. So, GitHub provide us protected branches.

Q. How to make a repository protected?
Go to the repository > Settings> Branches.



Click to add branch protection rules.



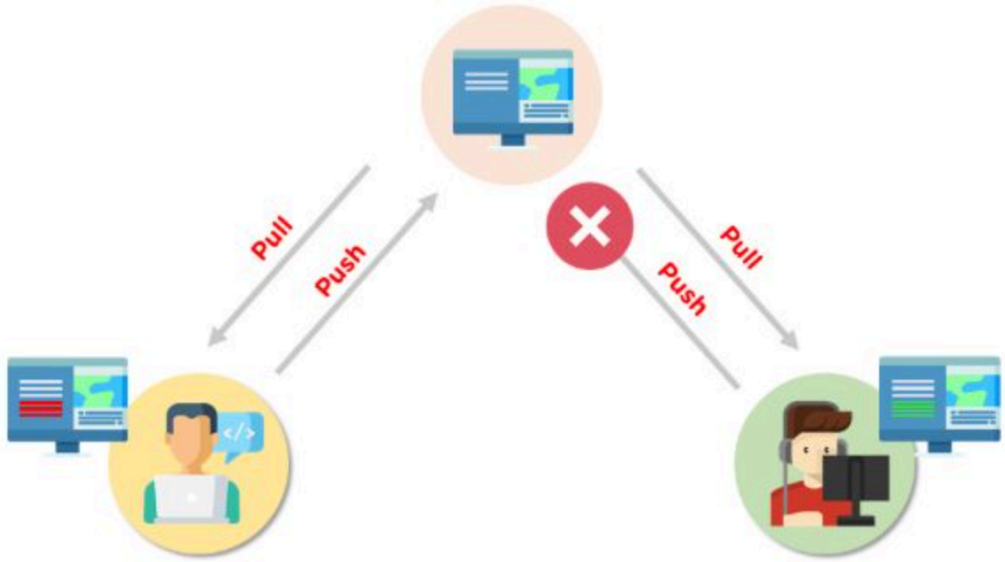
Now, before merging a branch we require a pull request. Scroll Down and click on Create.

Now, our GitHub repo is protected.

8. What is merge conflict.

A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

The following is an example of how a Git merge conflict works:



Let's assume there are two developers: Developer A and Developer B. Both of them pull the same code file from the remote repository and try to make various amendments in that file. After making the changes, Developer A pushes the file back to the remote repository from his local repository. Now, when Developer B tries to push that file after making the changes from his end, he is unable to do so, as the file has already been changed in the remote repository.