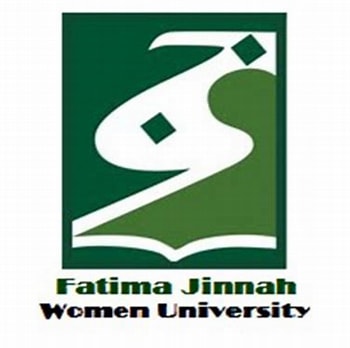
**CLOUD COMPUTING**

**LAB NO. 3**

****

**SUBMITTED TO:**

Engr. Shohaib

**SUBMITTED BY:**

**Name:** Amina Noor

**Reg No.** 2023-BSE-007

**Semester:** V-A

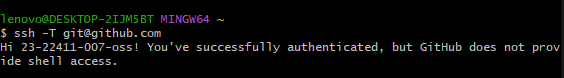
**CLOUD COMPUTING**

**LAB NO. 3**

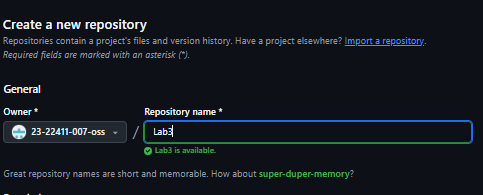
**Task List:**

**Prerequisites**

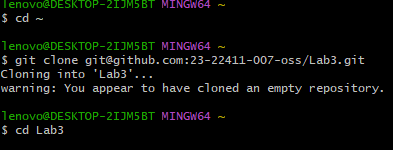
**Step 1)** Check an SSH key

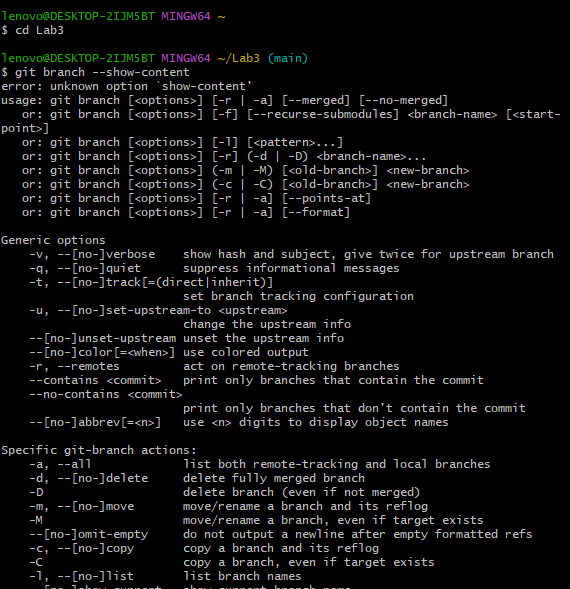


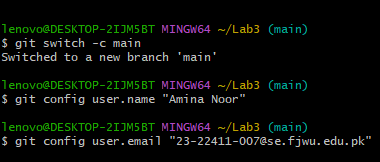
**Step 2)** Create the repo on GitHub



**Step 3)** Clone the repo locally

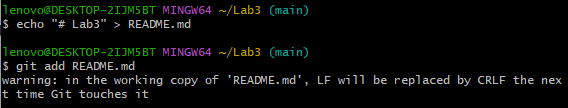


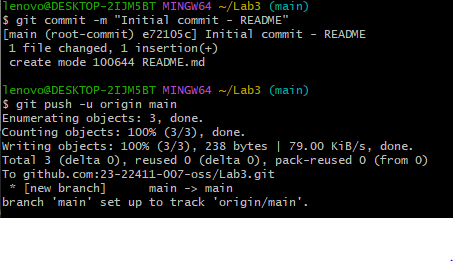


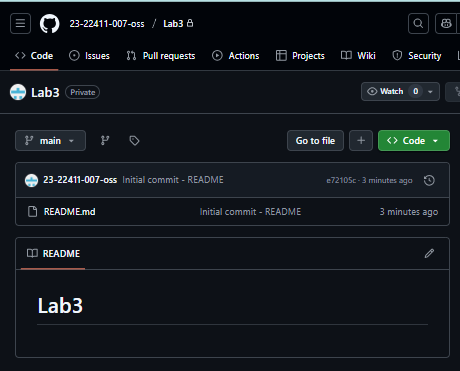


**Task 1 – Handling Local and Remote Commit Conflicts (Pull vs Pull --rebase)**

**Step A** — Prepare initial README





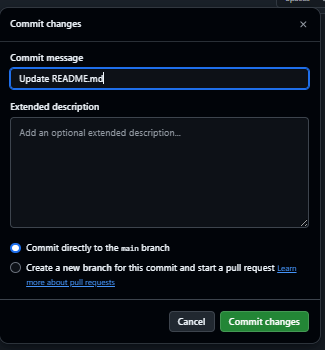


**Step B** **— Make a remote edit**

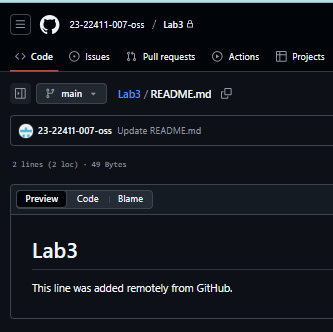
On GitHub, open README.md. Click the pencil. Add a line exactly.



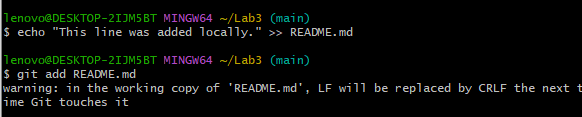
Click on commit changes, the following dialogue box will appear.

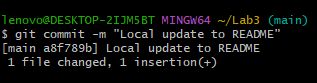


Again click on commit changes, to save the changes made in README.md.

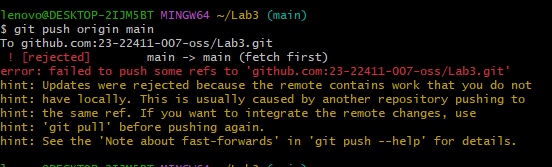


**Step C** — Make a local edit and commit it

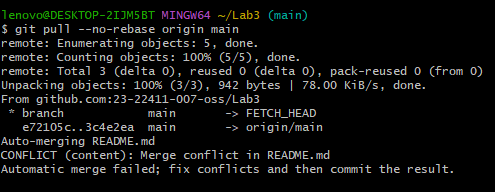




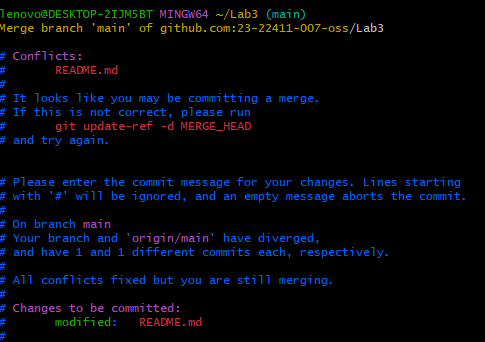
**Step D** — Attempt to push and capture the rejection



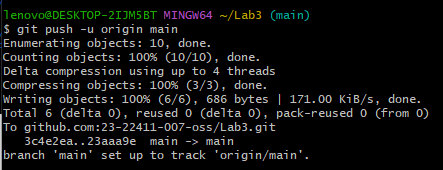
Step E — Pull and merge

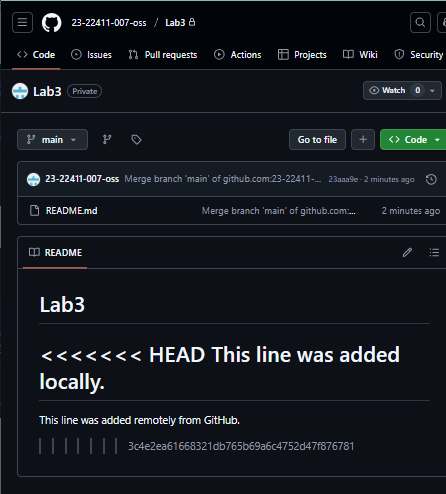


There’s a merge conflict, Git will print conflict messages. Resolve conflicts manually.



