**Aim**: Version control using Git.

**Objectives**: To understand working on github and solving conflicts.

**Tools used**: Github, Virtual box, ubuntu, linux terminal, Kdiff3

**Concept:** To understand how git works.

**Problem statement**:

2.1:

● Create a local git repository and move the entire code base to it

● Create a new branch for a new feature you want to add to the application

● Merge back the created branch with the master branch

● Create a remote repository

● Push the local repository to company’s remote repository

2.2:

● Create two separate branches from master

● Make changes in the same function of the source code in both the branches

● Merge branch1 into the master

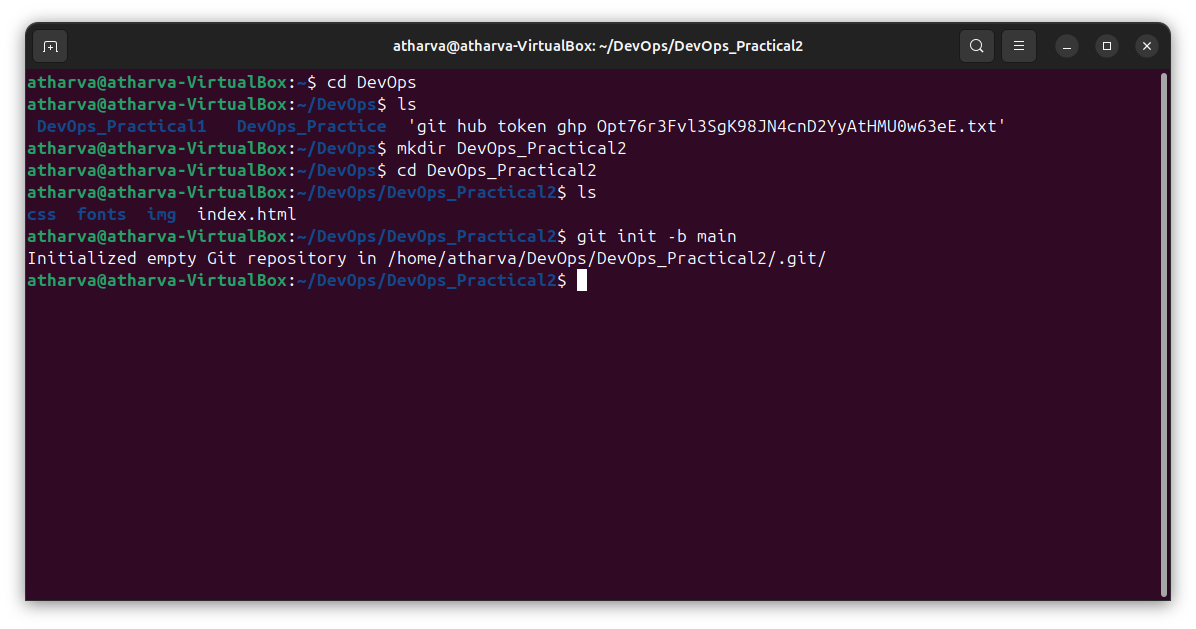
● Try and merge branch2 into the master (merge conflict should arise)

● Install a merge tool of your choice and resolve the merge conflict using git

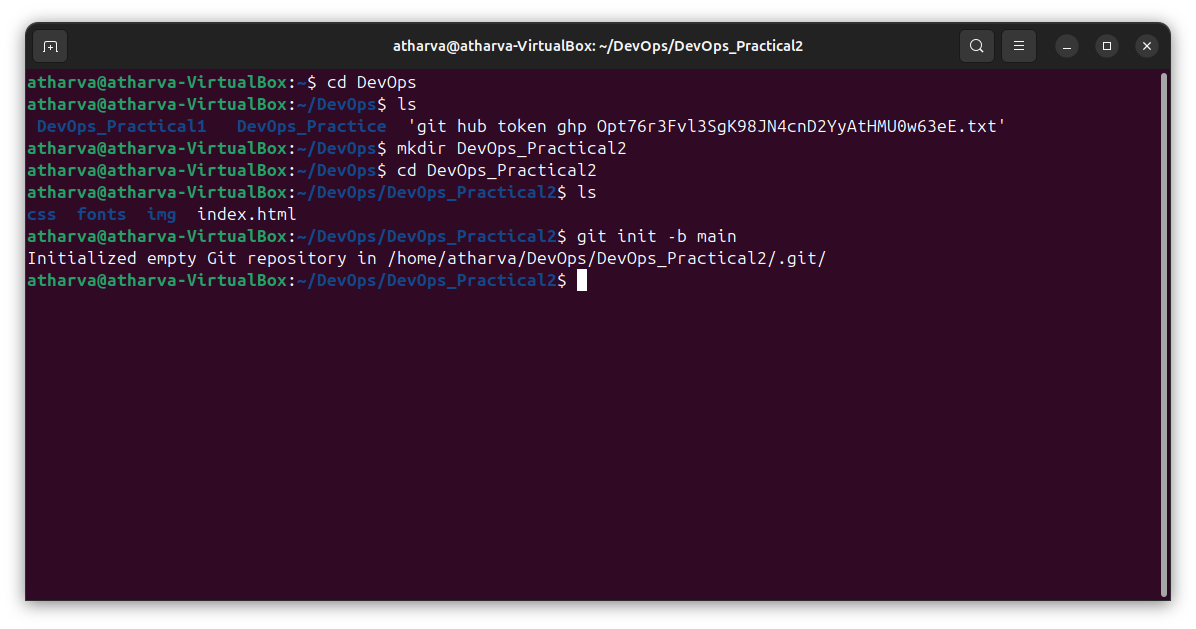
● mergetool command.

2.1:

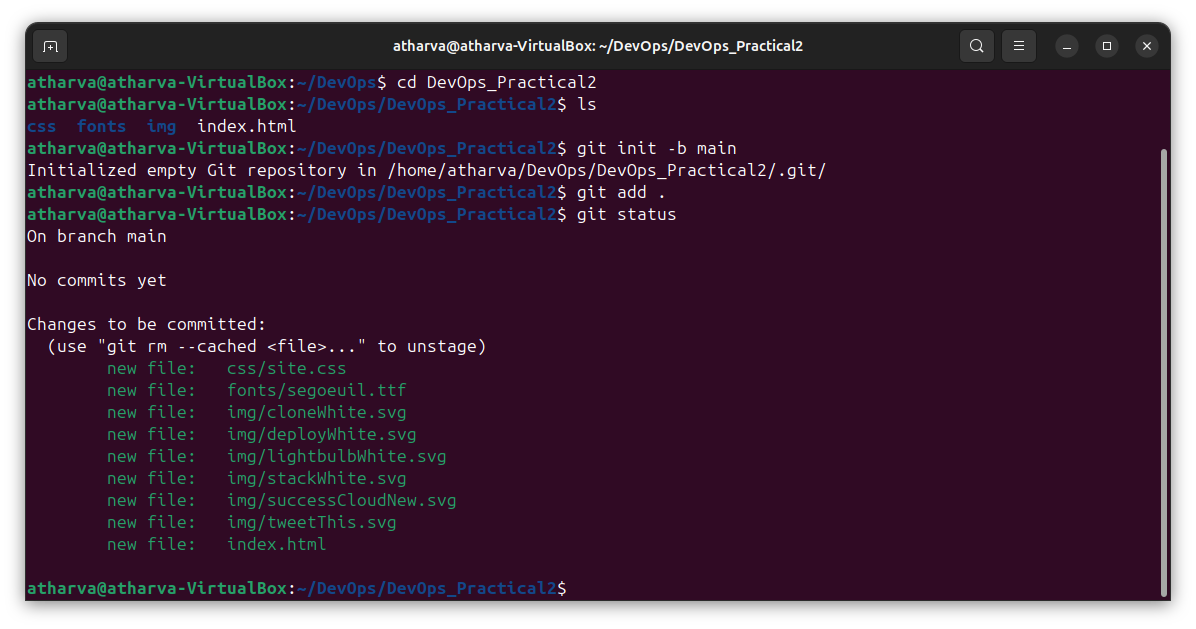
1. Create a Sample Project and open the terminal and navigate to the sample project folder



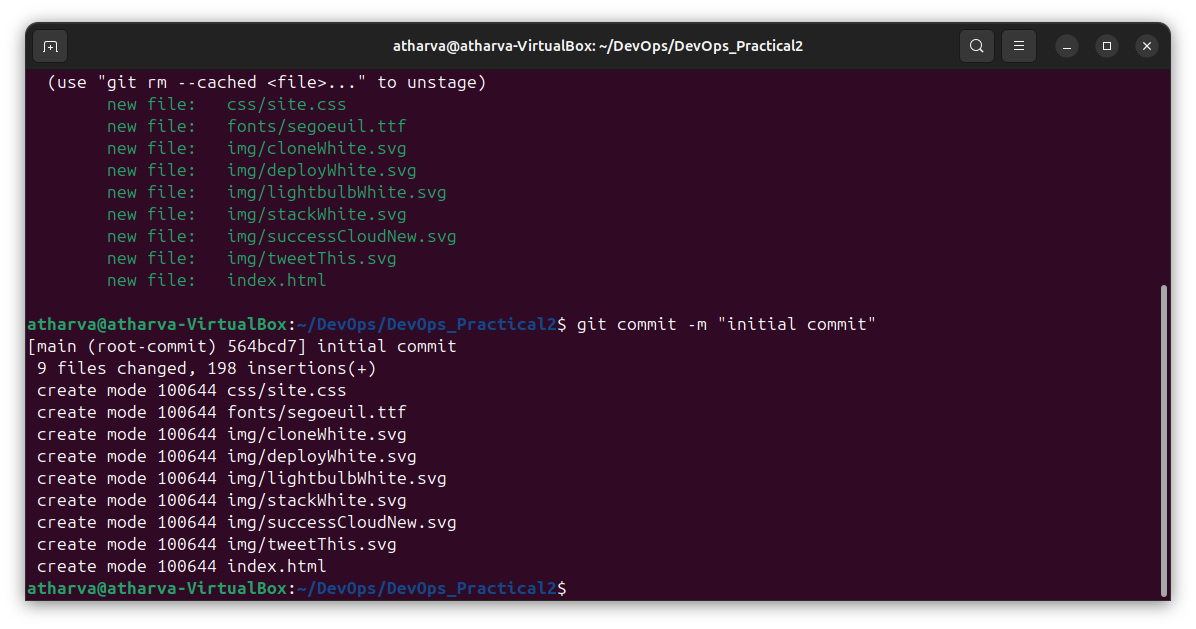
2. Initialize this folder as local git repository by git init command



