

**Name : Reena Qureshi**

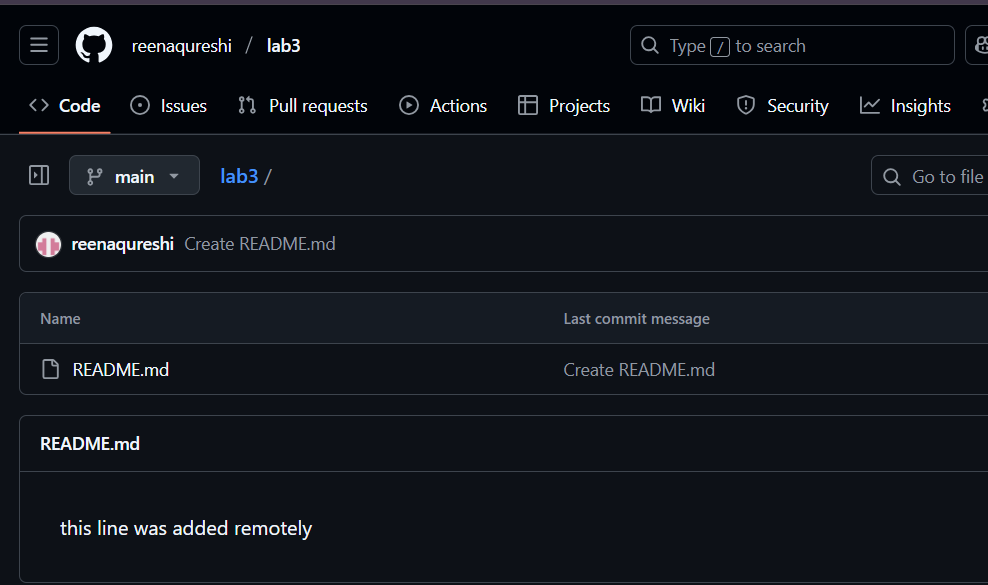
**Reg No: 2023-BSE-052**

**Section: V-B**

**Cloud Computing: Lab 3**

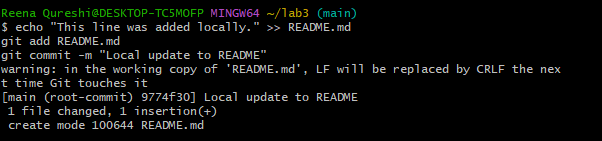
**Task 1 – Handling Local and Remote Commit Conflicts (Pull vs Pull --rebase)**

Open your repository on GitHub and edit the README.md file directly in the browser.



On your local machine, open the same repository folder.

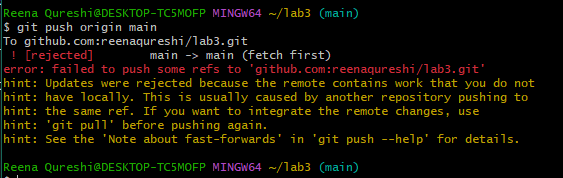
Edit the same README.md file locally



Try to push:

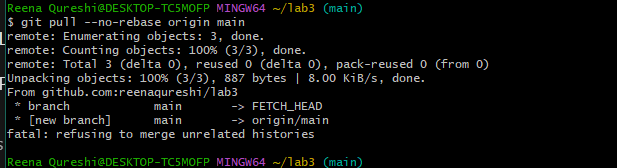
git push origin main

You’ll see an error message

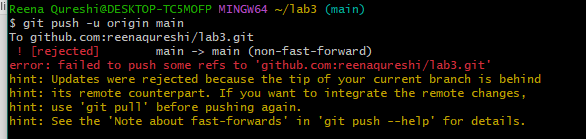


To fix this, pull the latest changes from remote:

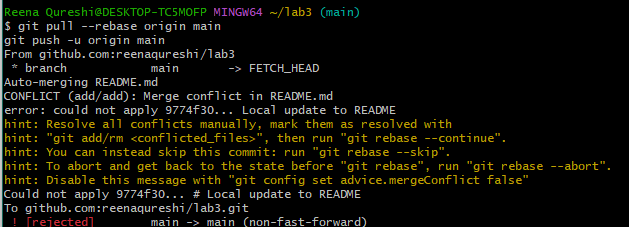
git pull --no-rebase origin main



Push again:

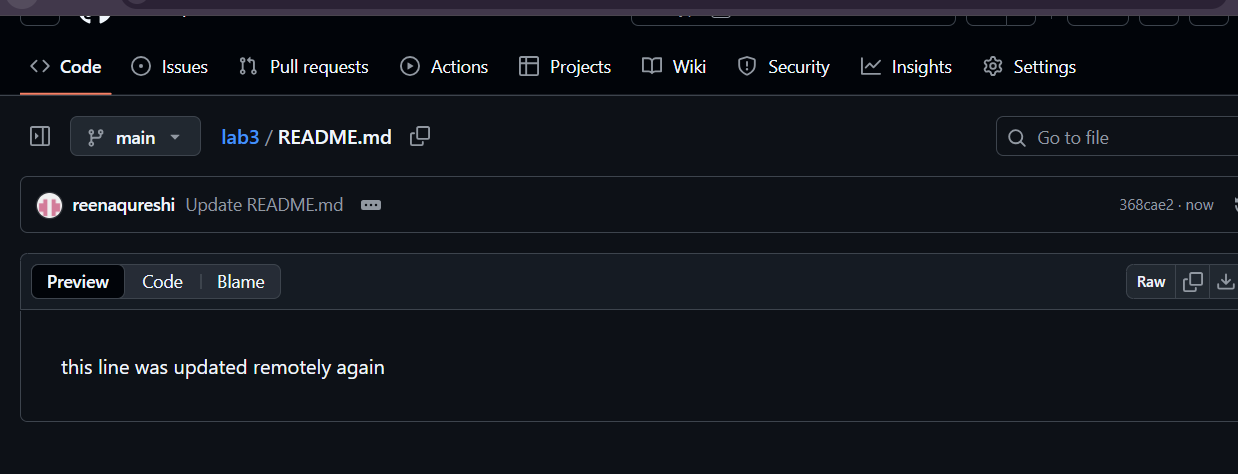


 this time use rebase instead of merge:

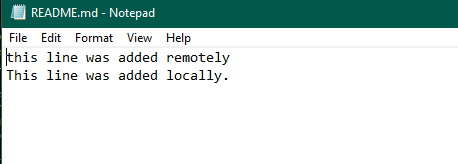


**Task 2 – Creating and Resolving Merge Conflicts Manually**

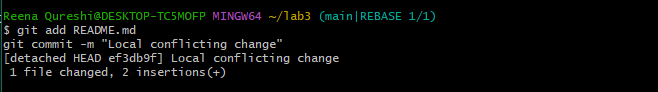
On GitHub (remote), open your README.md file and change an existing line.