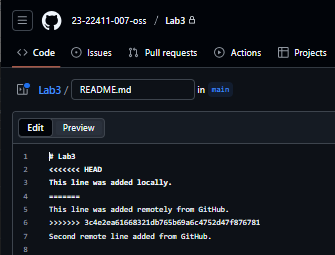
Step F — Push after merge



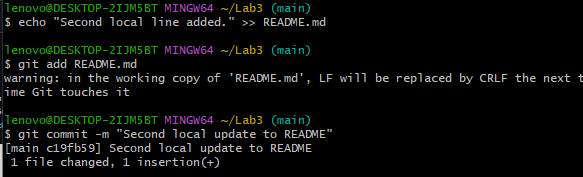


**Repeat but fix using rebase**

Step G — Make another remote edit

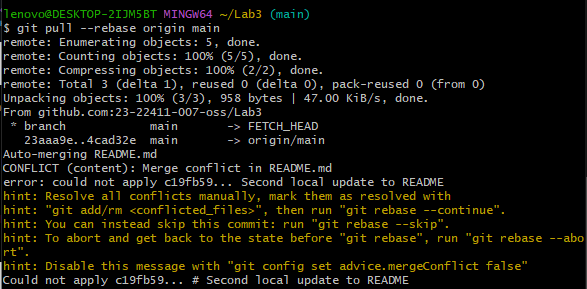


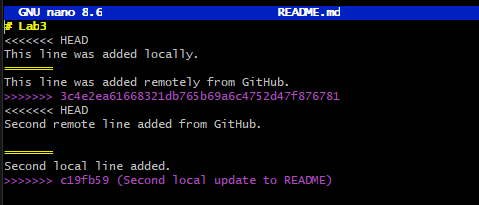
Step H — Make another local change and commit

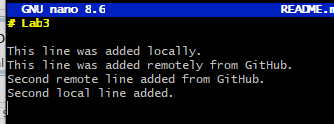


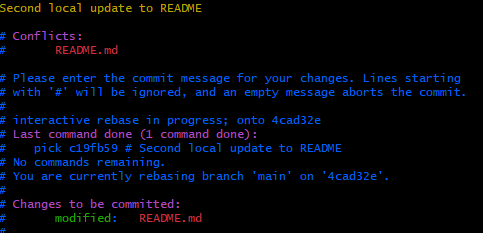
Step I — Pull with rebase

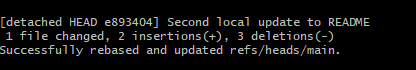
Merge conflict arise:



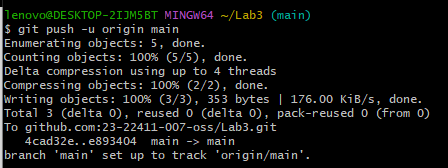


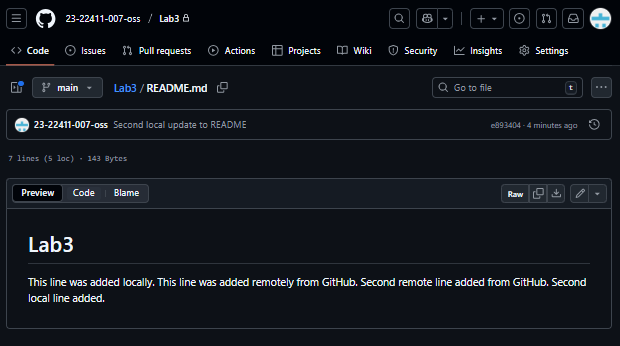






Step J — Push after rebase

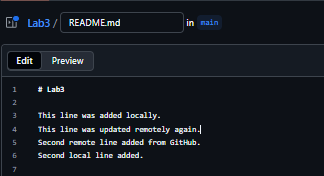


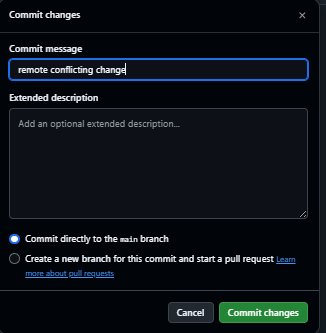


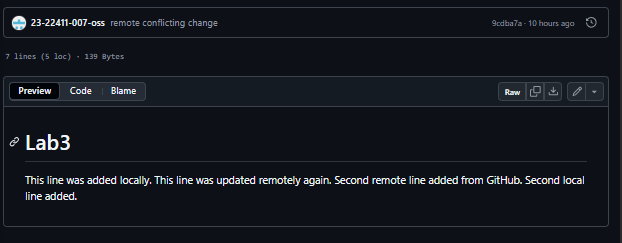
**Task 2 – Creating and Resolving Merge Conflicts Manually**

Step A — Make a **remote conflicting change** on GitHub

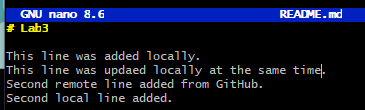
* Open README.md on your GitHub Lab3 repo. Find the line you edited before (e.g., This line was added remotely from GitHub.). Change it to something else.



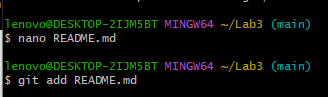


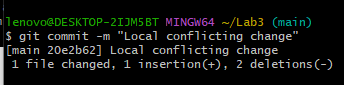


Step B — Make a **local conflicting change**

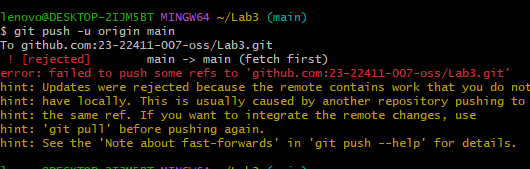


Step C — Stage and commit the local change

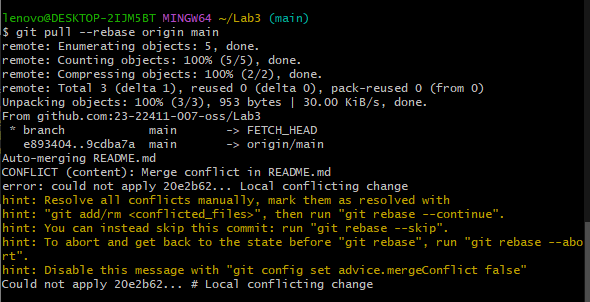




Step D — Try to push local commit

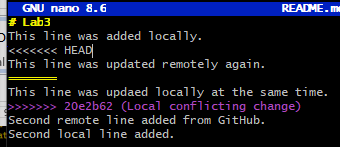


Step E — Pull with rebase to bring in remote changes

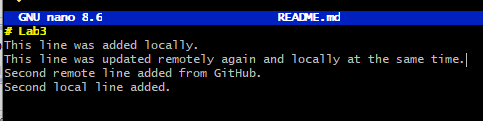


Step F — Resolve the conflict manually

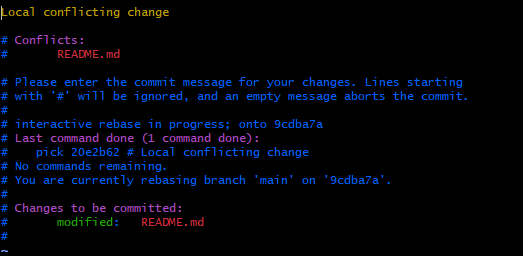
With conflict:



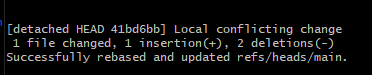
After resolving conflict:



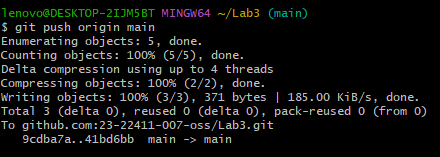
As shown below, merge completed.

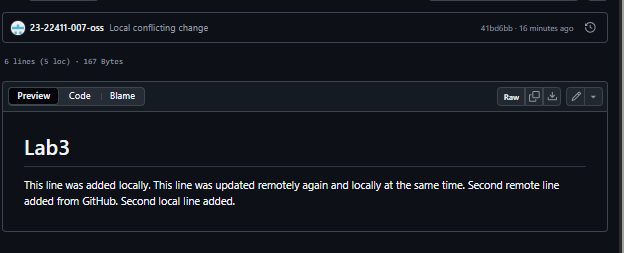


Step G — Mark the conflict as resolved and continue rebase



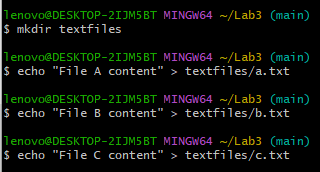
Step H — Push your resolved changes



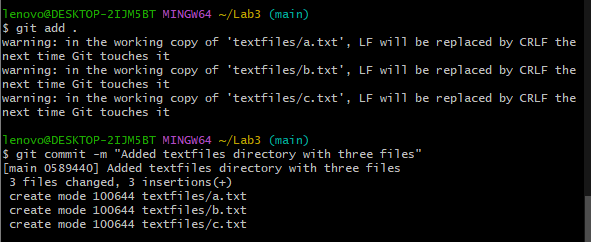


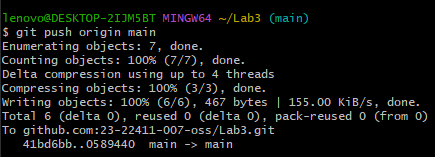
**Task 3 – Managing Ignored Files with .gitignore and Removing Tracked Files**

### Step A — Create a folder and files



Step B — Add and commit the new directory

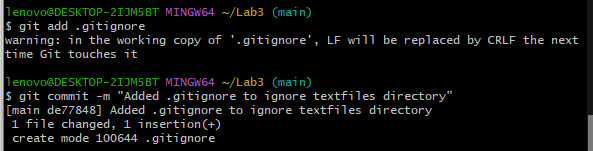


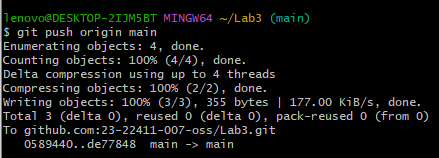


Step C — Create a .gitignore file

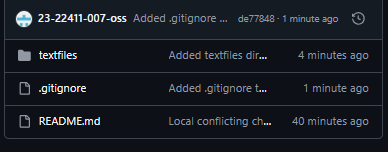


Step D — Add and commit .gitignore

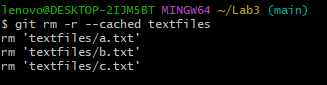




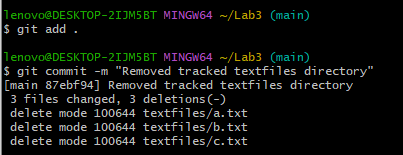
Step E — Check GitHub

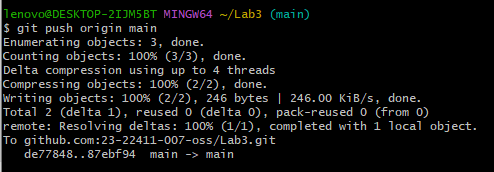


Step F — Remove tracked files (but keep them locally)

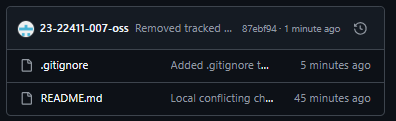


Step G — Commit and push the removal





Step H — Verify on GitHub



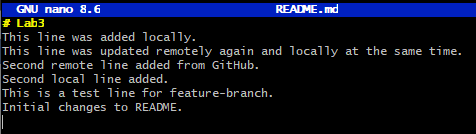
**Task 4 – Create Temporary Changes and Use git stash**

Step A — Create a feature branch

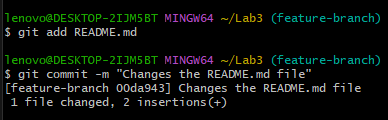


Step B — Make an initial change and commit

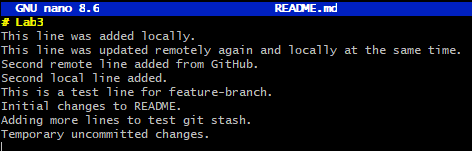
1. Open a file, for example README.md, and add a few test lines:



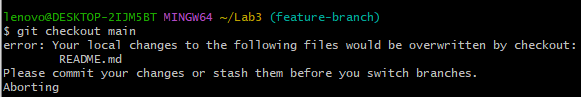
1. Stage and commit the changes:



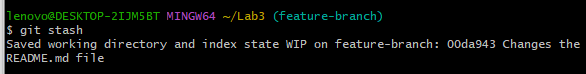
Step C — Make additional changes **without committing**



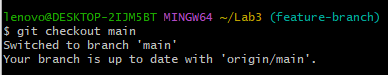
Step D — Try to switch back to main branch