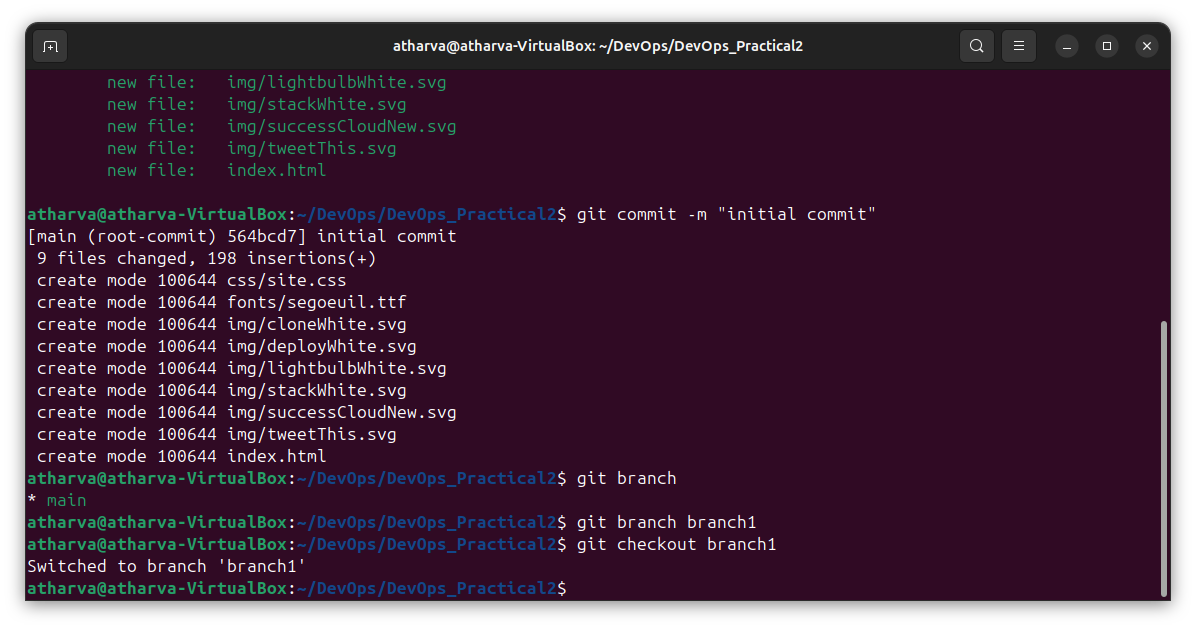
3. Add all the files to staging area and use status command



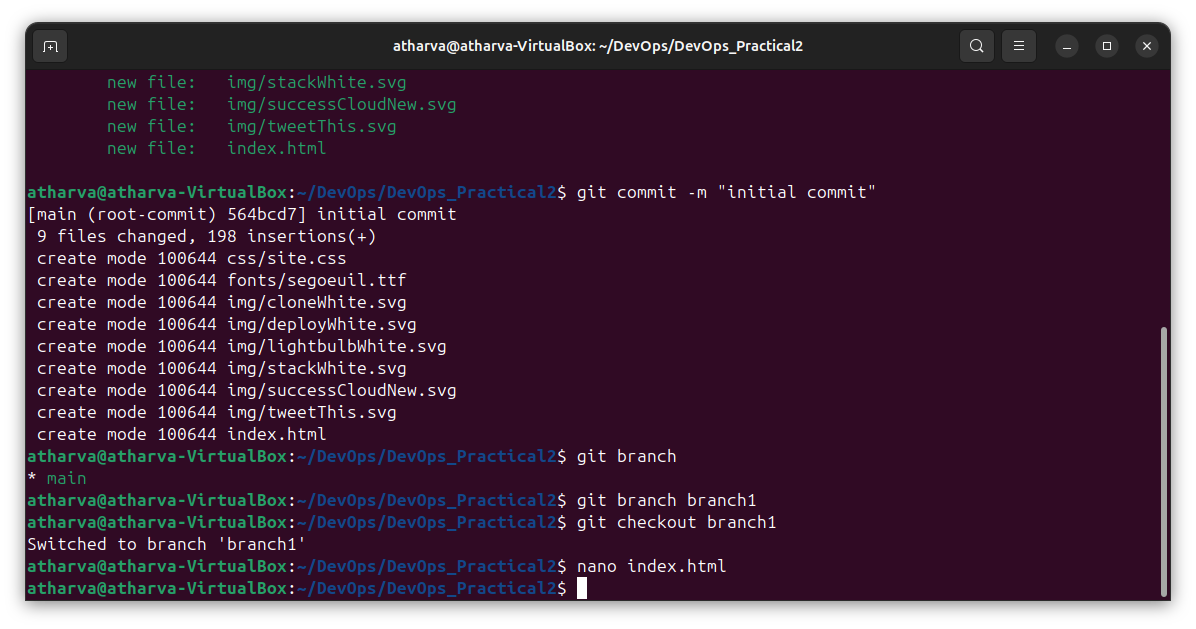
4. Make initial Commit

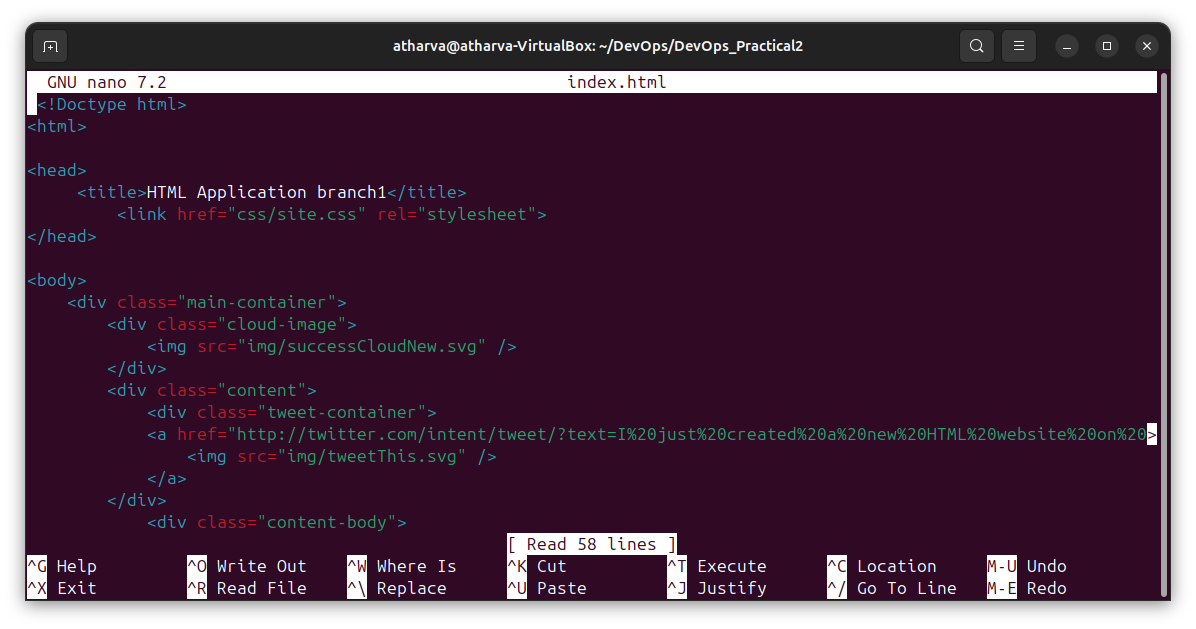


5.Now create a new branch named branch1 and switch to branch1 by using checkout command