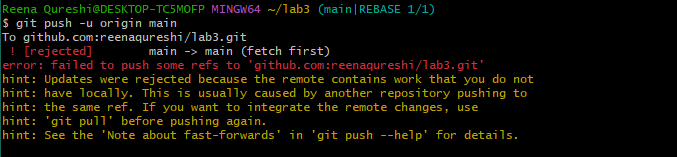
On your local machine, edit the same line in the same file but make a different change



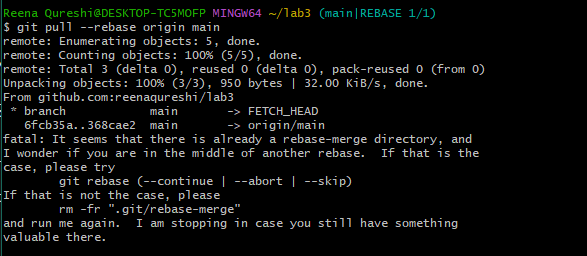
Stage and commit your local change:



Try to push:



Pull with rebase to bring in remote changes:



Open the README.md file in your editor — you’ll see conflict markers like this:

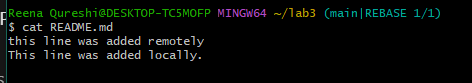
<<<<<<< HEAD

This line was updated locally at the same time.

=======

This line was updated remotely again.

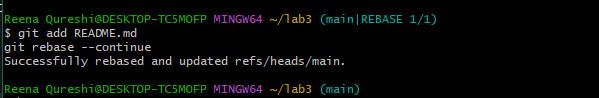
>>>>>>> origin/main



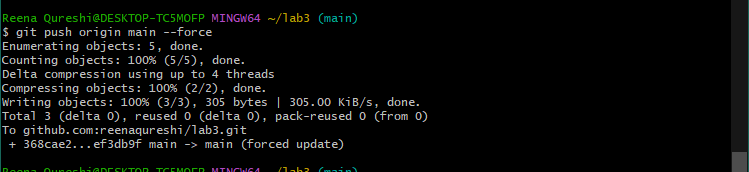
After fixing the file, mark the conflict as resolved and continue the rebase:

git add README.md

git rebase --continue



Finally, push your changes:



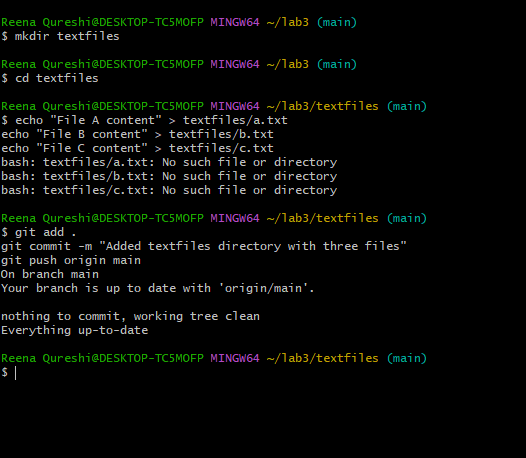
**Task 3 – Managing Ignored Files with .gitignore and Removing Tracked Files**

Create a new folder named textfiles inside your repository:

mkdir textfiles

Inside the textfiles folder, create three text files:

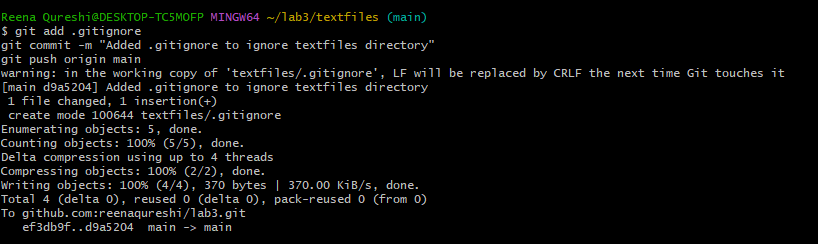
Add and commit the new directory



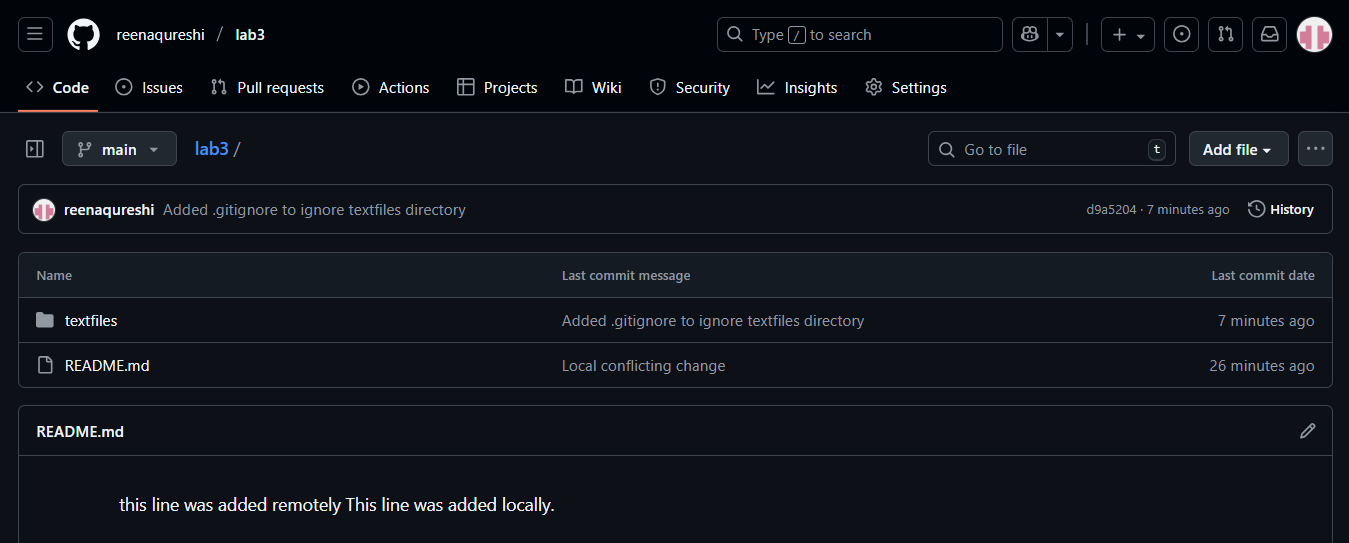
Now, create a .gitignore file in the root of your repository



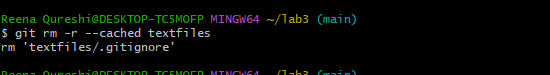
Add and commit the .gitignore file:



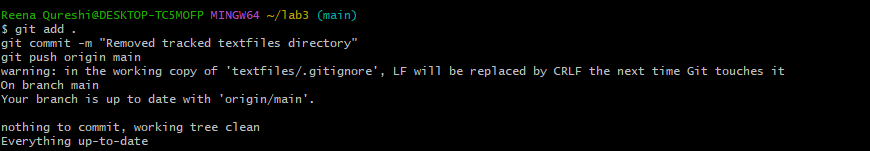
Go to GitHub and check your repository — notice that the textfiles directory is still visible on the remote.