Step E — Stash your changes



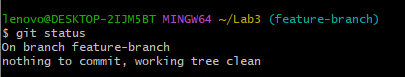
Step F — Switch back to main branch



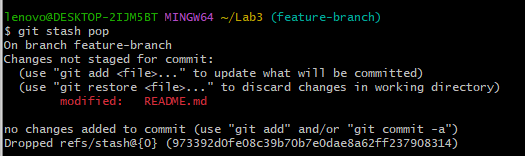
Step G — Switch back to feature branch



Step H — Check working directory status

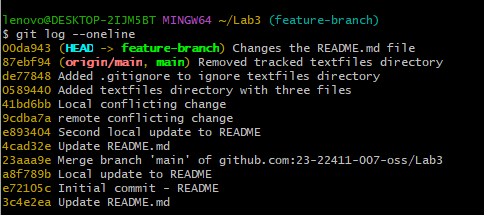


Step I — Restore your stashed changes

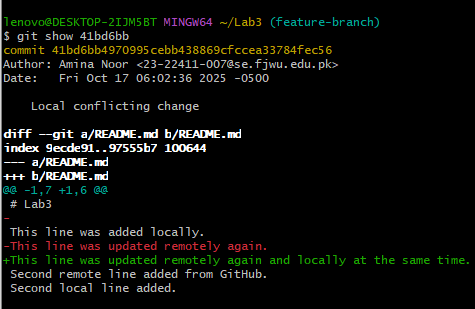


**Task 5 – Checkout a Specific Commit Using git log**

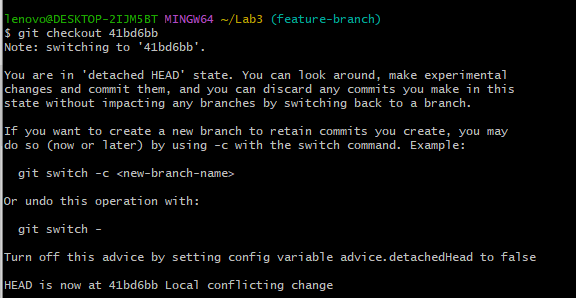
Step A — View commit history



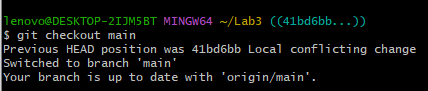
Step B — Copy a previous commit hash



Step C — Checkout that commit



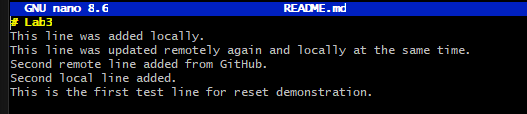
Step D — Return to the main branch



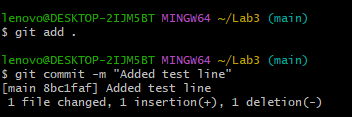
**Task 6 – Resetting Commits (Soft vs Hard Reset) (With Verification Steps)**

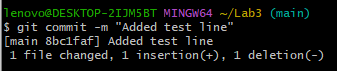
Step A — Add a new line and commit it

1. Edit any file (e.g., README.md) and add a new line:



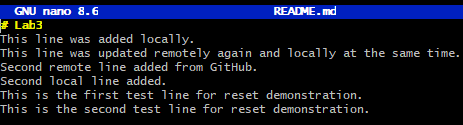
1. Stage and commit:



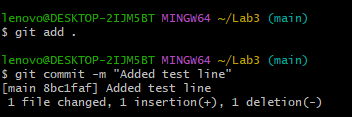


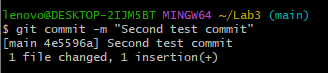
Step B — Add another change and commit again

1. Edit the file again, add another line:

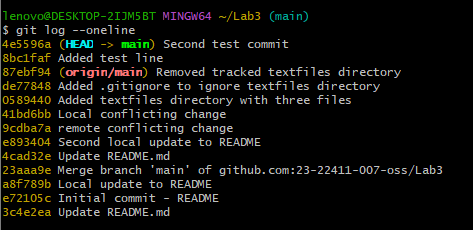


1. Stage and commit:

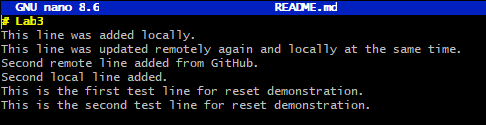




Step C — View commit history before reset



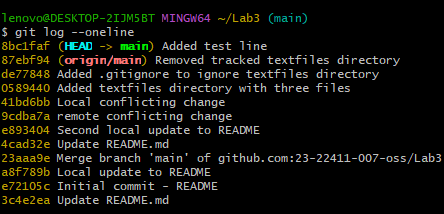
Step D — Verify file contents before reset



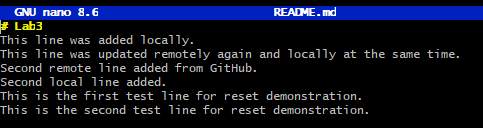
Step E — Perform a soft reset



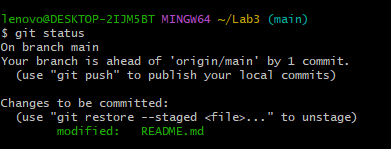
Step F — Check commit history after soft reset



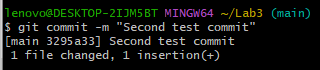
Step G — Verify changes in file after soft reset



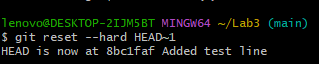
Step H — Check git status



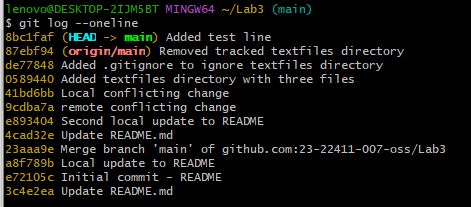
Step I — Recommit the second change



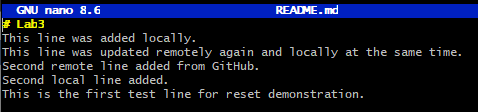
Step J — Perform a hard reset



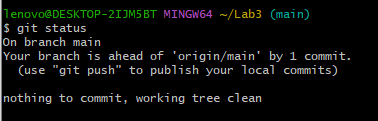
Step K — Check commit history after hard reset



Step L — Verify file contents after hard reset

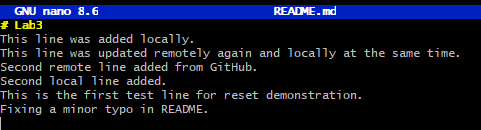


Step M — Check git status

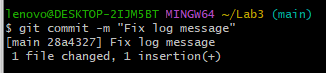


**Task 7: Amending the Last Commit**

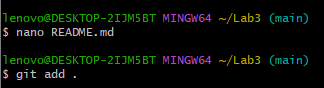
Step A — Make a small change and commit

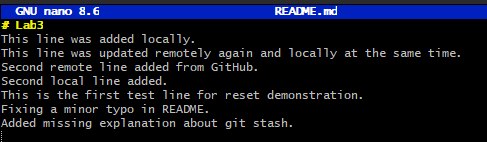


Step B: Stage and commit it:

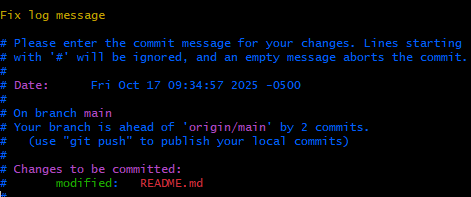


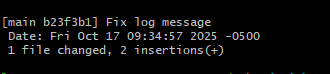
Step C — Make another change to amend





Step D— Amend the last commit





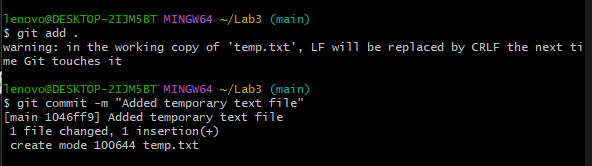
**Task 8 – Reverting a Commit (Safe Undo on Remote Branch)**