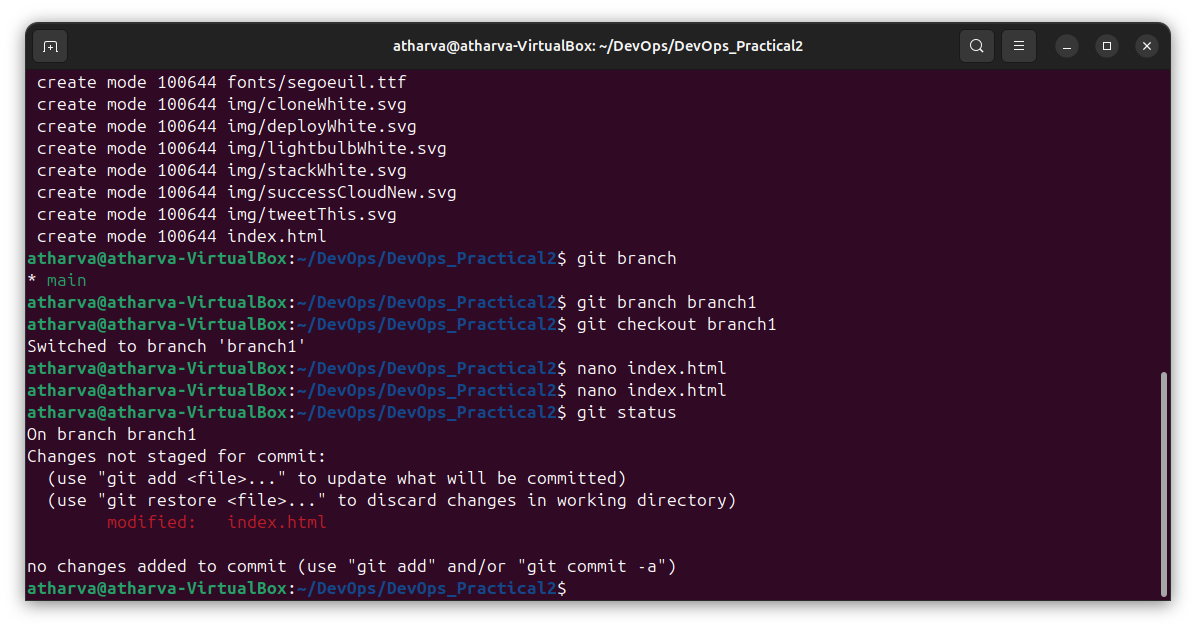


6. Change the title in index.html by using nano command making changes save the changes

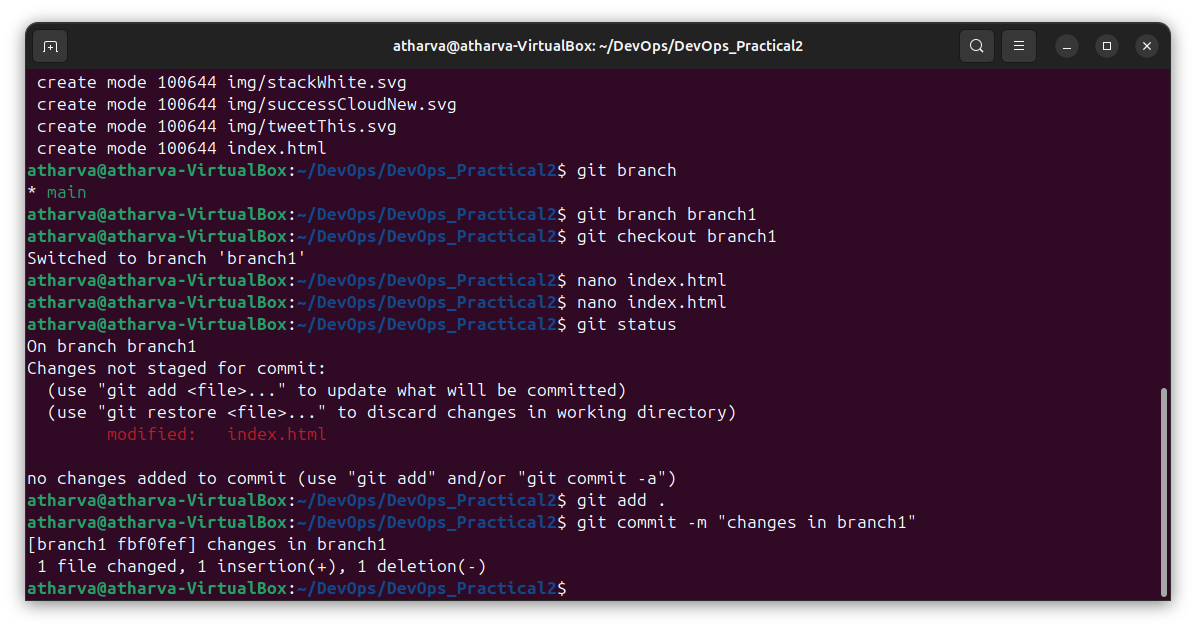




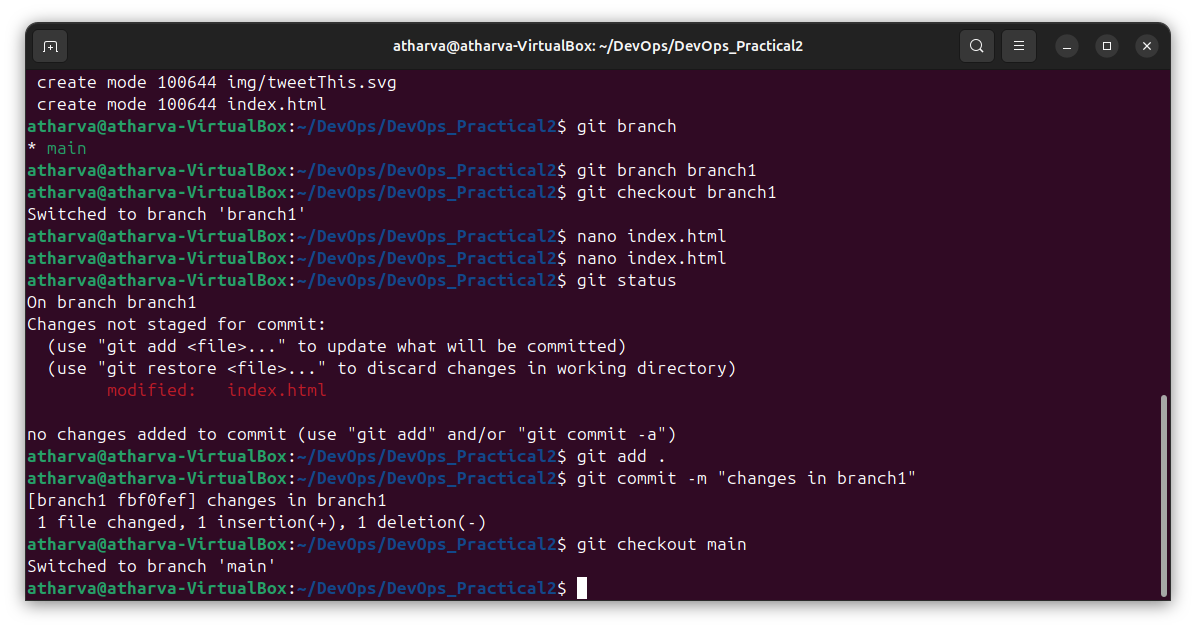
8. Now check the status by using git status



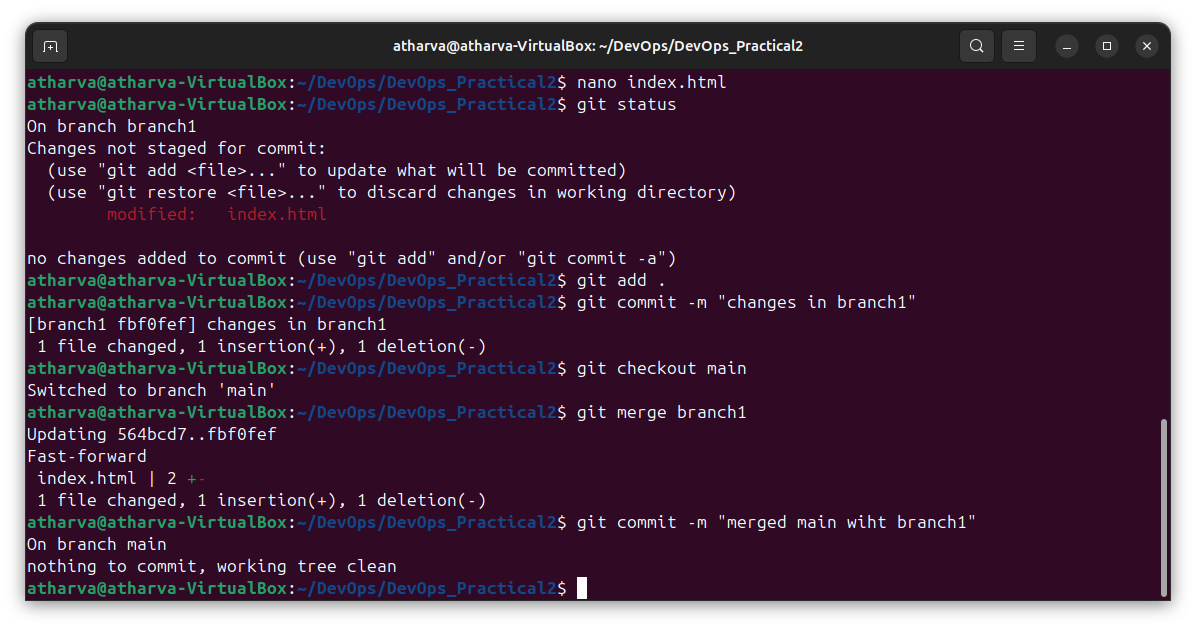
9. Now add all the files in the staging area and then commit