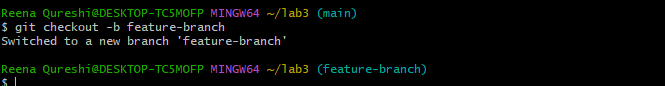
To remove already tracked files from Git (but not from your local system), run:

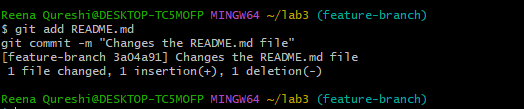


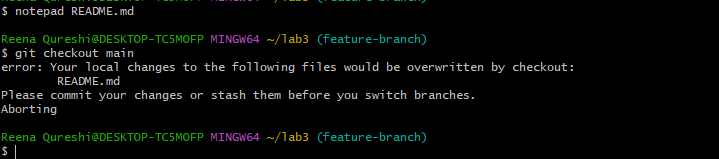
Commit and push again:

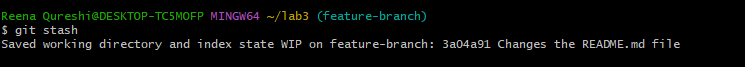


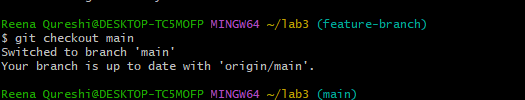
**Task 4 – Create Temporary Changes and Use git stash**

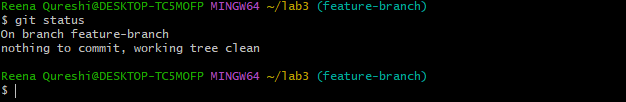


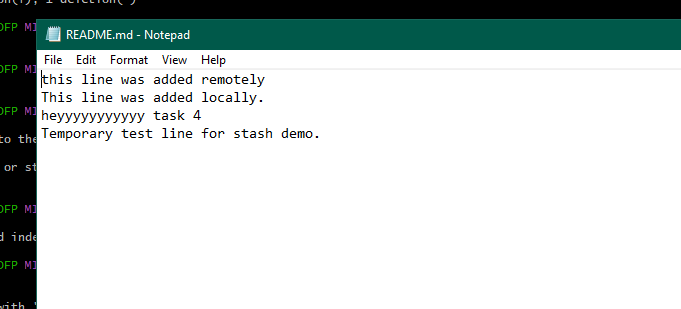








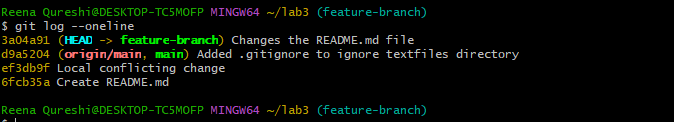
  




**Task 5 – Checkout a Specific Commit Using git log**

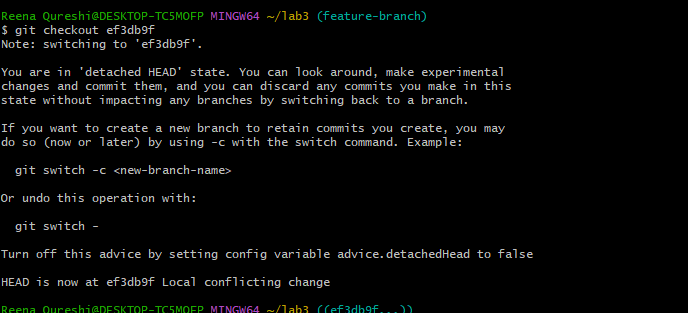
View commit history:

git log --oneline



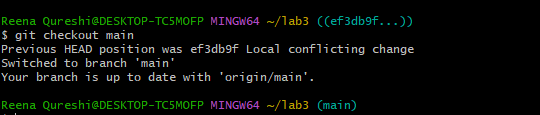
Copy any previous commit hash (e.g., a1b2c3d).

Checkout that commit:



To return to your main branch:

git checkout main



Task 6 – Resetting Commits (Soft vs Hard Reset) (With Verification Steps)

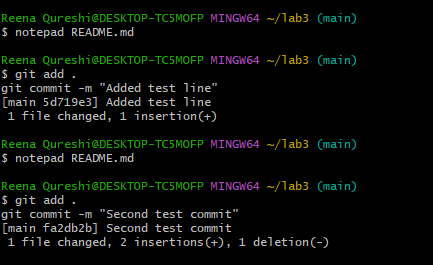
Add a new line in any file and commit it:

# Edit any file (e.g., README.md), add a line

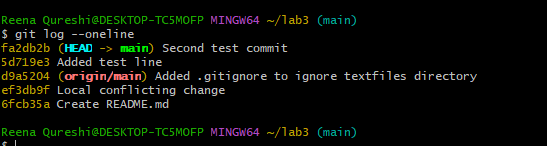
git add .

git commit -m "Added test line"

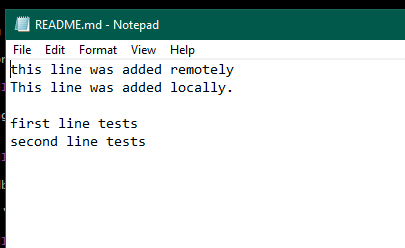
Add another change and commit again:



View the history before reset



Check file contents for both changes:

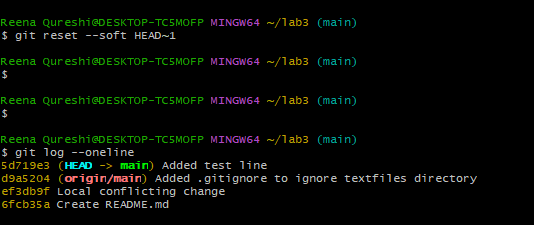


Perform a soft reset (keeps changes in working directory):

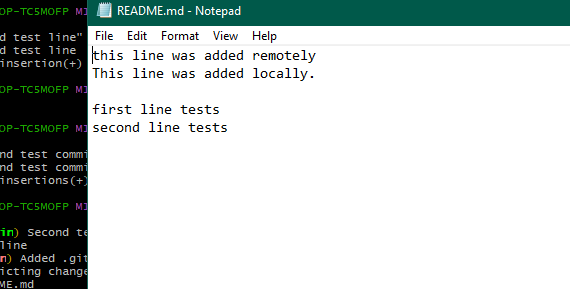
git reset --soft HEAD~1

heck commit history after soft reset:

git log --oneline

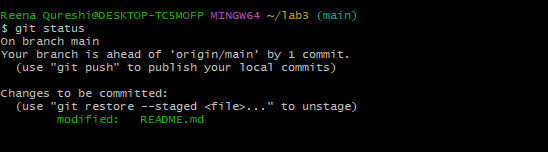


Verify changes in the file



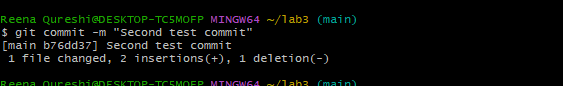
Check git status:

git status



Perform commit

git commit -m "Second Test commit"