**Step A — Make a change and commit it**.

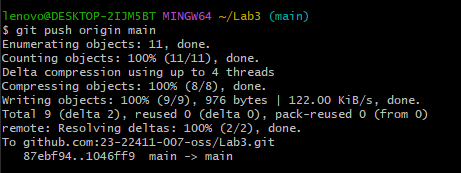
1. Add a temporary file:



1. Stage and commit:

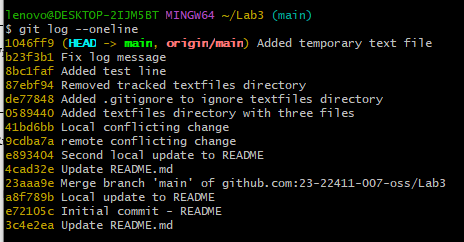


1. Push to GitHub:



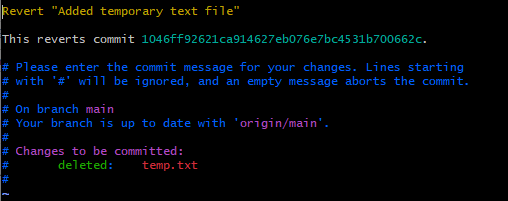
### Step B — Identify the commit to revert

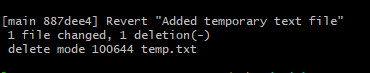
1. View recent commits:

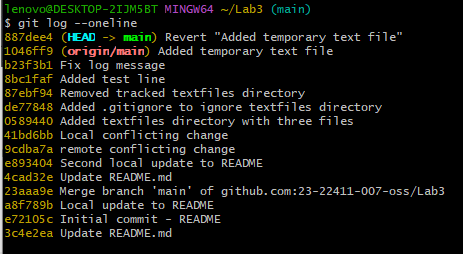


### Step C — Revert the commit

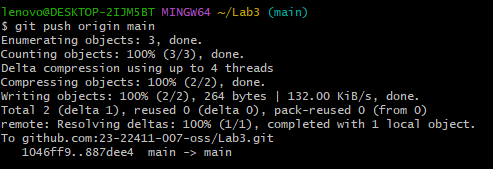
1. Run revert:







**Step D — Push the revert commit**



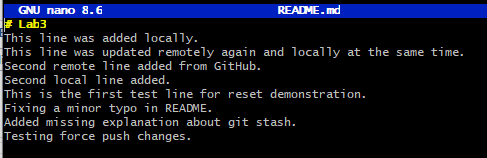
**Task 9 – Force Push (With Caution)**

Step A — Create a new branch



Step B — Make and commit a small change

1. Edit a file (e.g., README.md) and add:

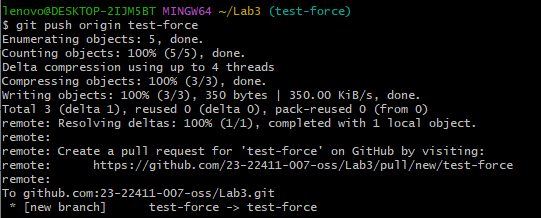


1. Stage and commit:

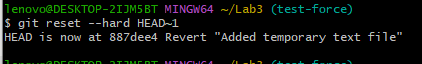




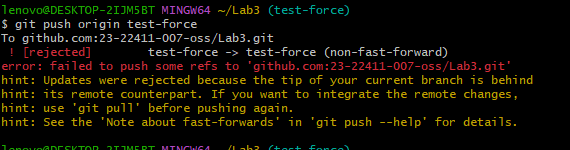
Step C — Push the new branch



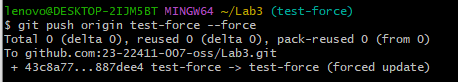
Step D — Perform a hard reset



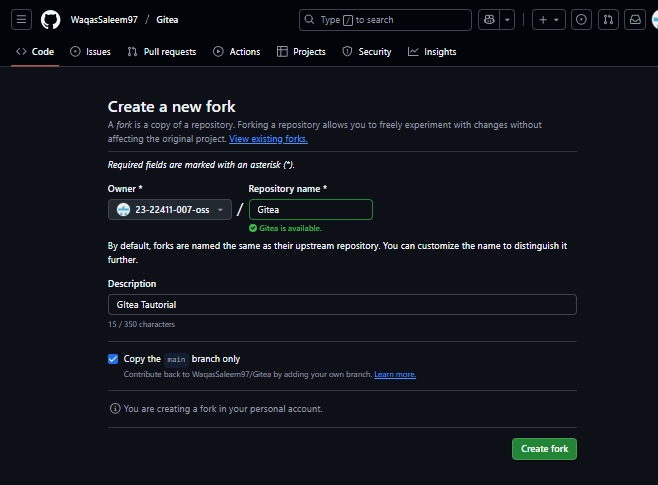
Step E — Try to push normally

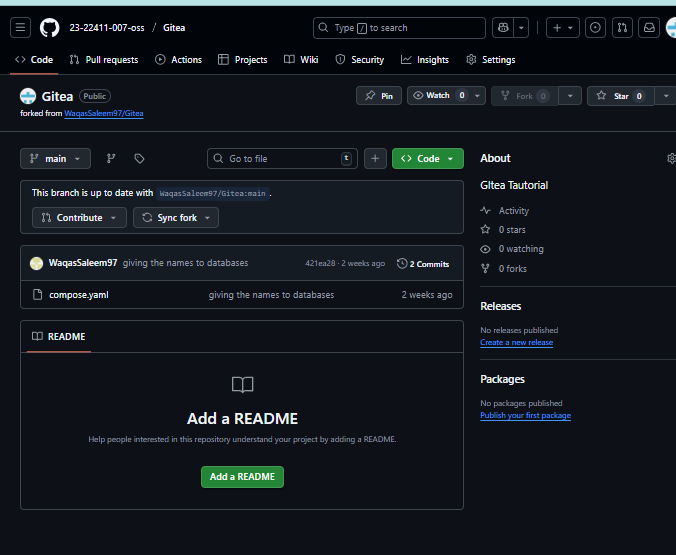


Step F — Force push (overwrites remote history)

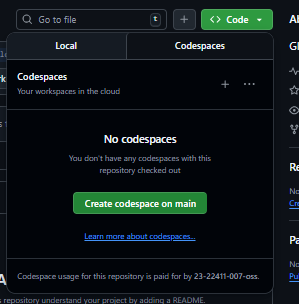


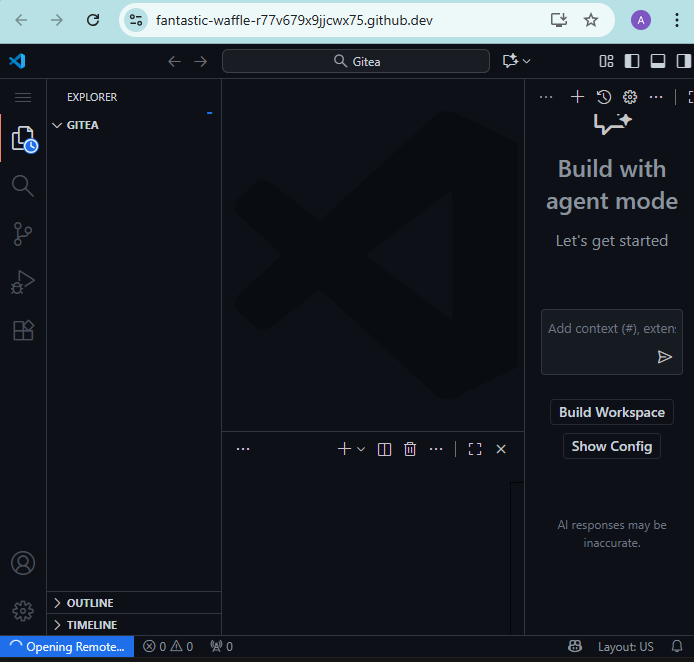
**Task 10 – Running Gitea in GitHub Codespaces via Docker Compose**