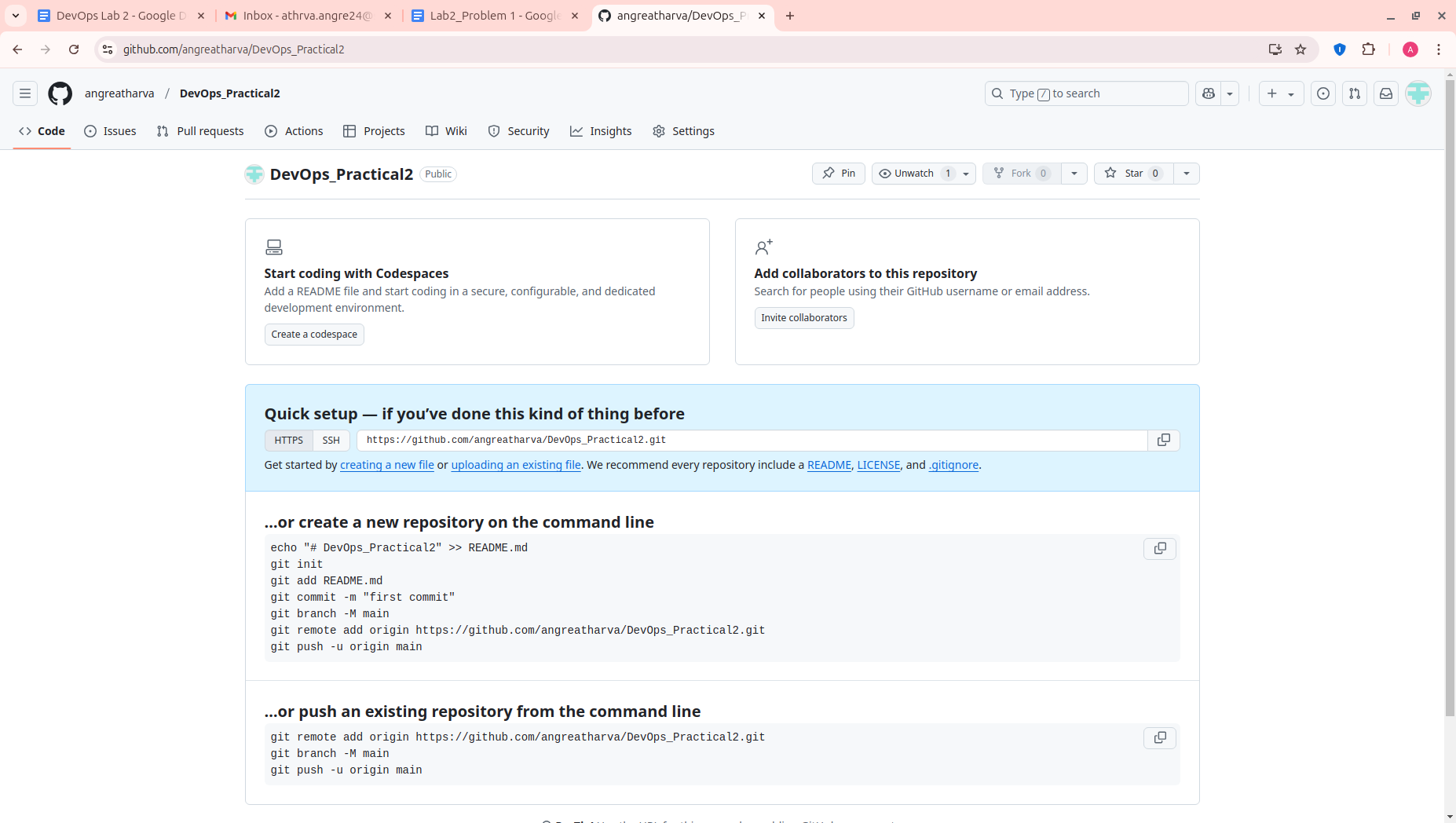
10. Now switch to main branch



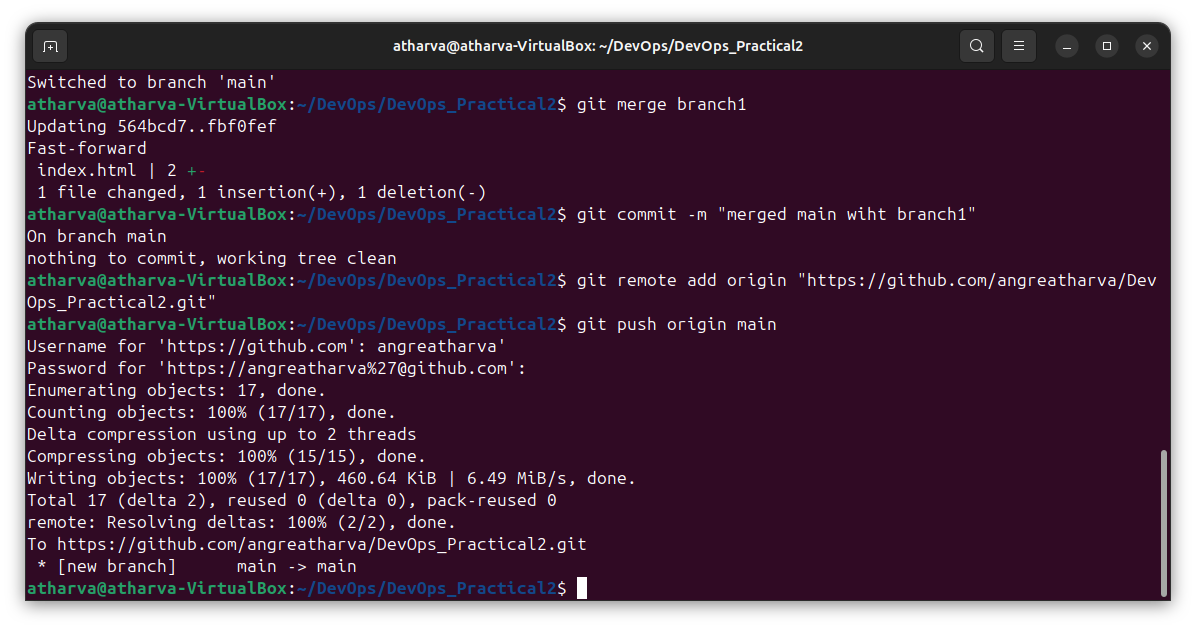
11. Now the merge the changes and make the commit



12. Now create a git repository

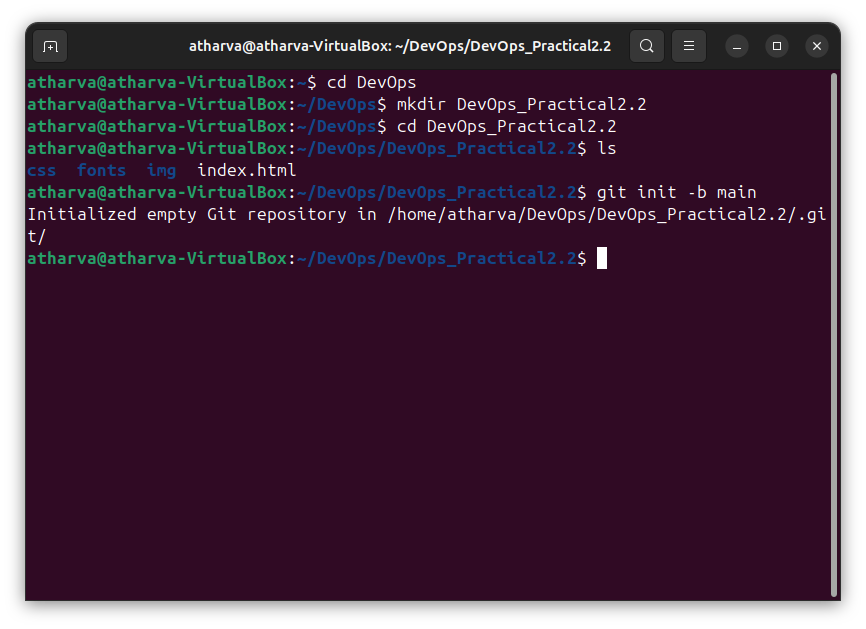


13.Add remote and Push it.

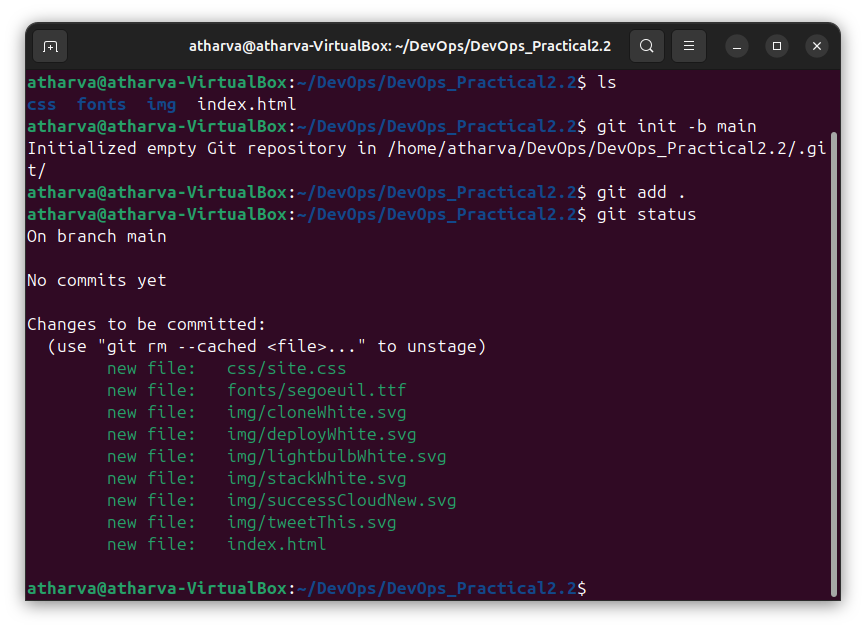


2.2

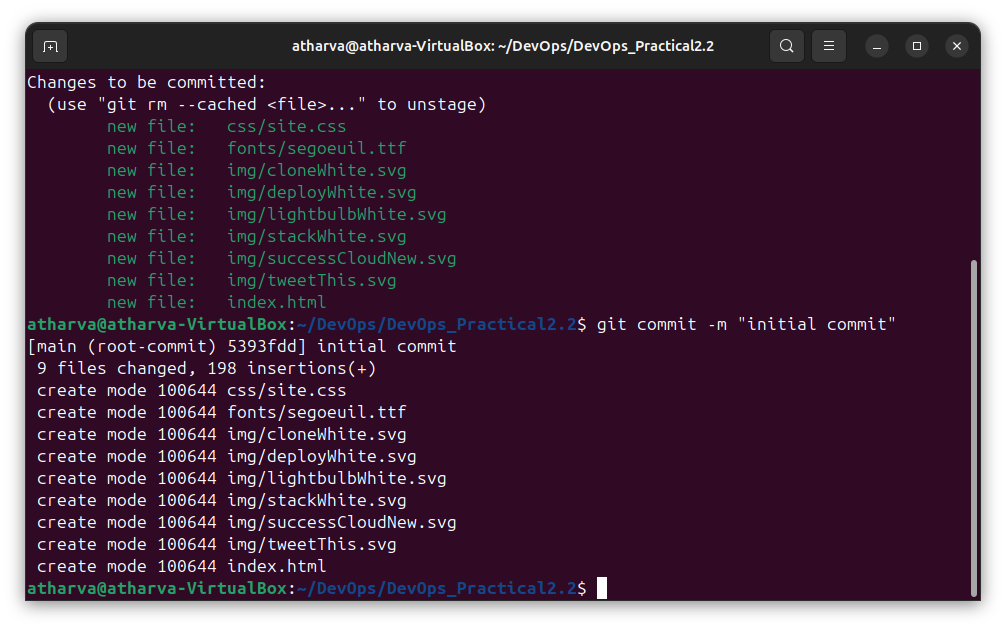
Step 1: Create a local repo and initialize a git



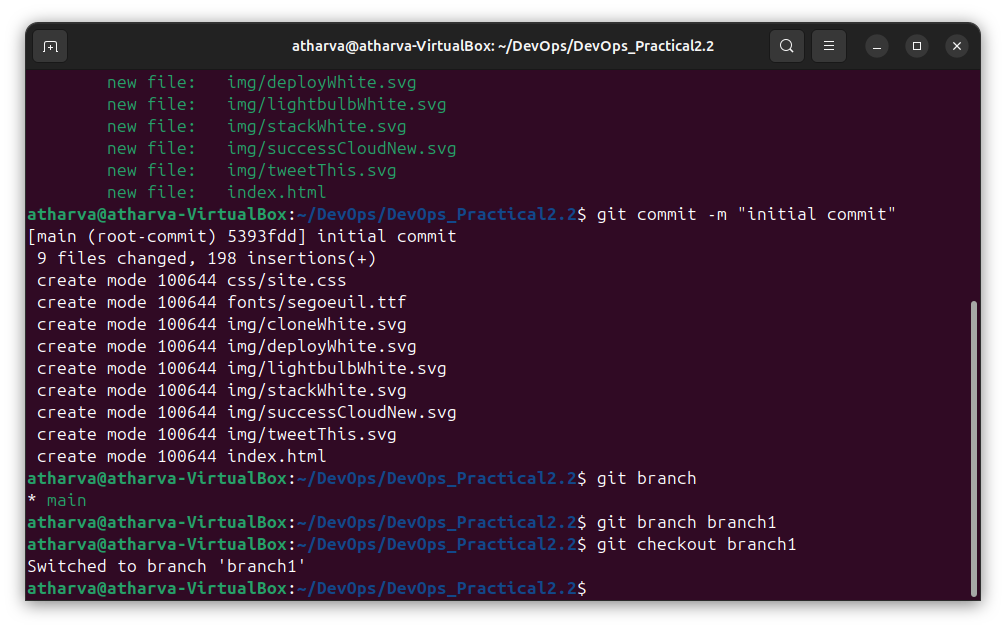
Step 2: Add all the files to staging area and use status command