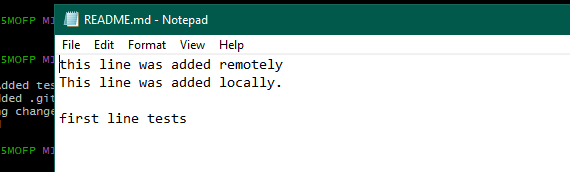
Perform a hard reset (discards changes completely):

git reset --hard HEAD~

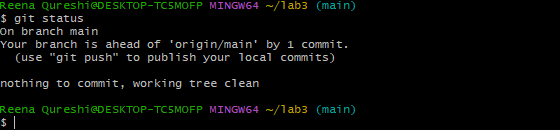
heck commit history after hard reset:

git log --oneline

Verify changes in the file:

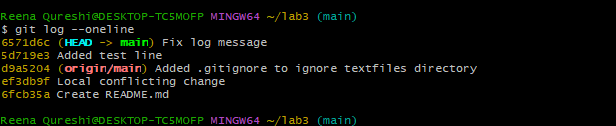


Check git status:

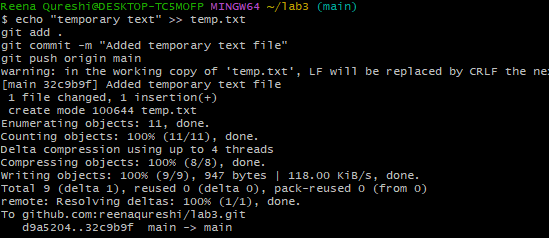


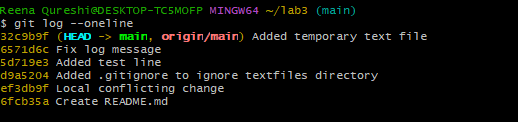
**Task 7 – Amending the Last Commit**

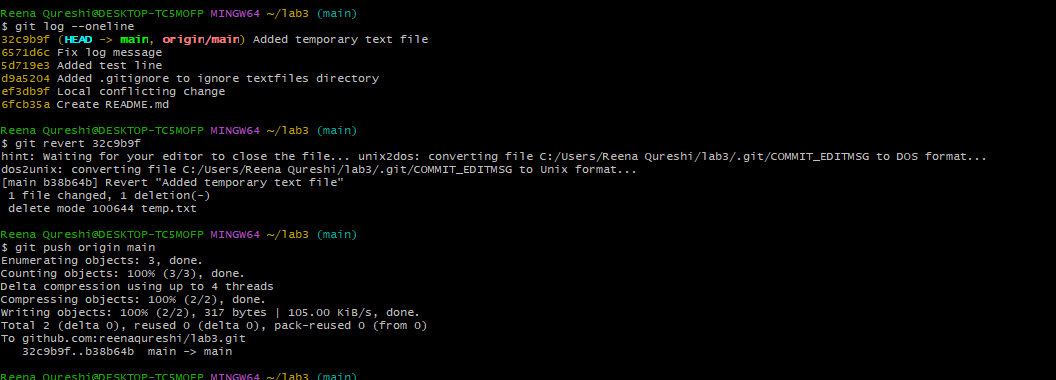




**Task 8 – Reverting a Commit (Safe Undo on Remote Branch)**







**Task 9 – Force Push (With Caution)**

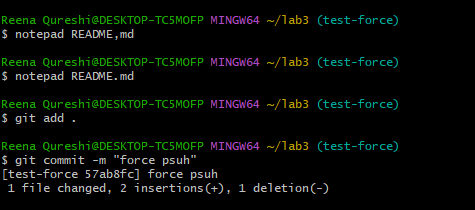
Create a new branch:

git checkout -b test-force

Save a screenshot as new\_branch.png.

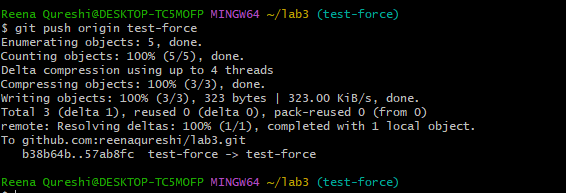
Make and commit a small change.

Save a screenshot as force\_commit.png.



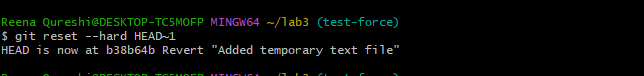
Push it:

git push origin test-force

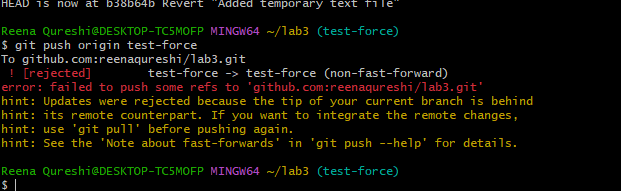


Perform a hard reset:

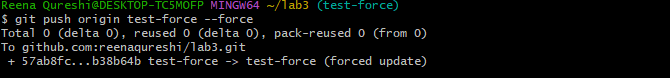
git reset --hard HEAD~1



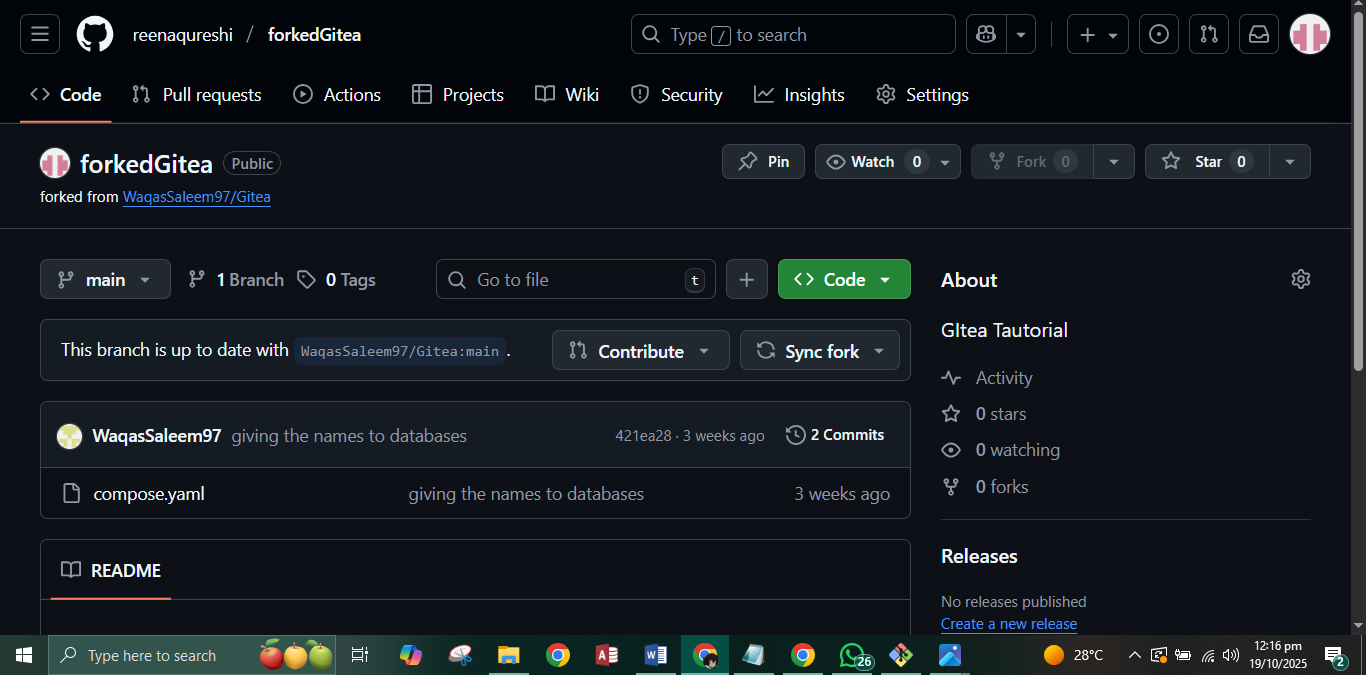
Push again

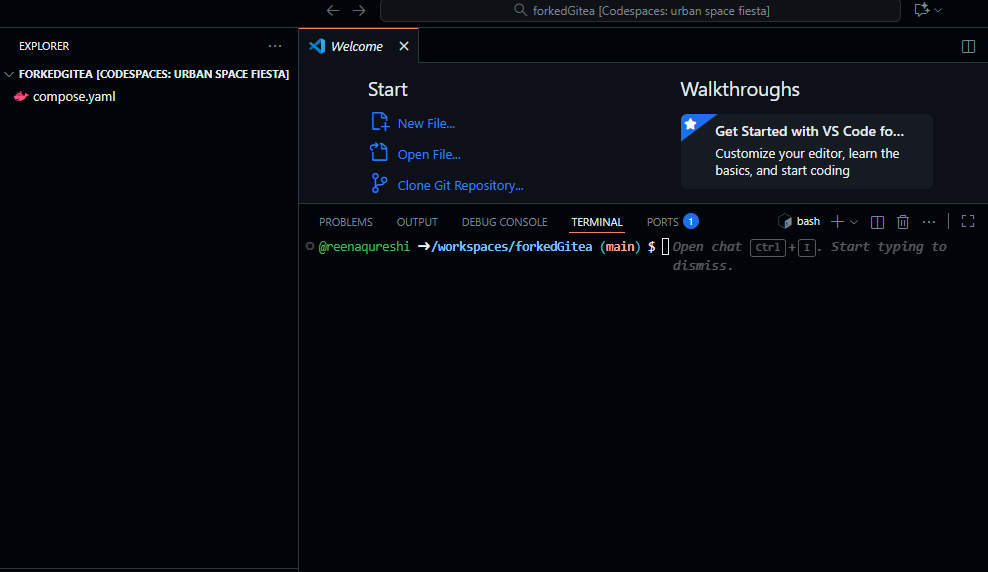


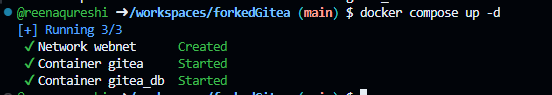
Now force-push to remote:

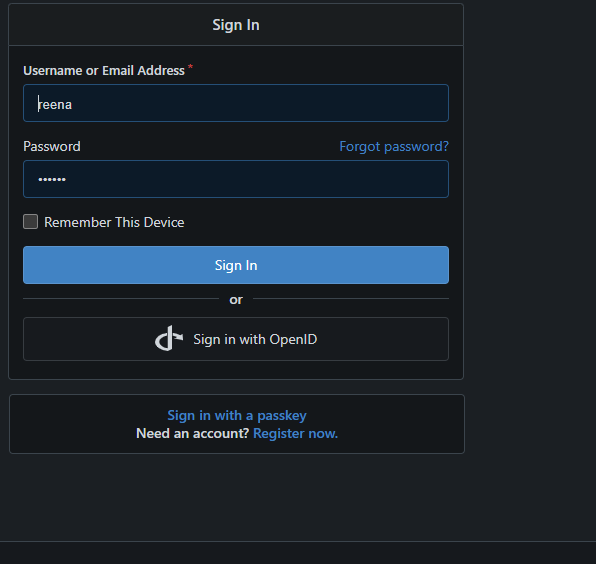


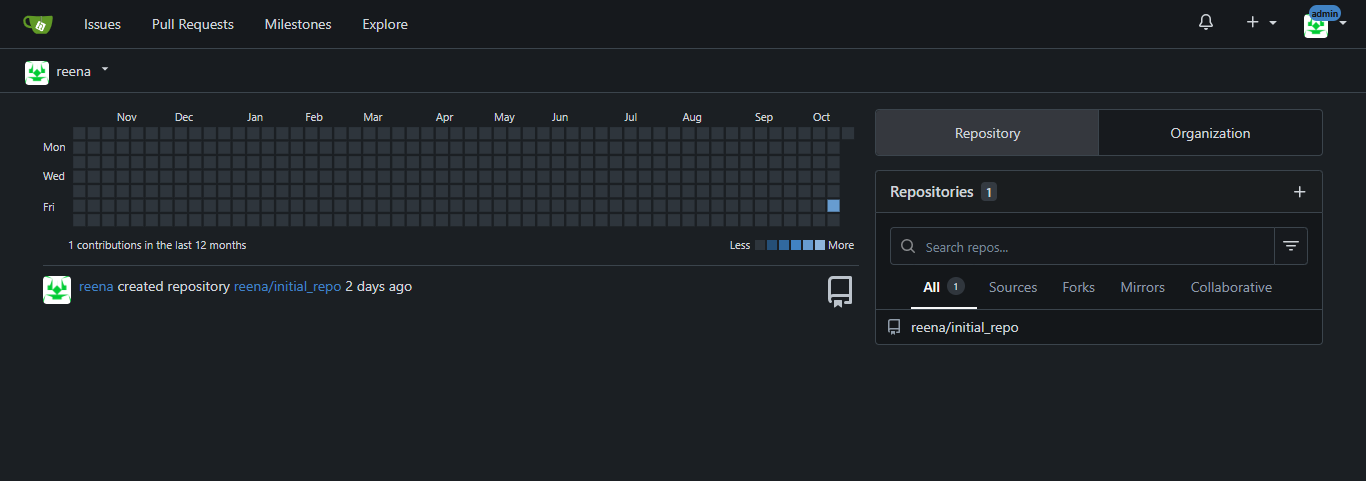
**Task 10 – Running Gitea in GitHub Codespaces via Docker Compose**