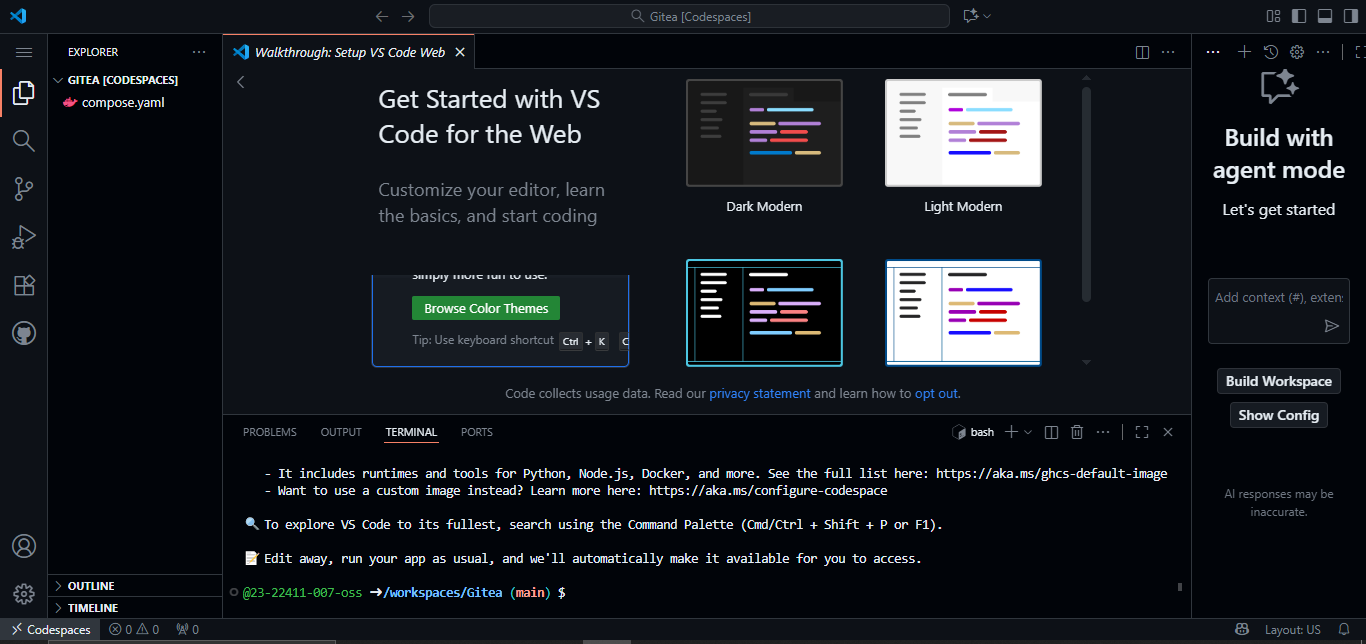
Step A — Fork the Gitea Repository  




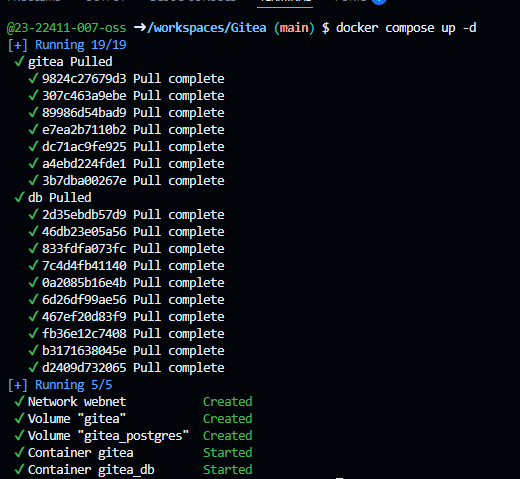
Step B — Open Forked Repo in Codespaces



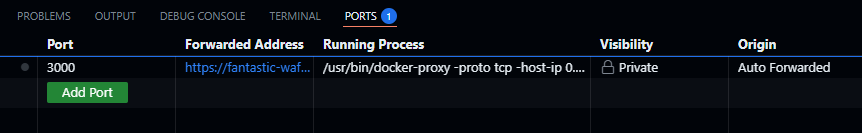




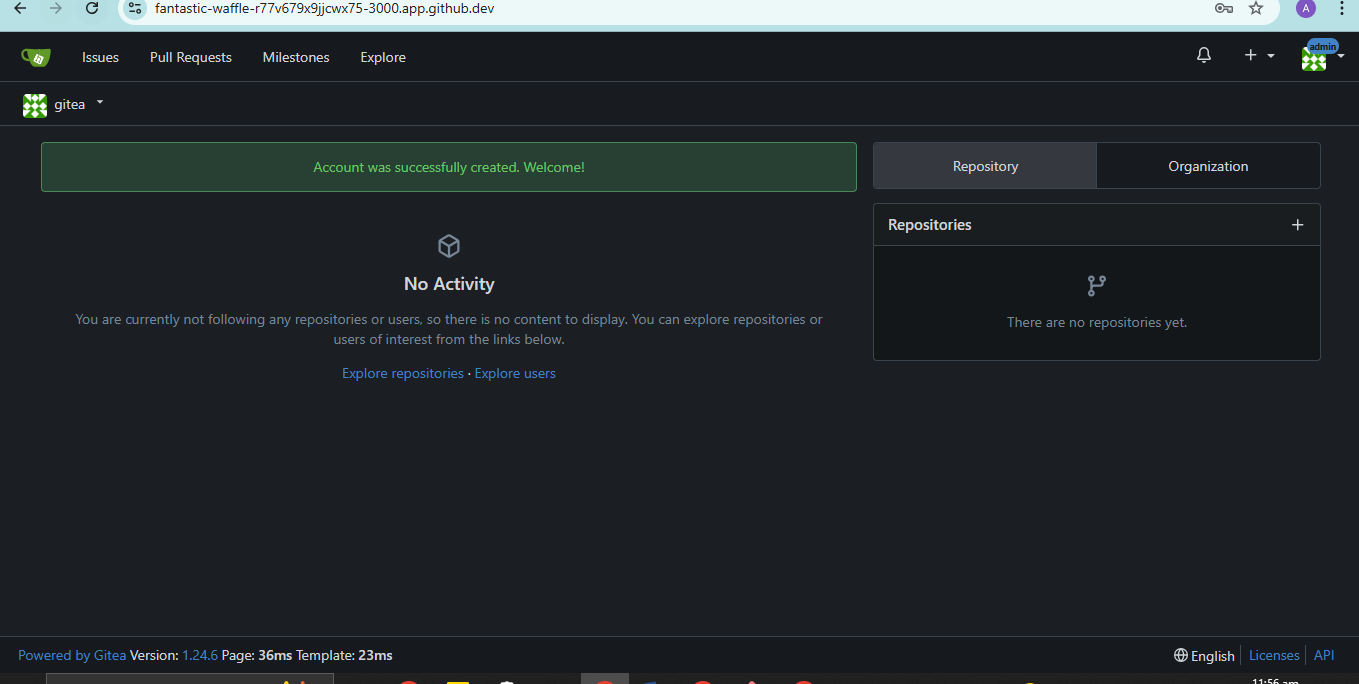
Step C — Start Gitea with Docker Compose