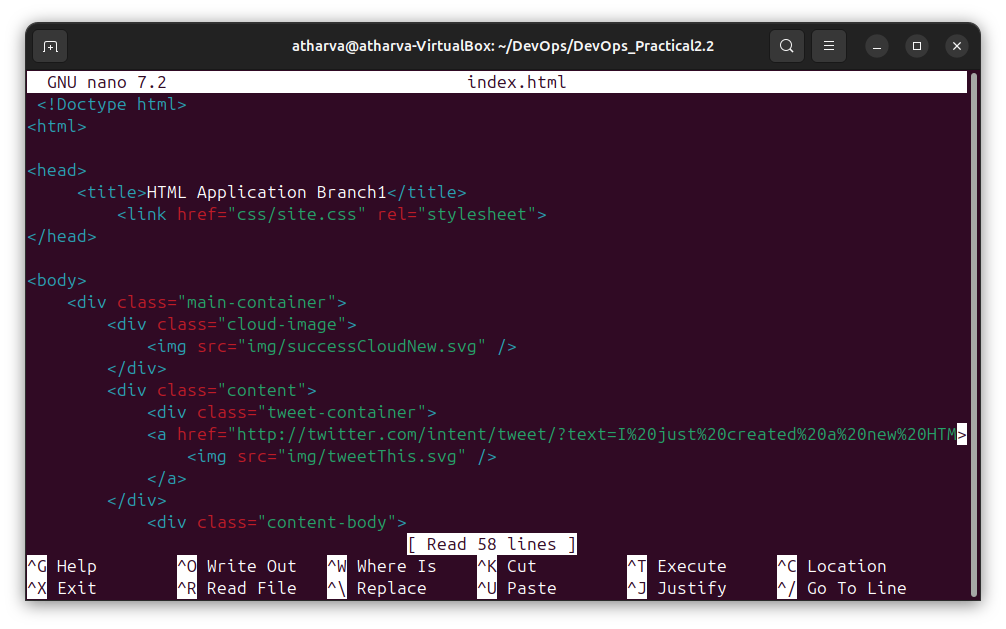
Step 3: Make initial Commit



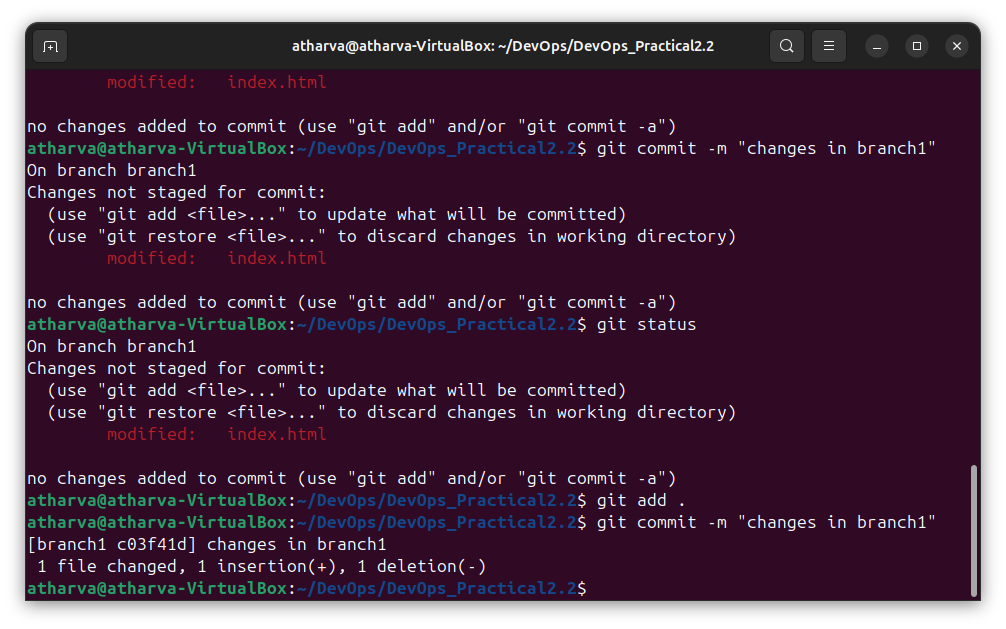
Step 4: create a new branch named branch1 and move to that branch



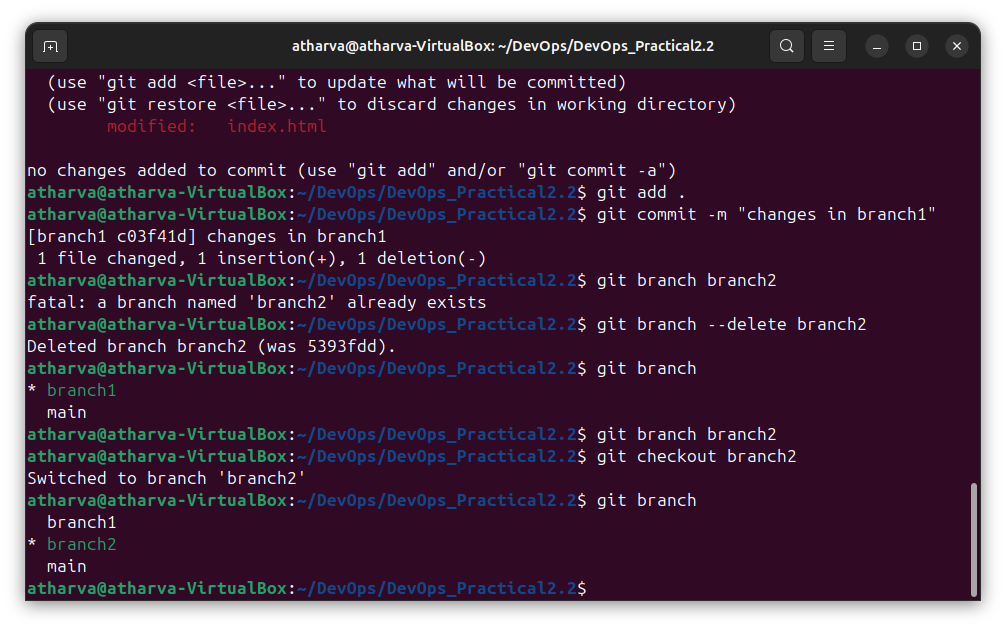
Step 5: make changes to the index.html file in branch1



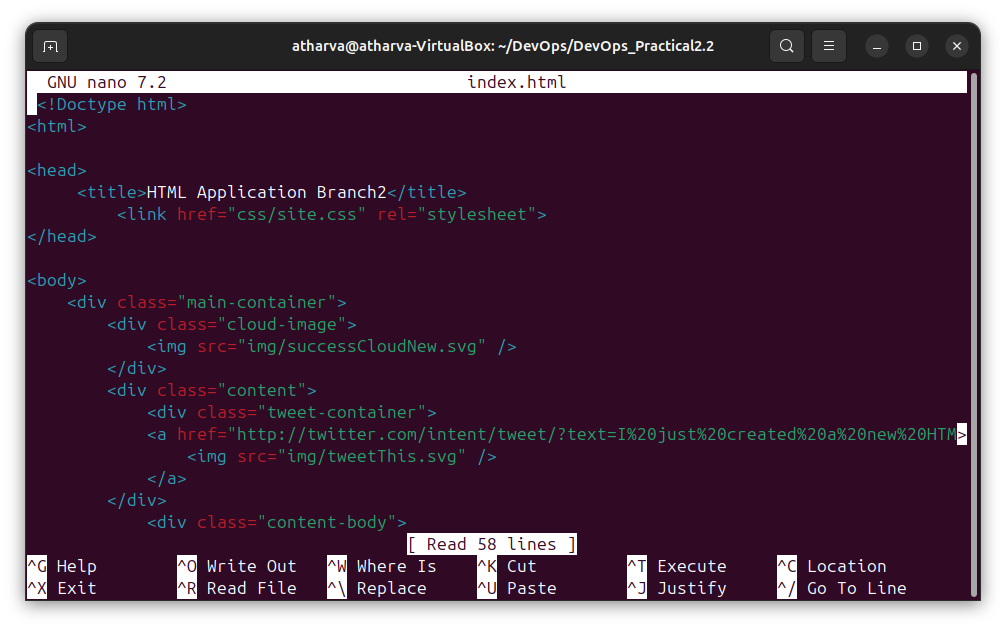
Step 6: commit the changes in branch1



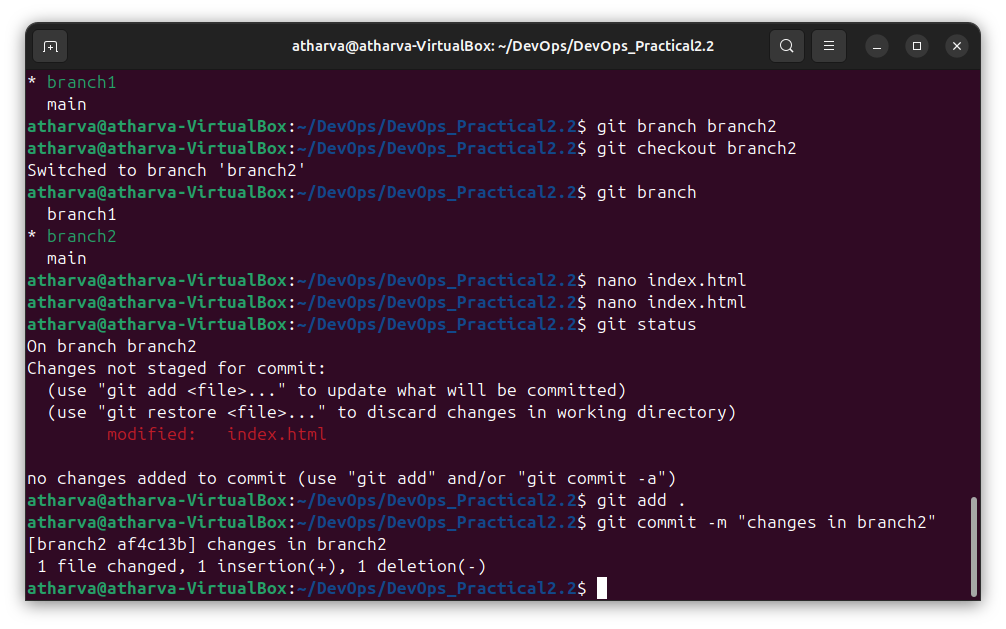
Step 7: create a new branch named branch2 and move to that branch



Step 8: make changes to the index.html file in branch2



Step 9: commit the changes in branch2



Step 10:

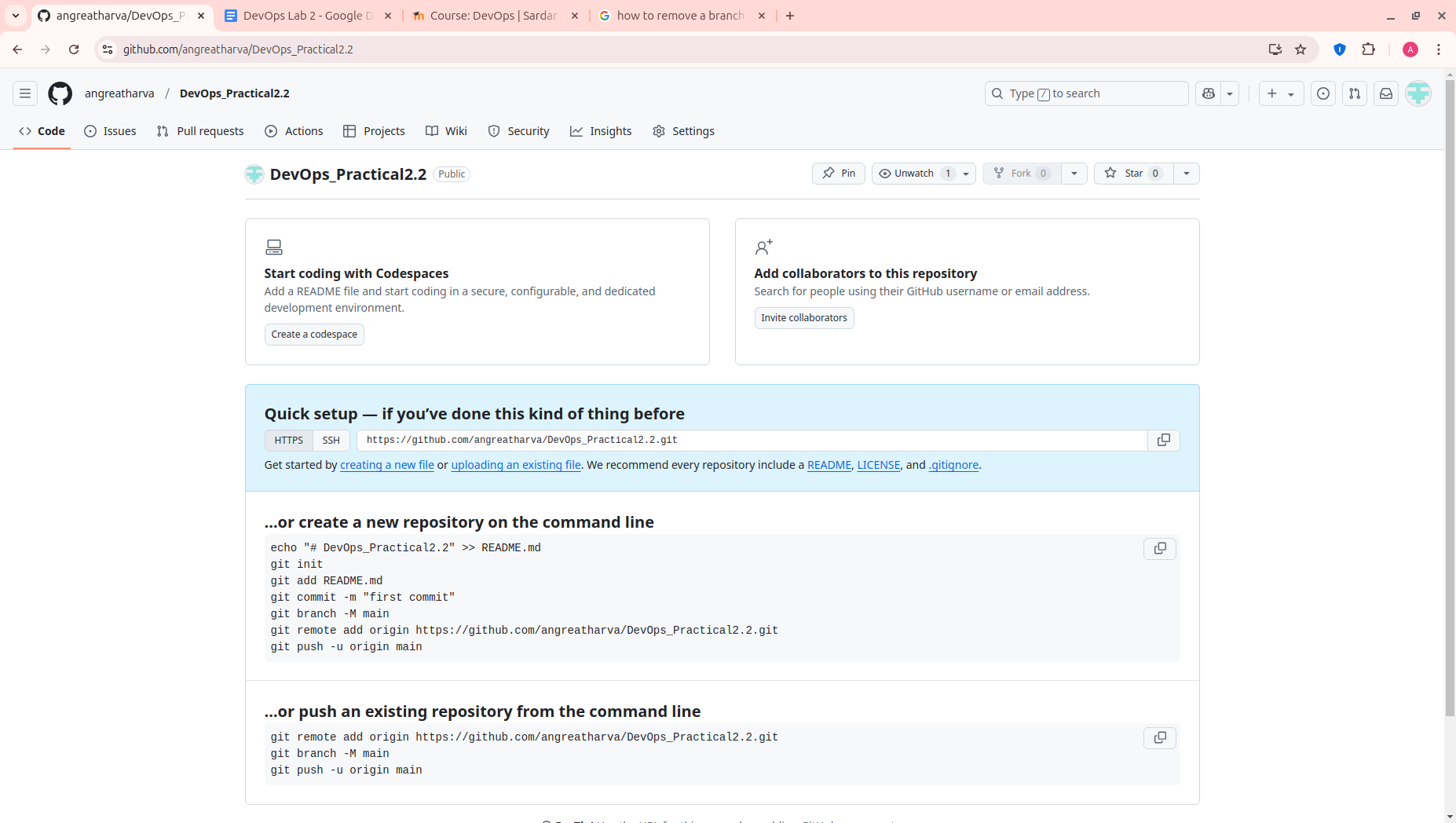
So now we have 3 branches -> main, branch1, branch2 each having different title in the index.html file.

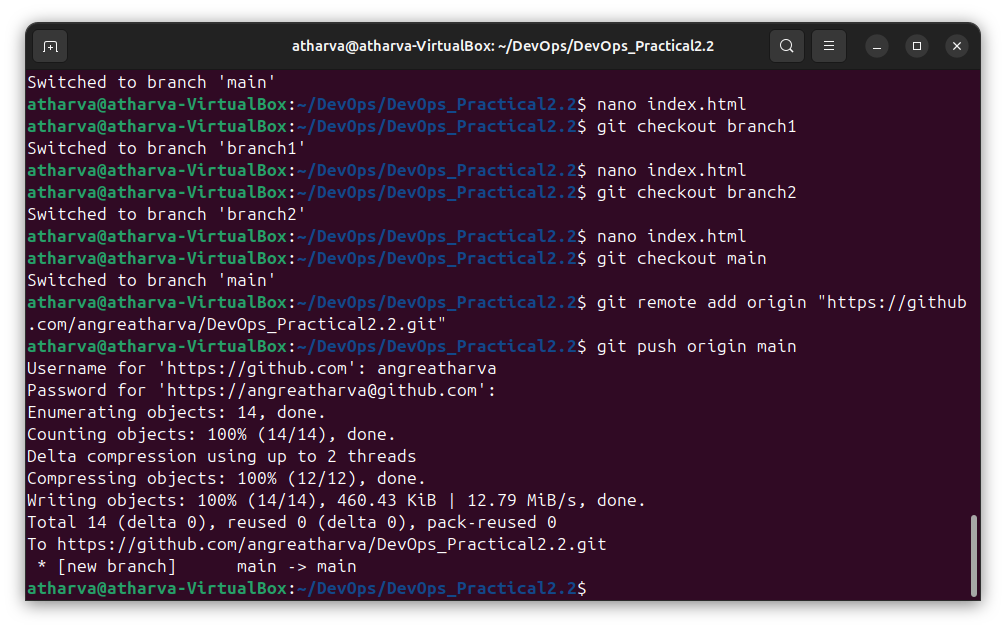
Switch to main branch and add the remote of the git repository