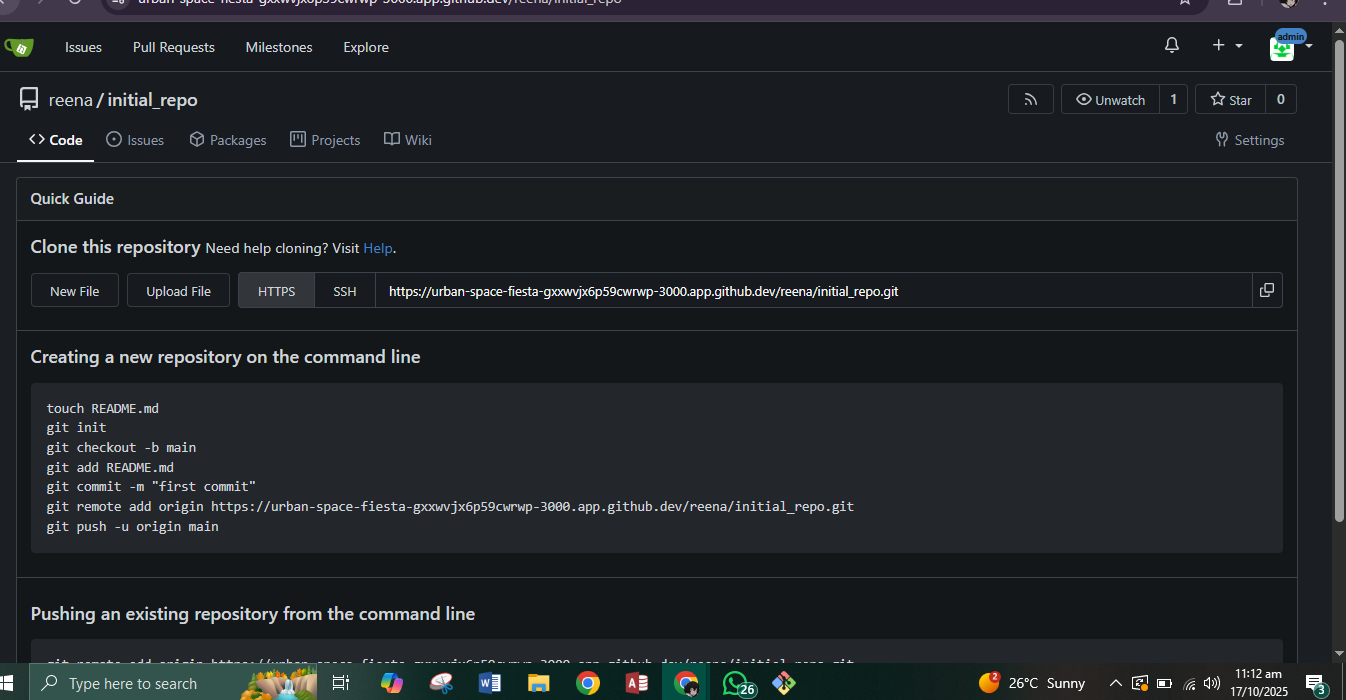






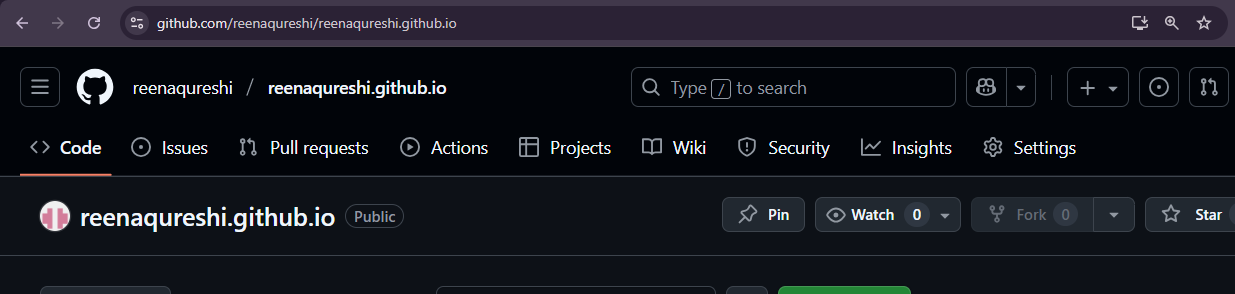




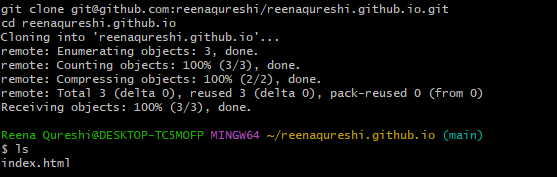


**Task 11 – Creating a GitHub Pages Portfolio Site**

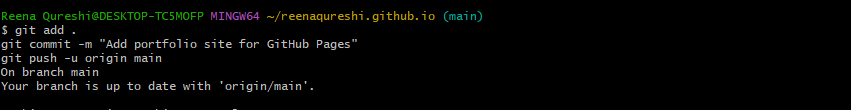
Create a GitHub Pages Repository



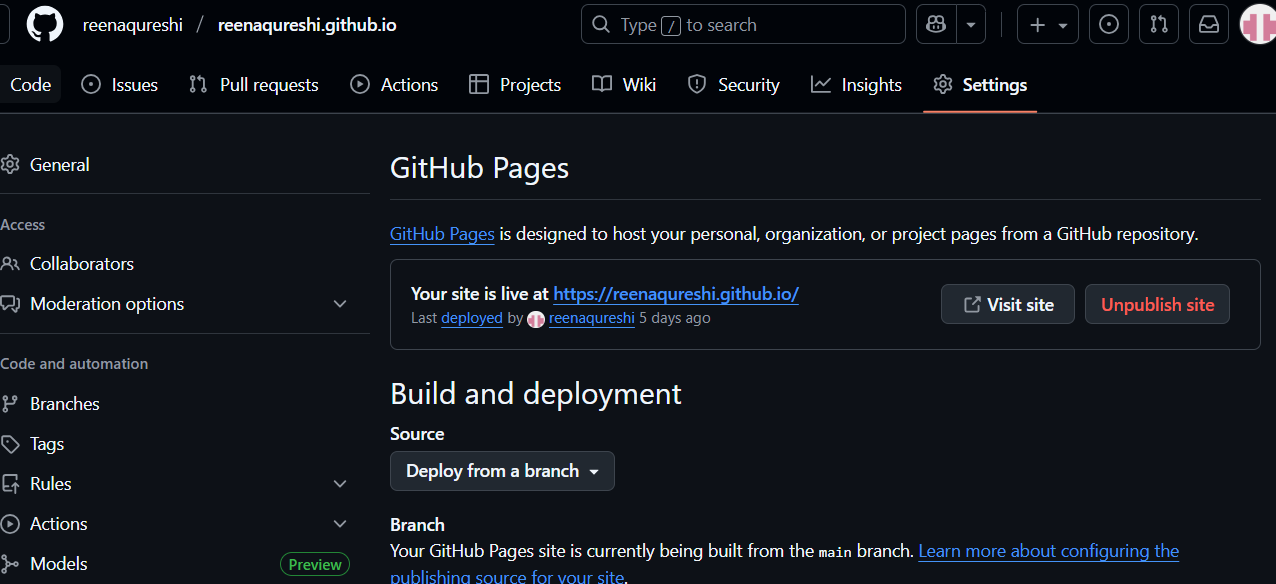
Add Static Website Code



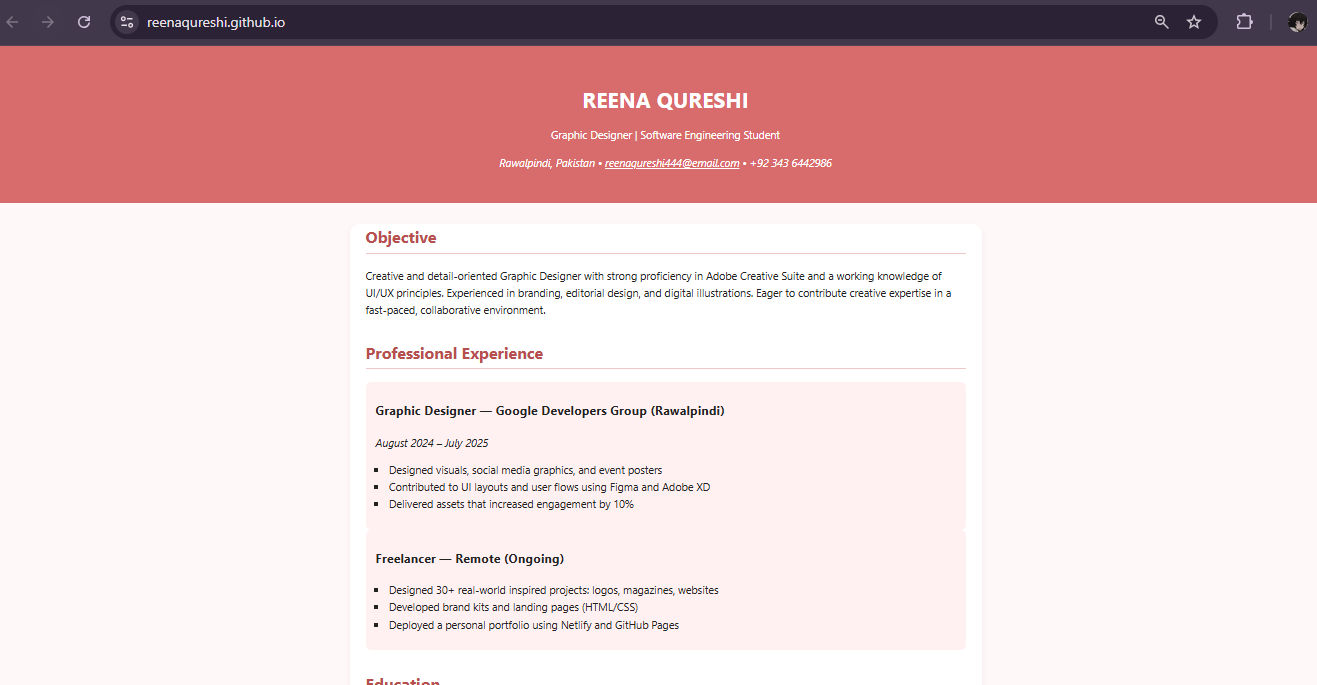
Push the Files to GitHub



Check GitHub Pages Settings

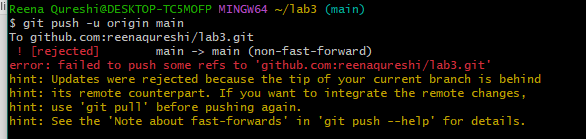


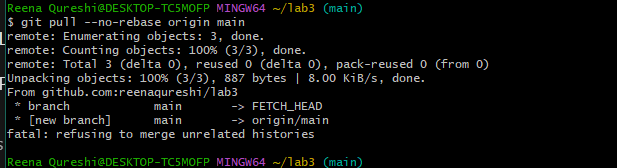
Visit Your Live Site



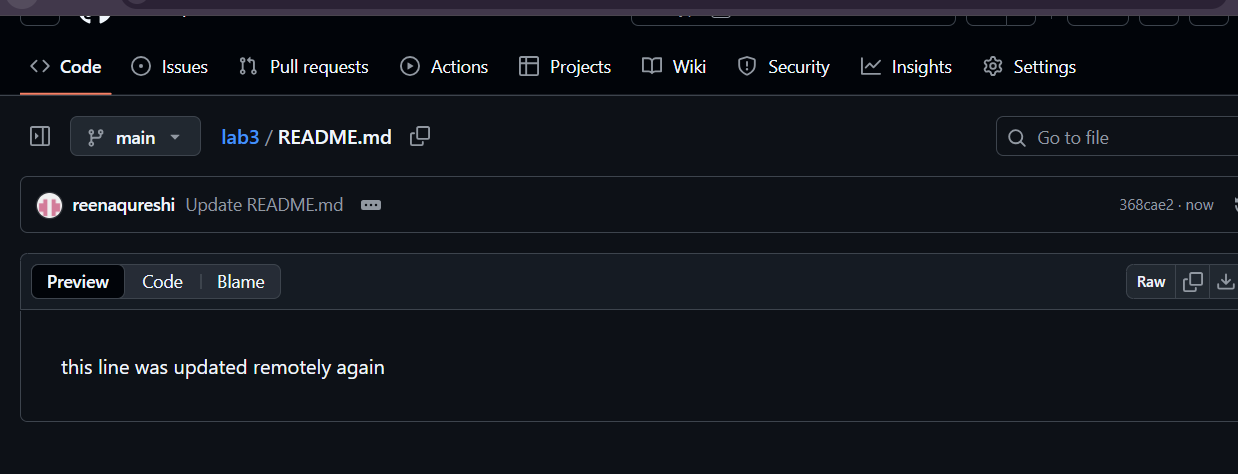
**Exam Evaluation Questions**

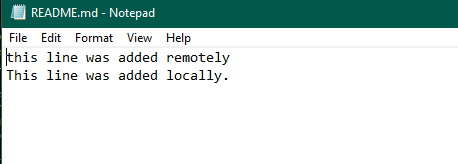
1. Local vs Remote Conflict Resolution

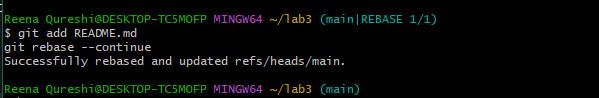