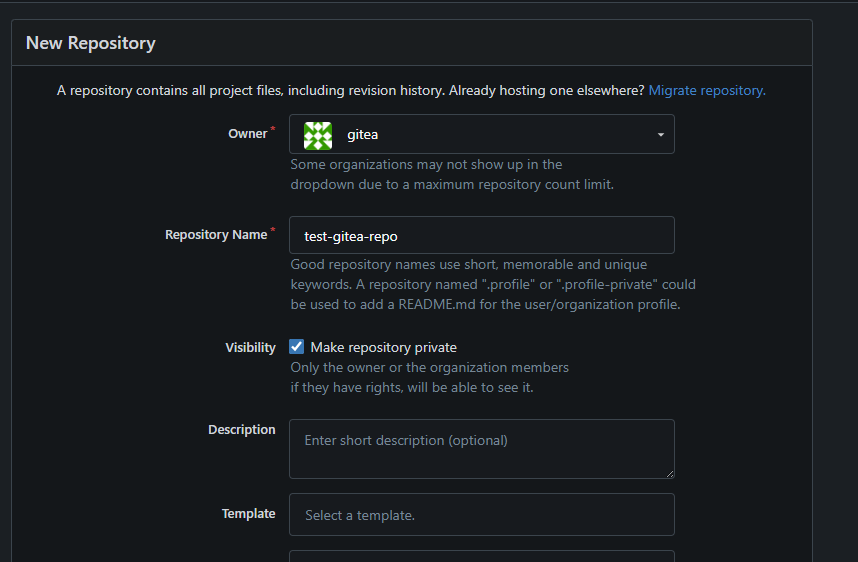
Step D — Access Gitea Web Interface

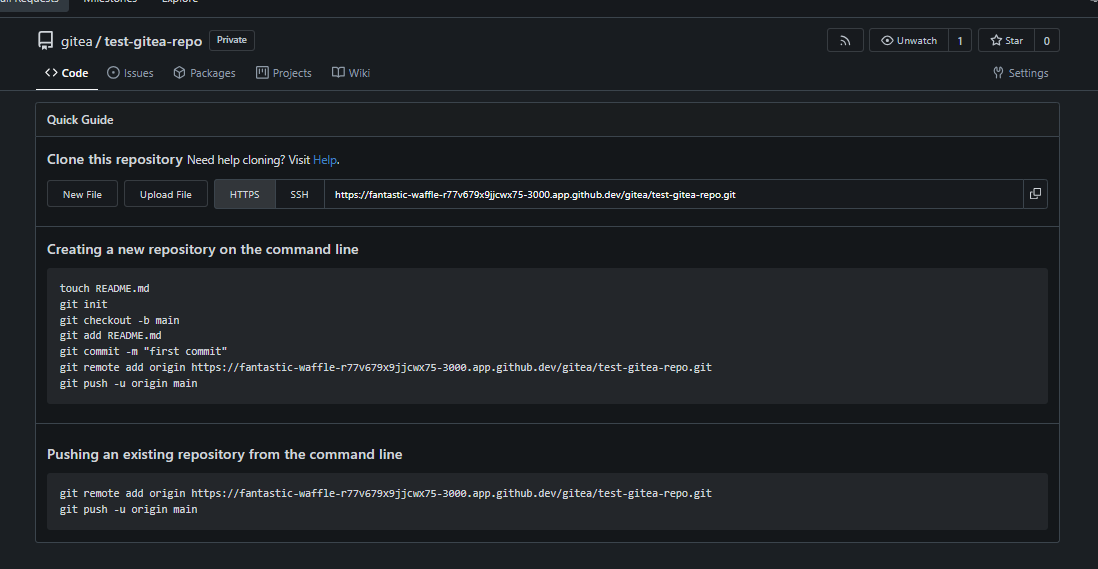


Step F — Log In to Gitea



Step G — Create a New Repository in Gitea

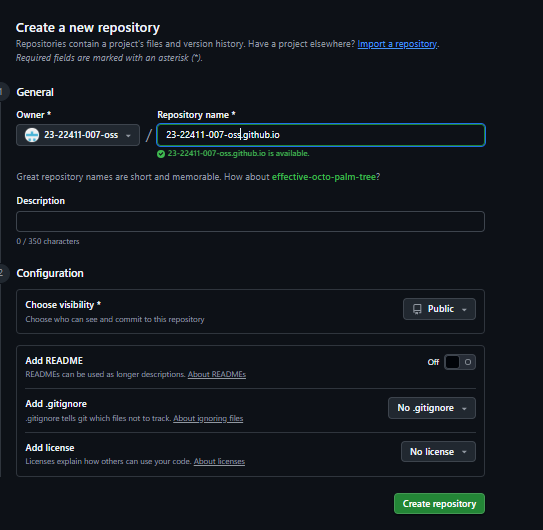


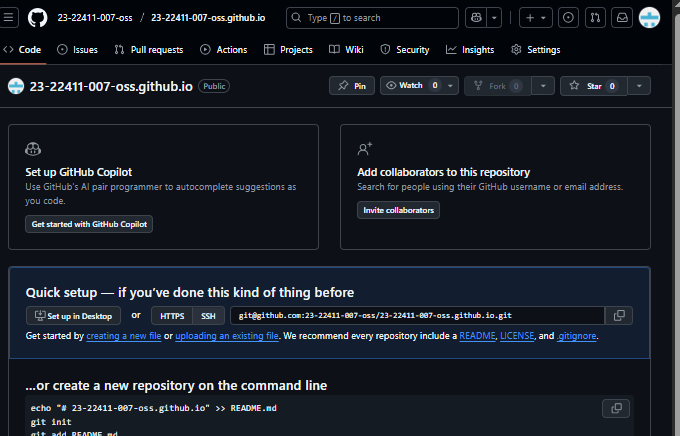


**Task 11 — Creating a GitHub Pages Portfolio Site**

Step A — Create GitHub Pages Repository

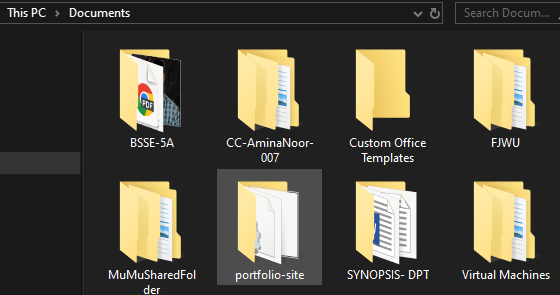
1. Go to GitHub → click **New Repository**.
2. Name it.