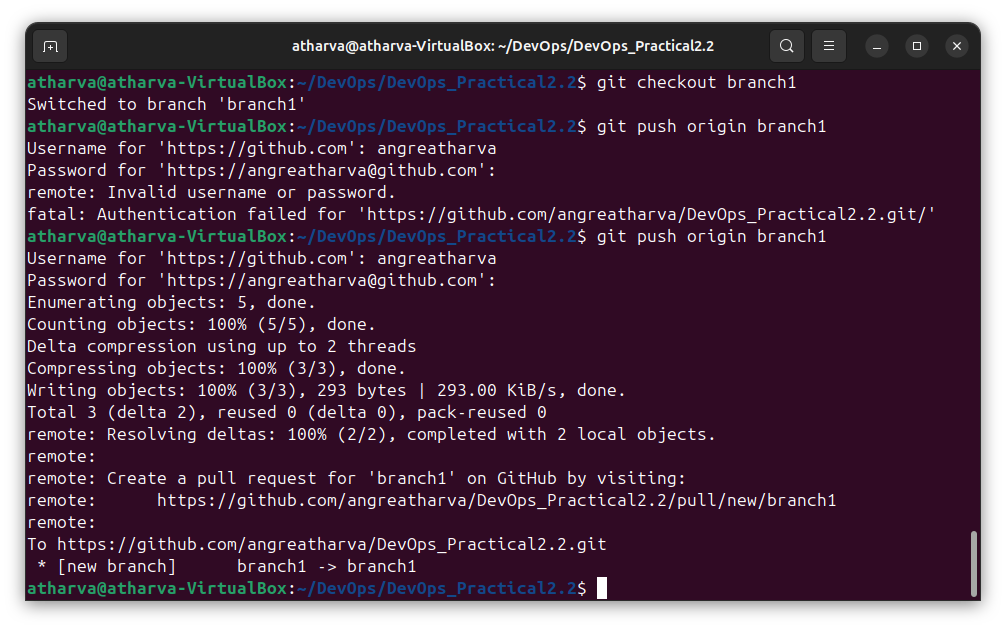
Push main branch

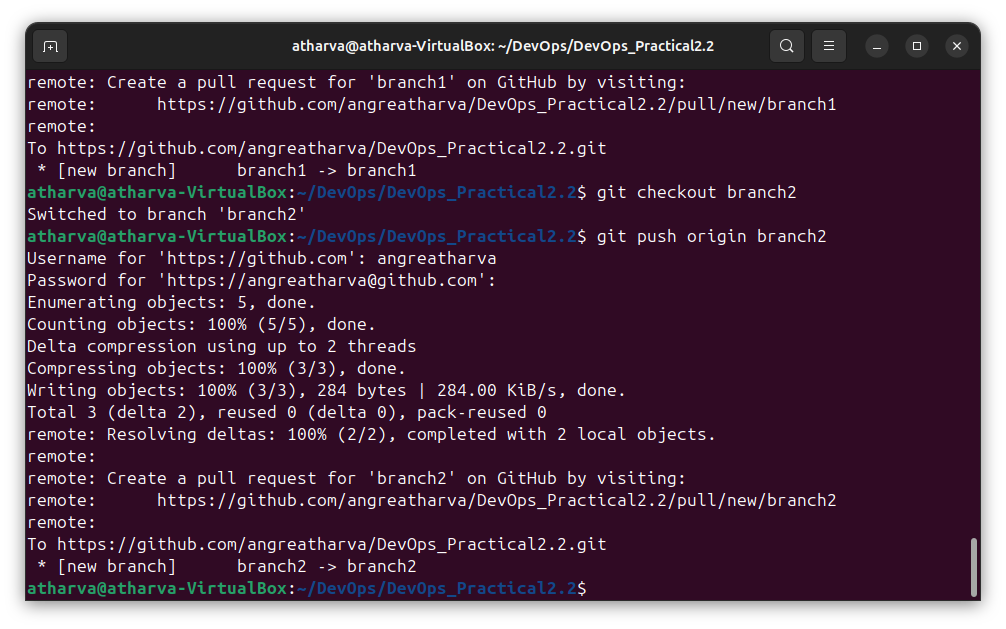
Switch to branch1 and push it

Switch to branch2 and push it

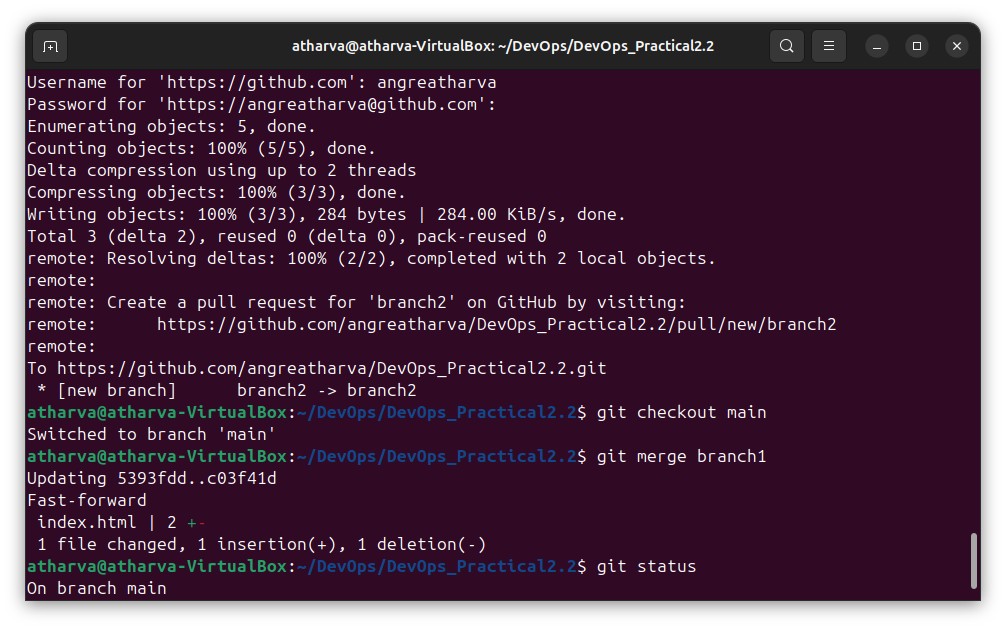




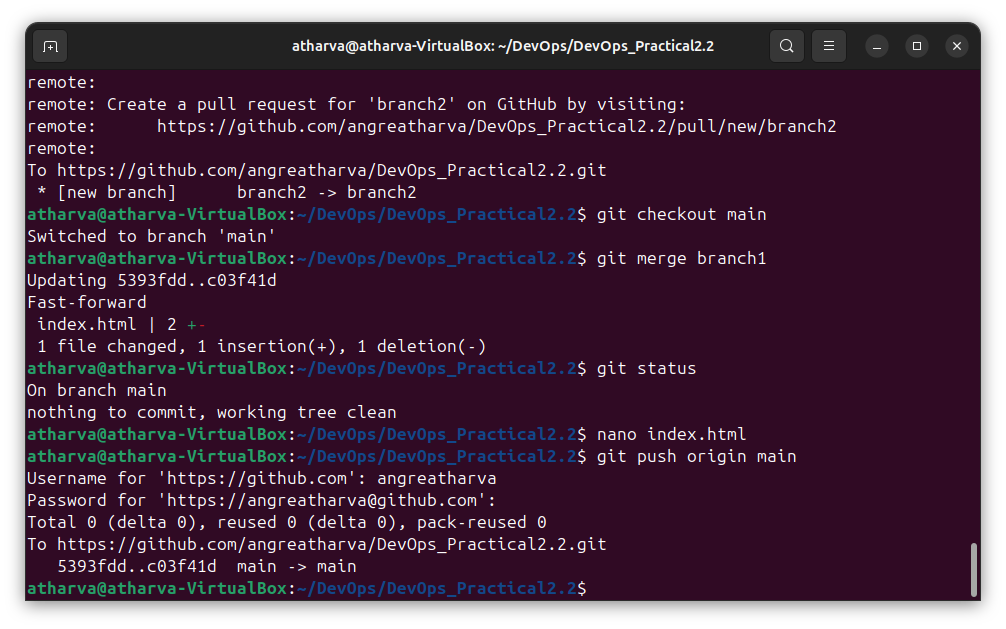




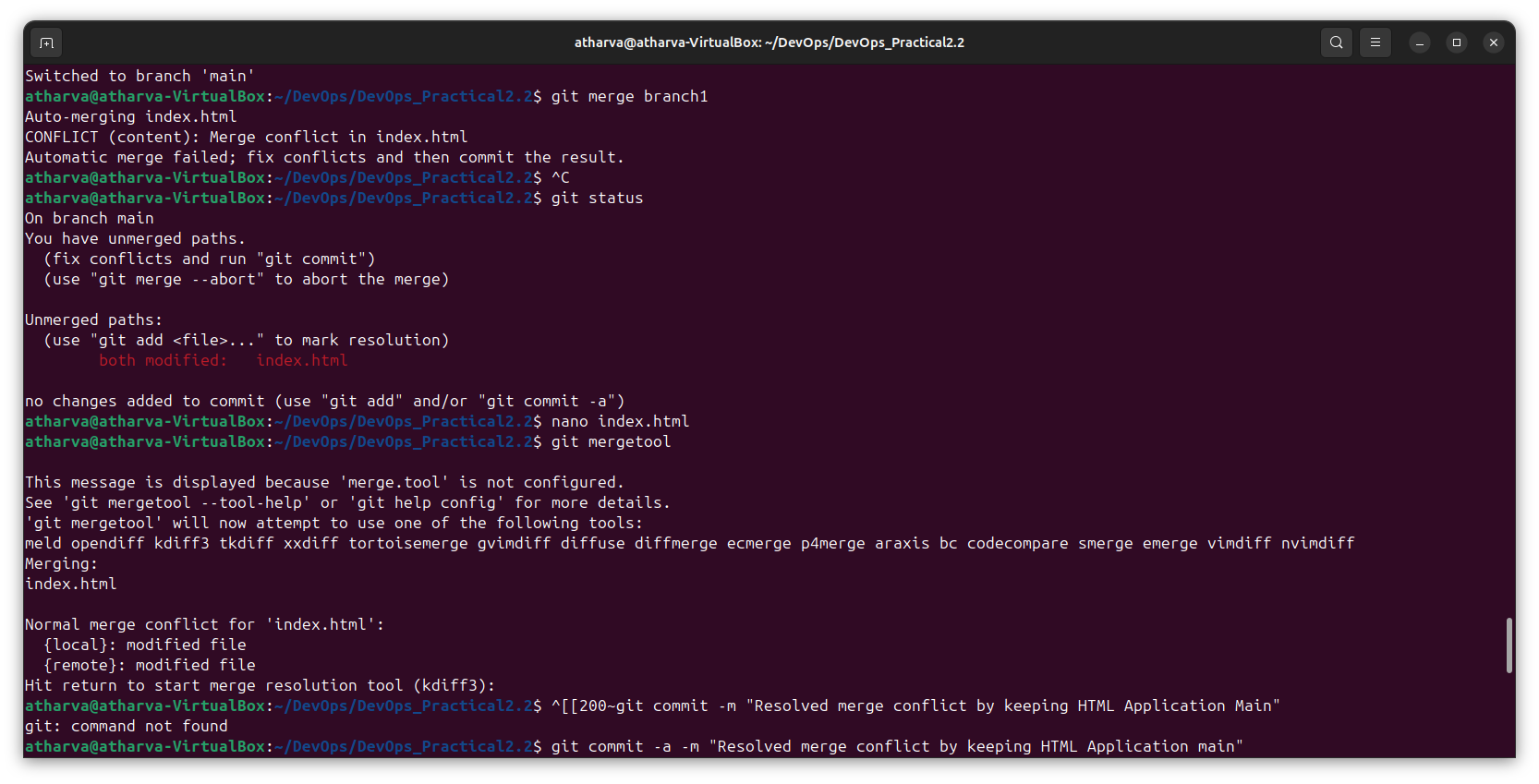
Step 11: Now to move to main branch and then merge branch1 into it