Manual Merge Conflict Handling



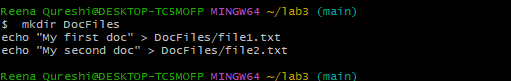


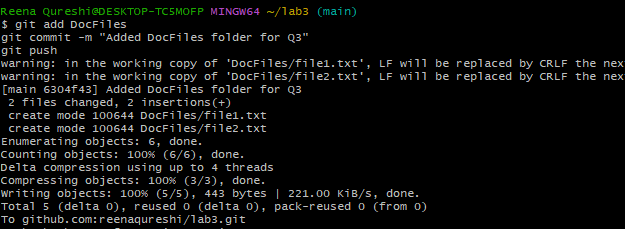


Managing Ignored and Tracked Files

Steps:

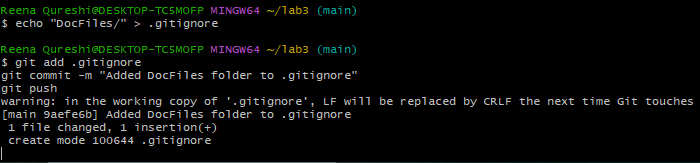
Create a new folder (e.g., DocFiles) and add several files inside



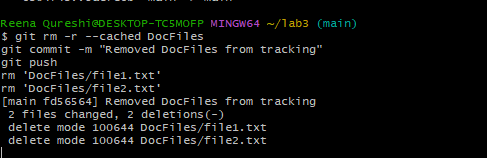
Commit and push the folder/files to GitHub. 

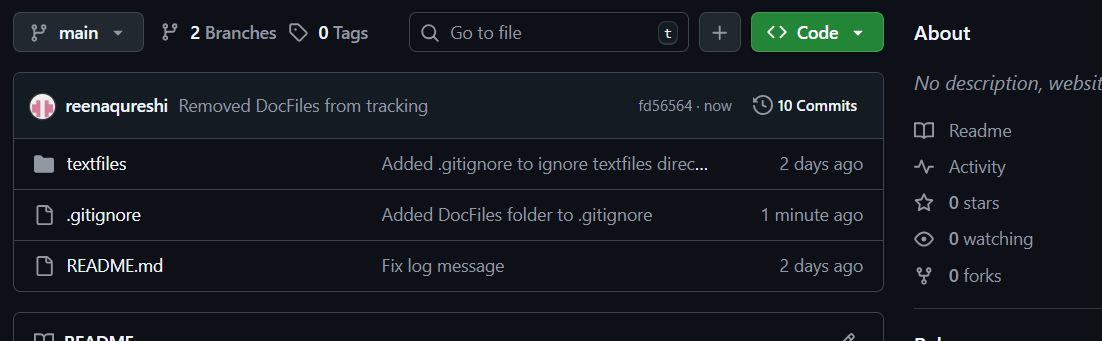
Add the folder to your .gitignore file

Commit and push the .gitignore update



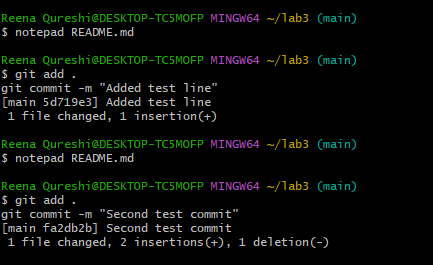
Remove the folder from tracking using git rm -r --cached <folder>

Commit and push the change, then verify the folder is no longer tracked on GitHub  


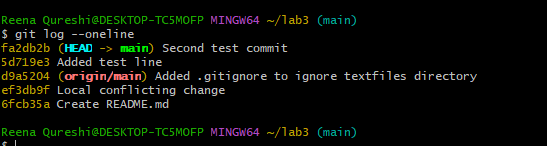


Commit History Manipulation and Recovery