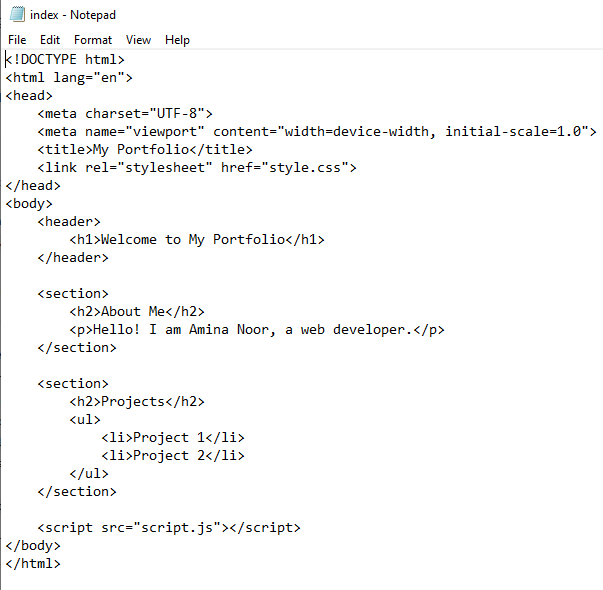


Step B — Add Static Website Code

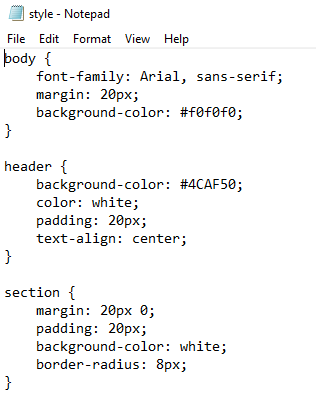
## Create a project folder



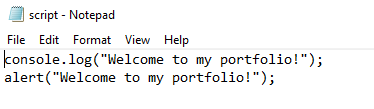
1. Create the HTML file (index.html)



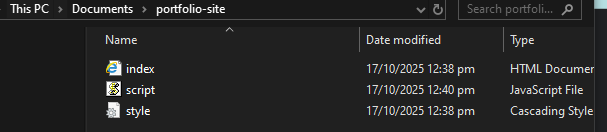
1. Create the CSS file (style.css)



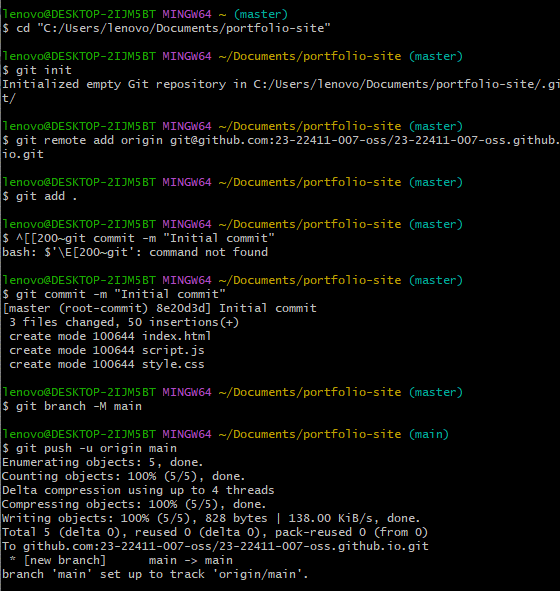
1. Create the JavaScript file (script.js)

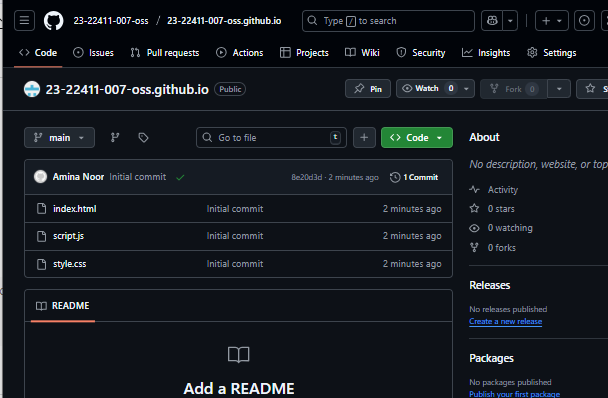


All the files are shown below,

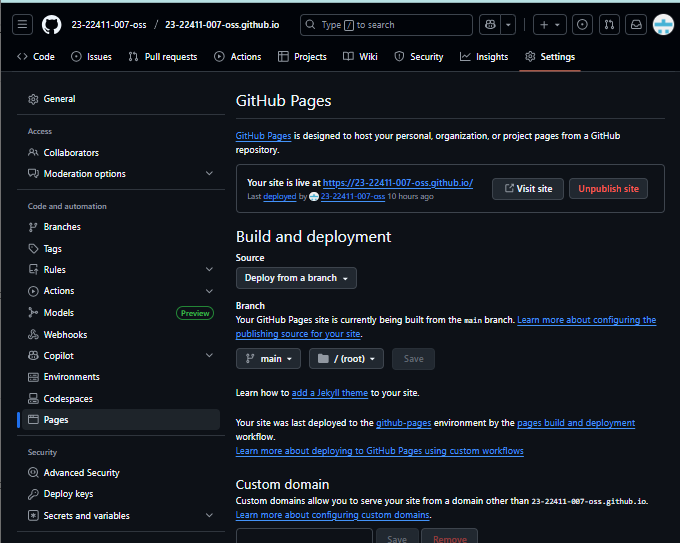


Step C — Push the Files to GitHub

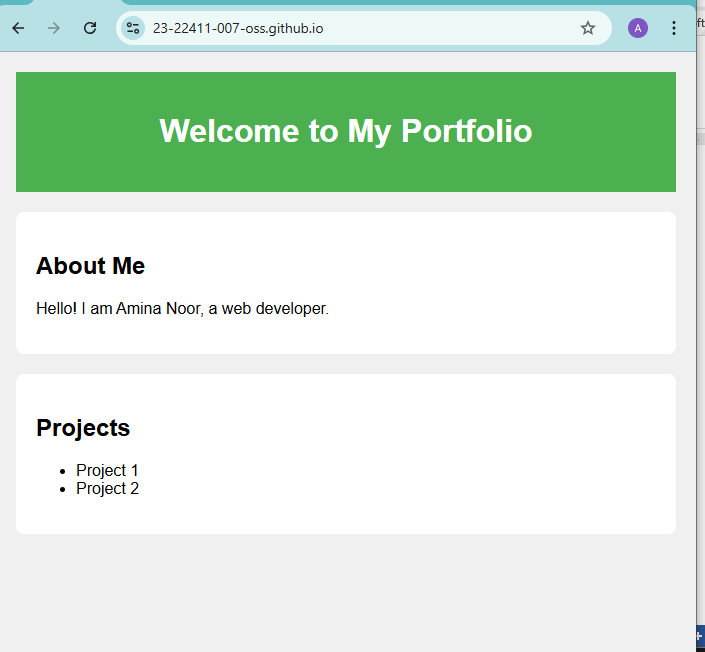




Step D — Check GitHub Pages Settings



Step E — Visit Your Live Site



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**