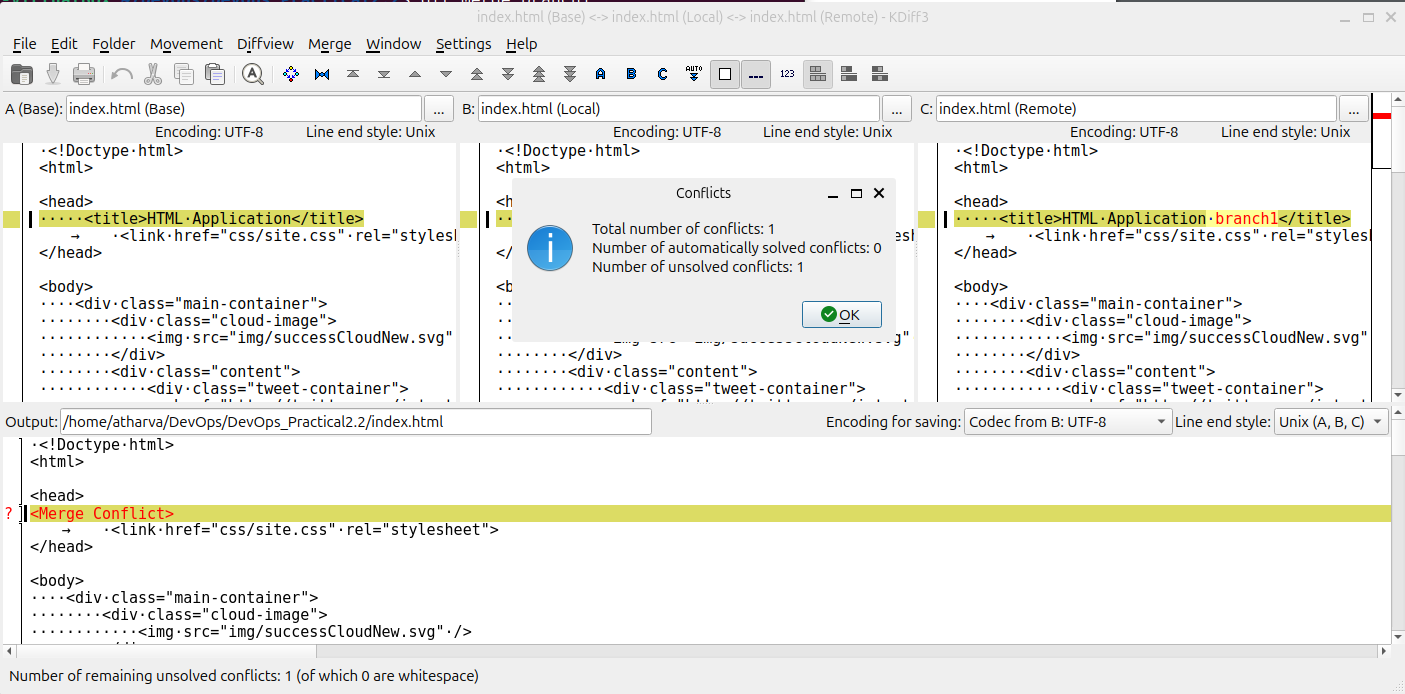
Also push it to the main branch



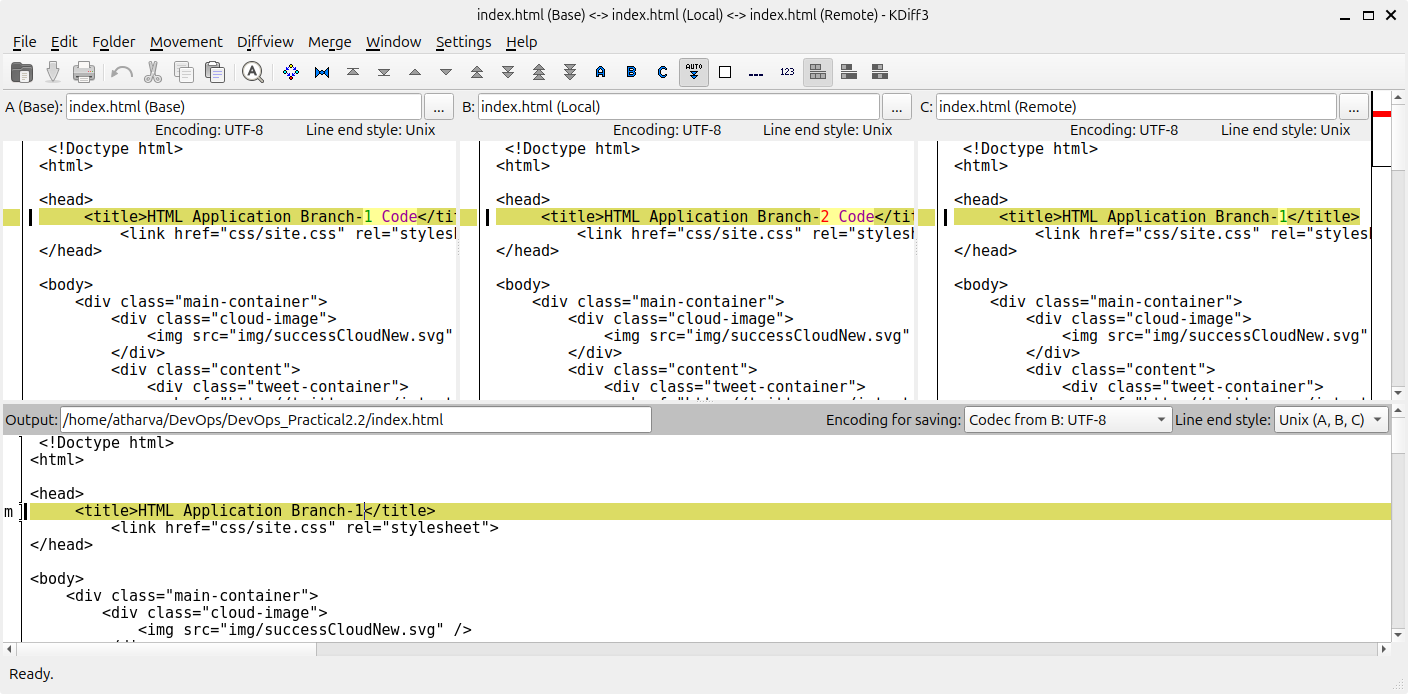
Step 12: Now try to merge branch1



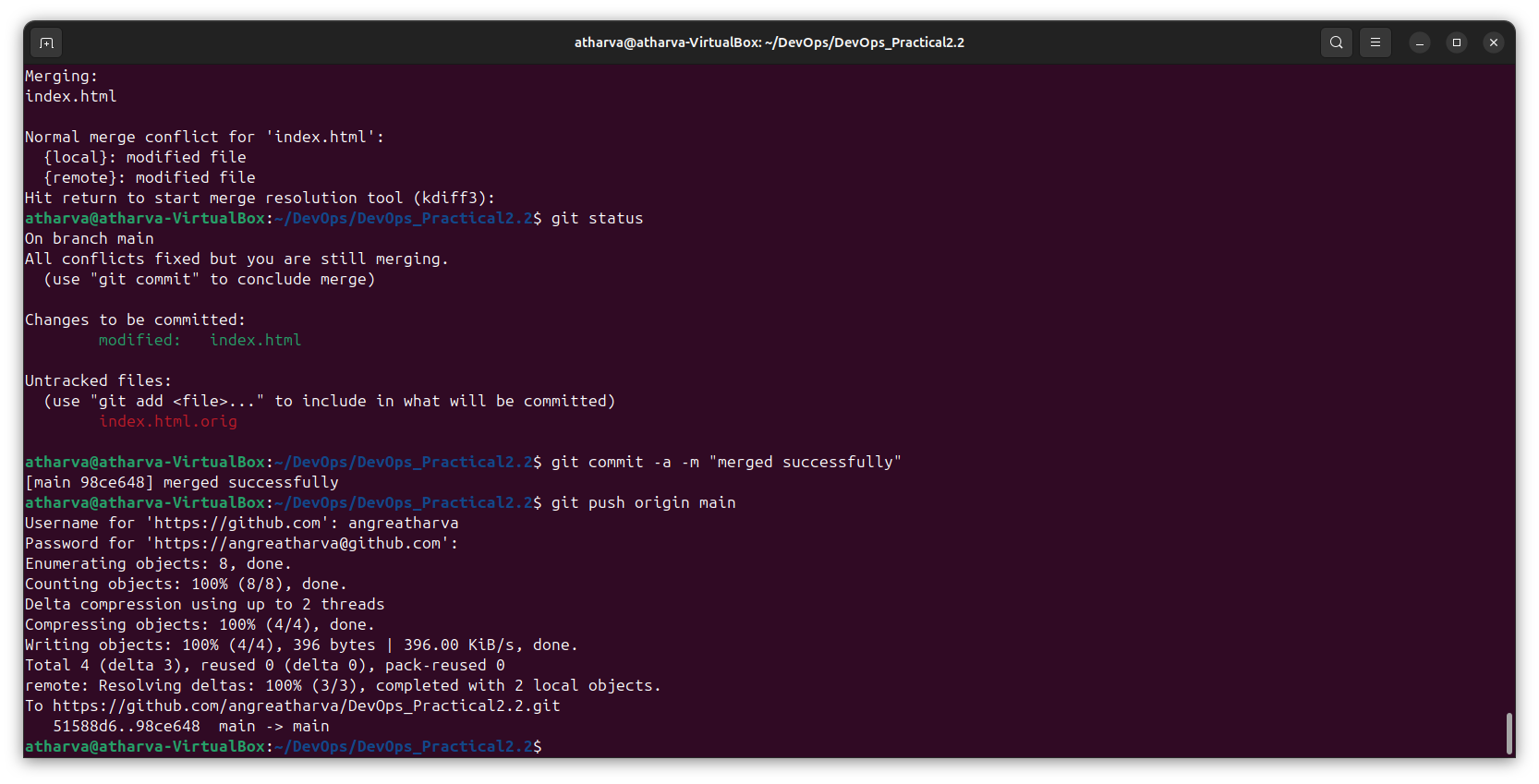
Step 16: Open the kdiff tool



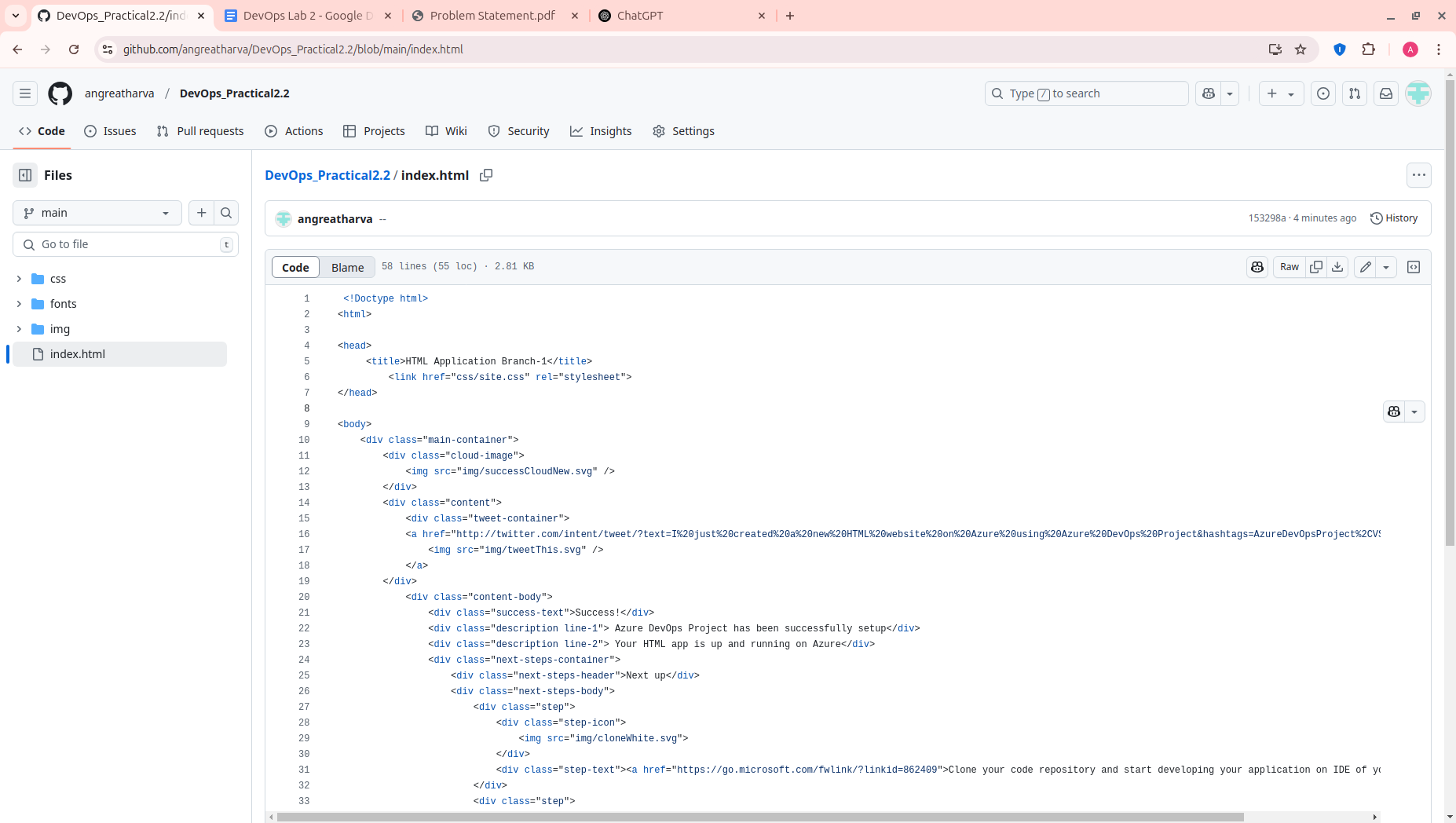
Step 17: Select which line u want to involve from with file and resolve the conflict this process has to be done manually



Step 19: Push the changes to github.



Step 20: Check if the changes are made in github



**Conclusion**: In this practical Lab, we gained hands-on experience in creating different project versions, managing branches, and establishing both local and remote repositories using terminal commands. We also learned the essential workflow of committing local changes to the global repository. Additionally, we acquired skills in efficiently resolving conflicts using tools like kdiff. Overall, this session equipped us with essential techniques for structured version control and collaborative development.