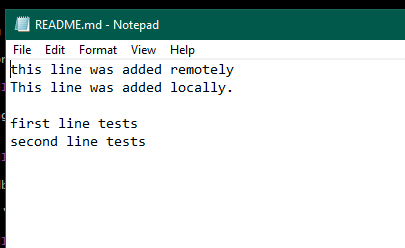
Hard and Soft reset



View the history before reset



Check file contents for both changes:

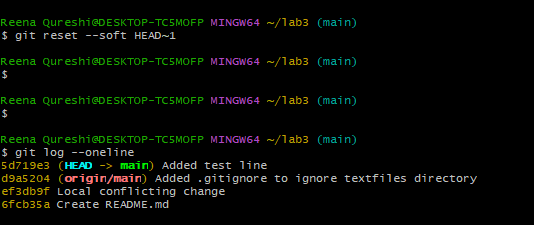


Perform a soft reset (keeps changes in working directory):

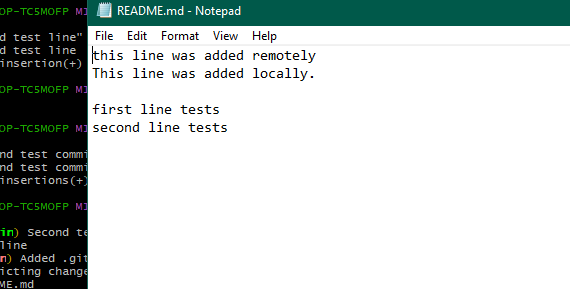
git reset --soft HEAD~1

heck commit history after soft reset:

git log --oneline

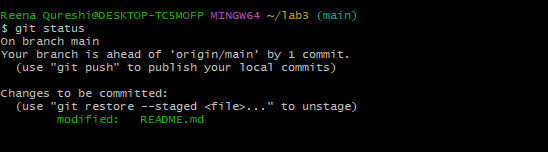


Verify changes in the file



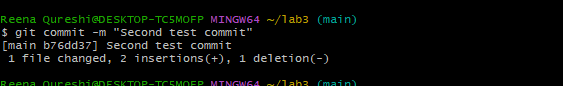
Check git status:

git status



Perform commit

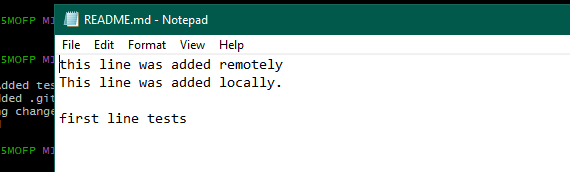
git commit -m "Second Test commit"



Perform a hard reset (discards changes completely):

git reset --hard HEAD~

heck commit history after hard reset:



Check git status:

