Fatima Jinnah Women University

*Department of Software Engineering*



**SUBJECT: CLOUD COMPUTING**

**SUBMITTED TO: SIR WAQAS**

**SUBMITTED BY: ZUNAIRA NOOR**

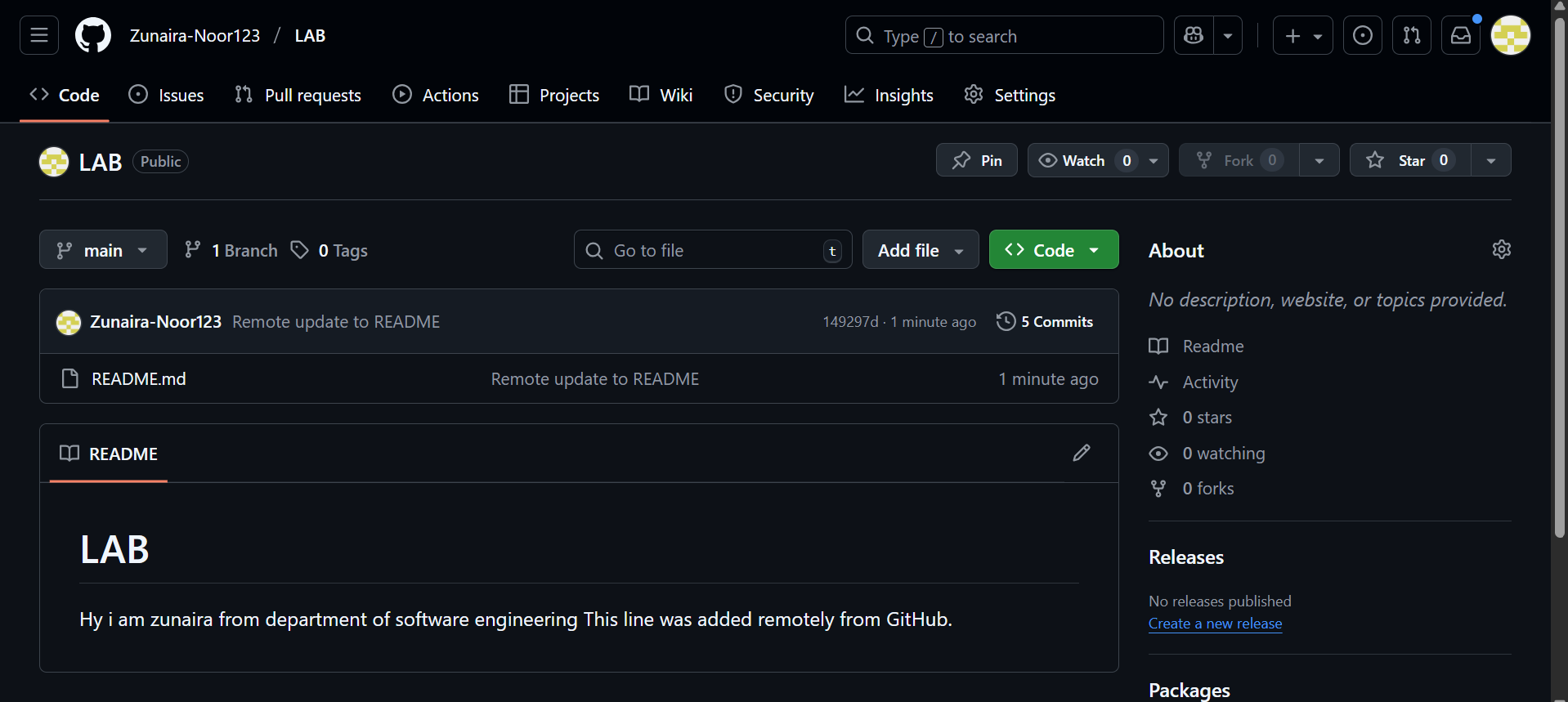
**REGISTRATION NO: 2023-BSE-075**

**CLASS: BSSE IV-B**

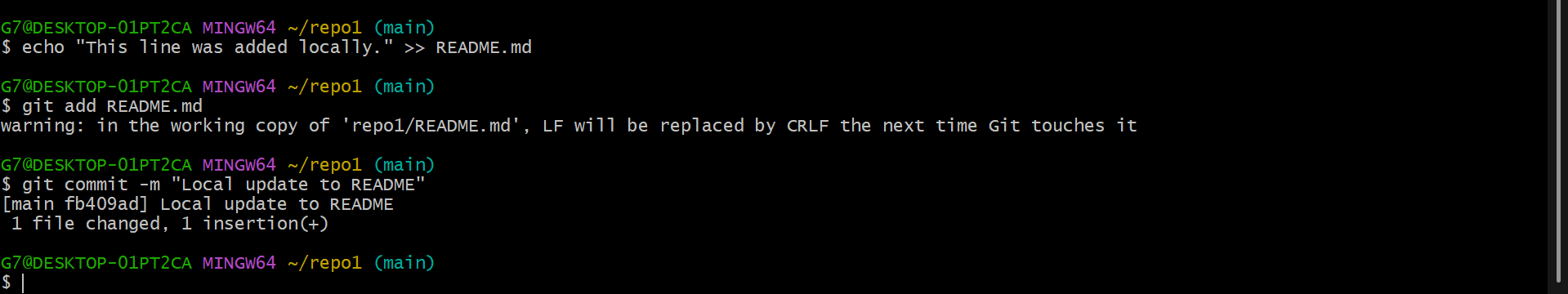
***LAB NO 3***

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

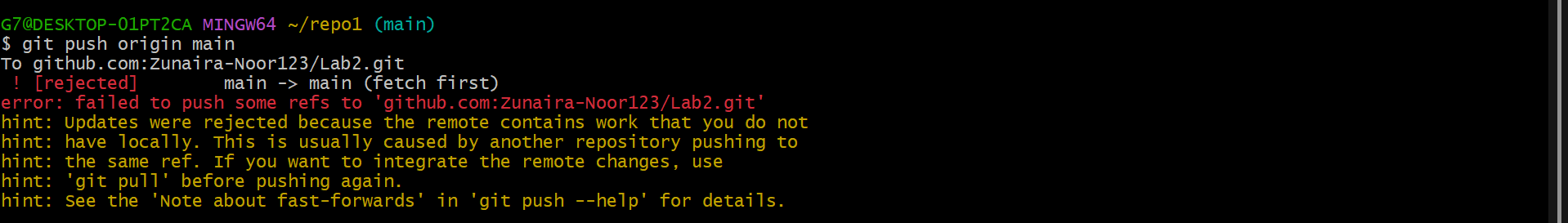
Remote update to README



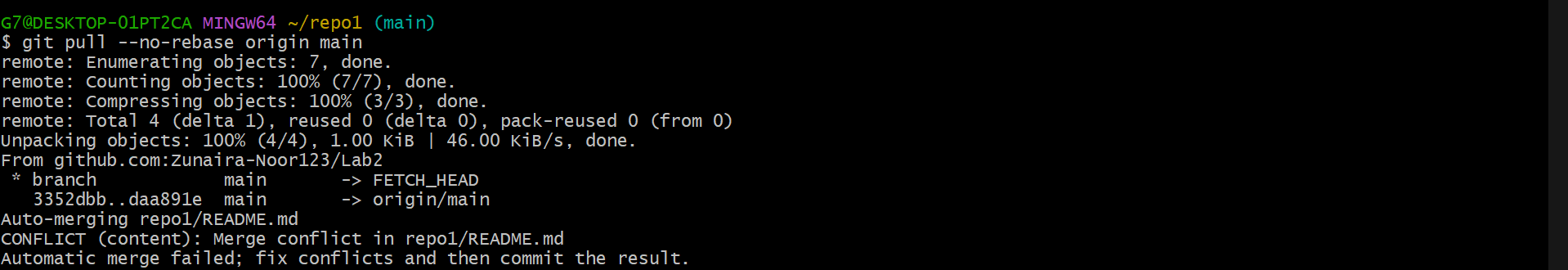
Local update to README



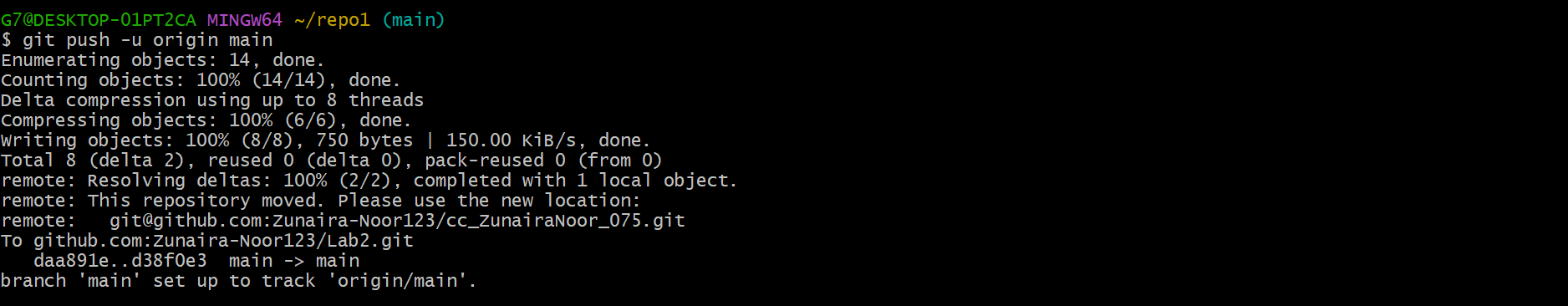
Push Attempt



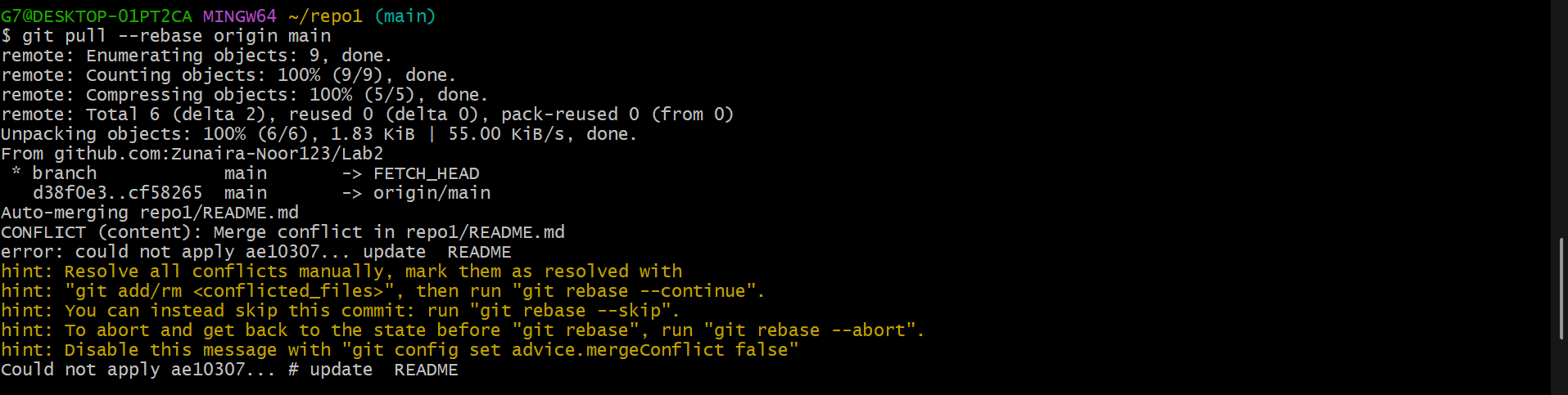
Pull with Merge



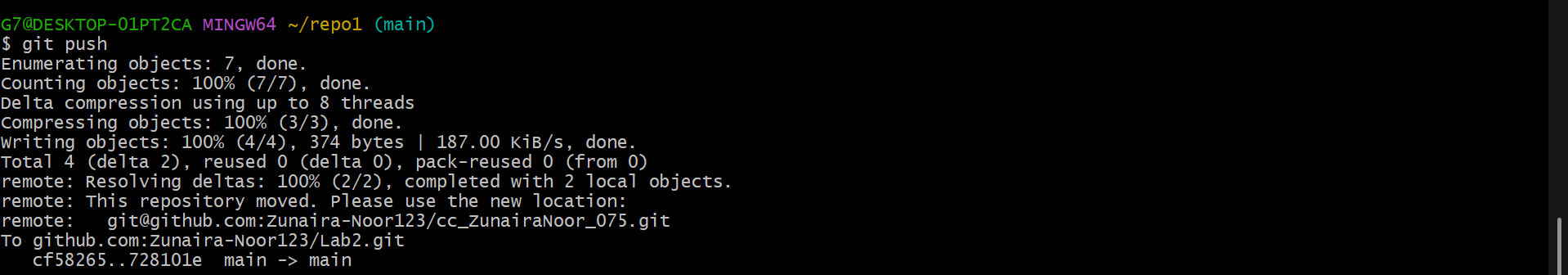
Push After Merge



**Pull Using Rebase**

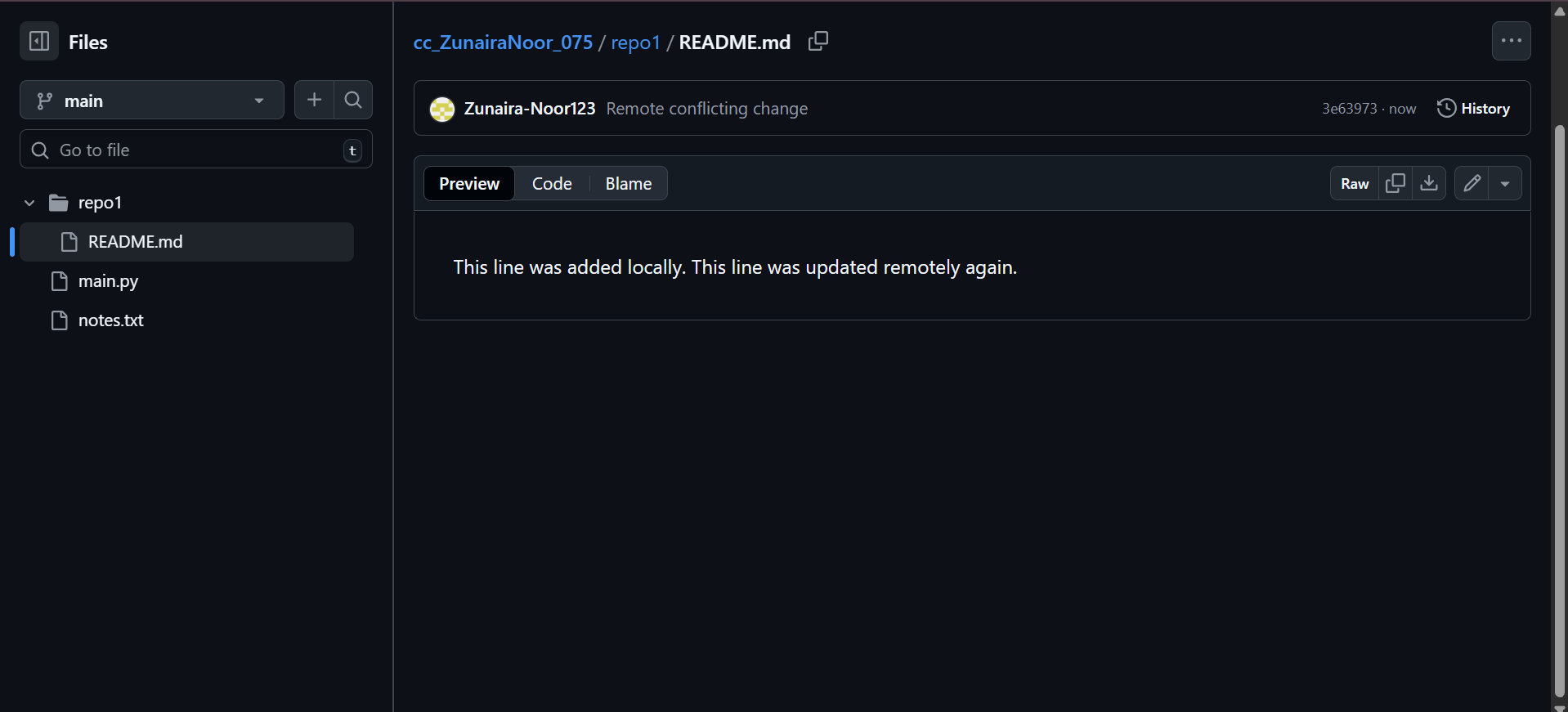


Push After Rebase

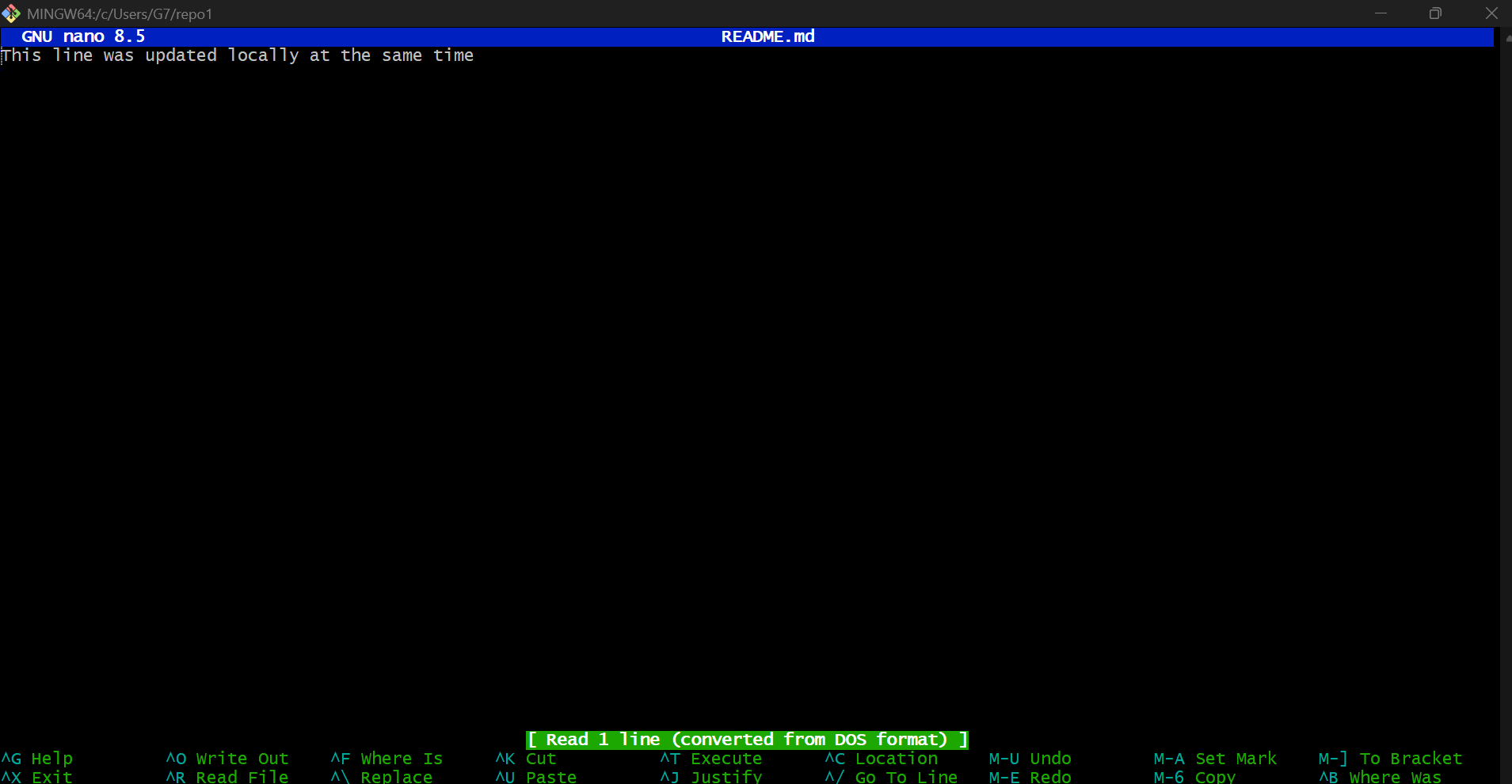


**Task 2 – Creating and Resolving Merge Conflicts Manually**

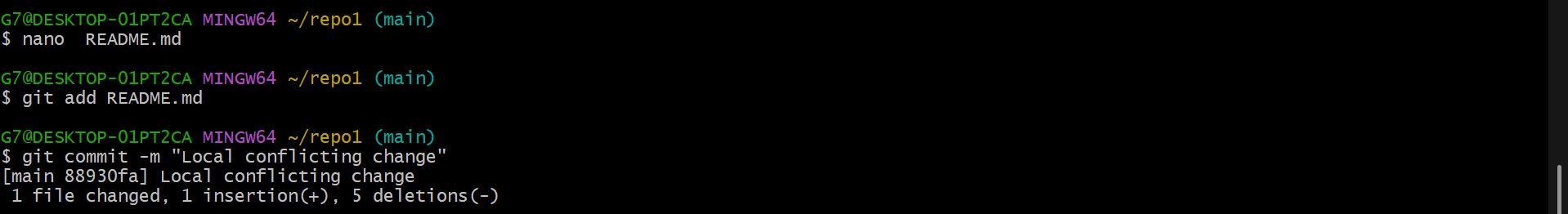
On **GitHub (remote)**, open your README.md file and change an existing line.



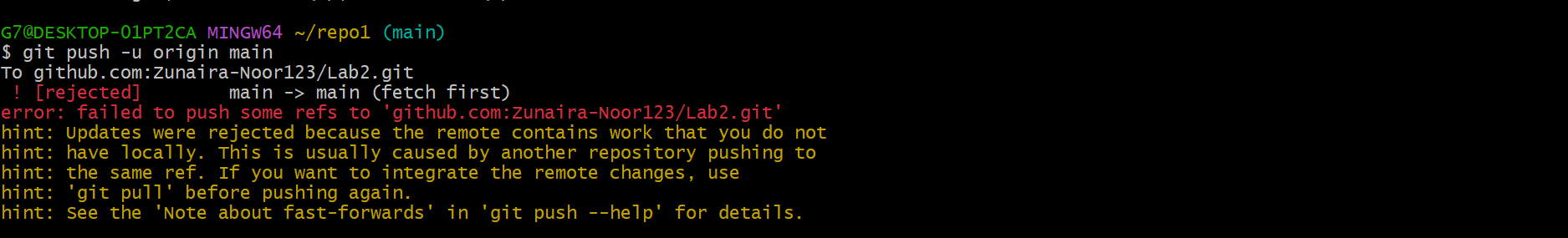
On your **local machine**, edit the **same line** in the same file but make a **different change**:



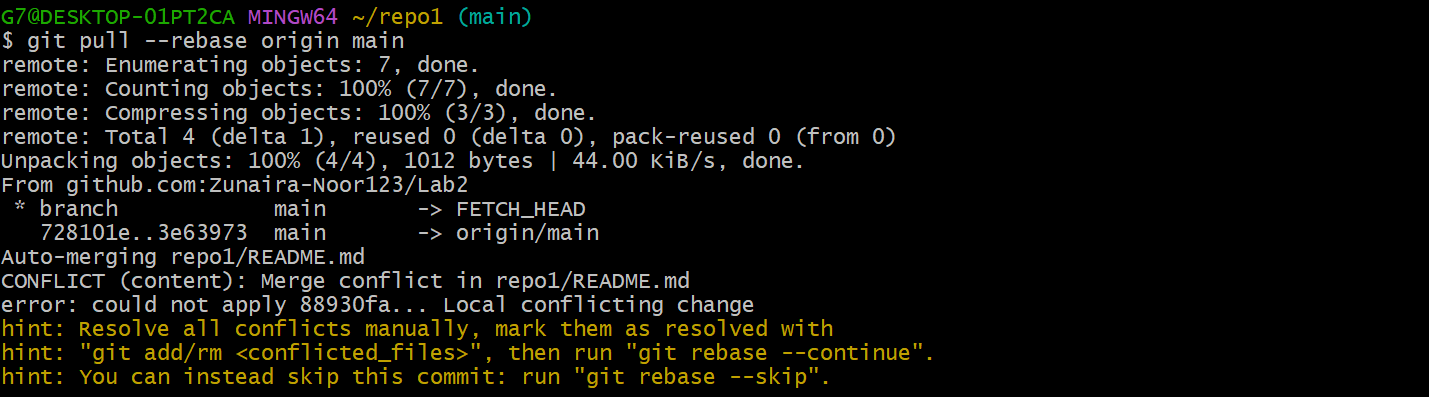
Stage and commit your local change:



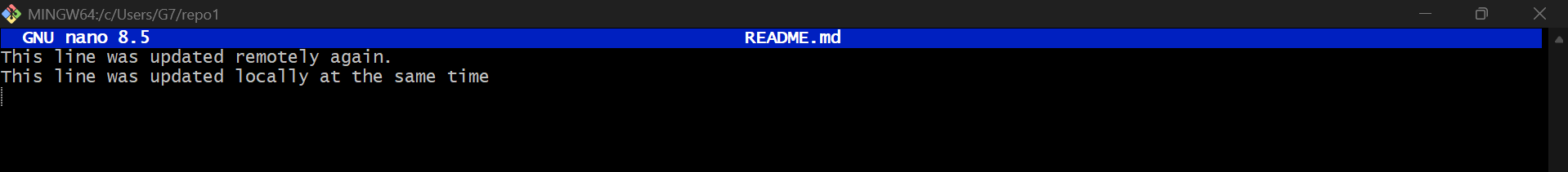
Try to push:



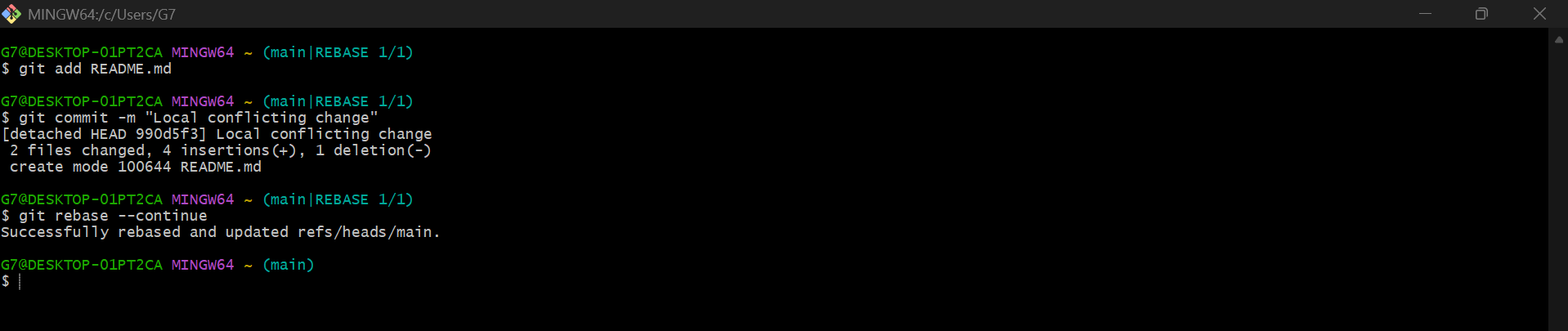
Pull with rebase to bring in remote changes:



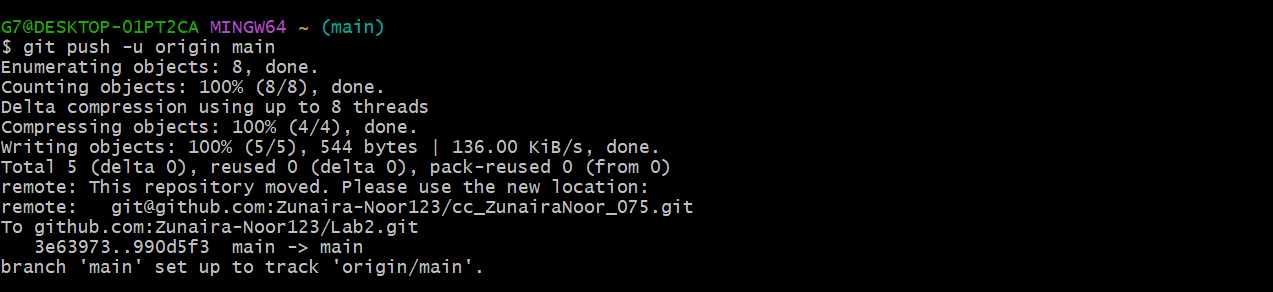
Open the README.md file in your editor — you’ll see conflict markers like this:



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



Finally, push your changes:



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

**Create a new folder named textfiles inside your repository:**

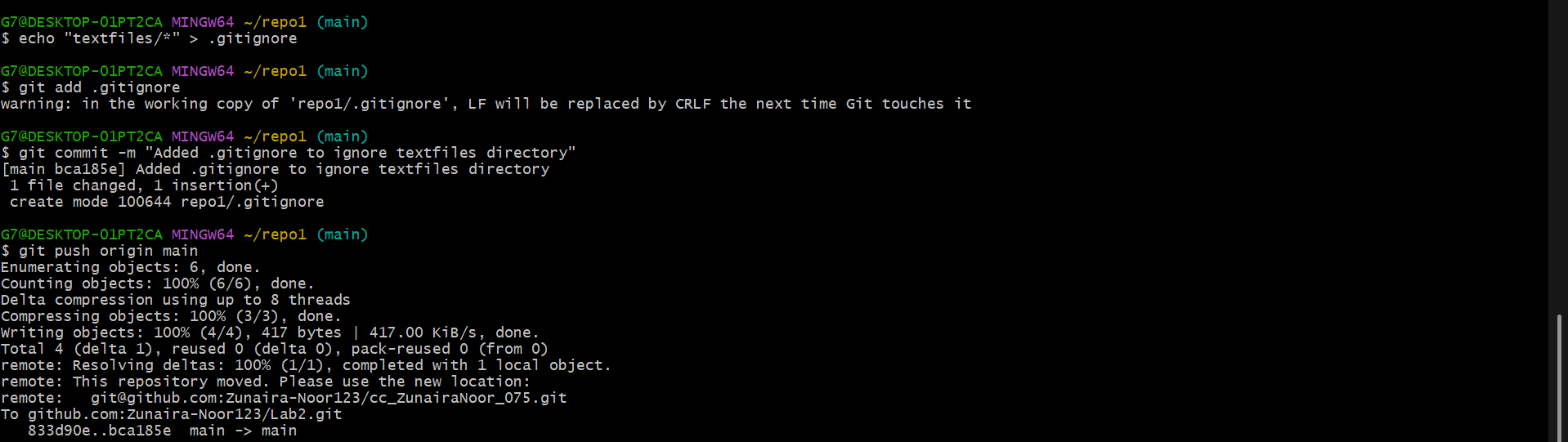
Inside the textfiles folder, create three text files:

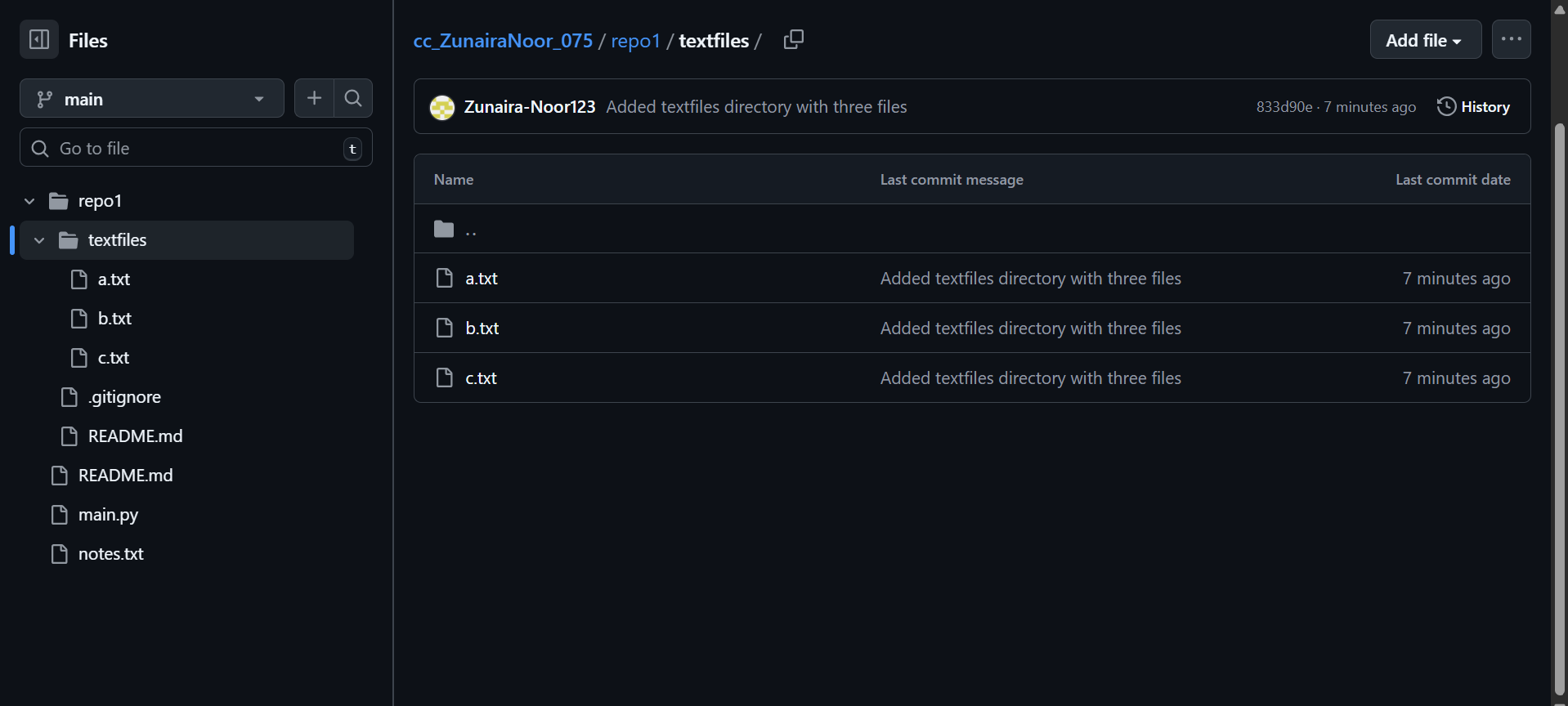
Add and commit the new directory:



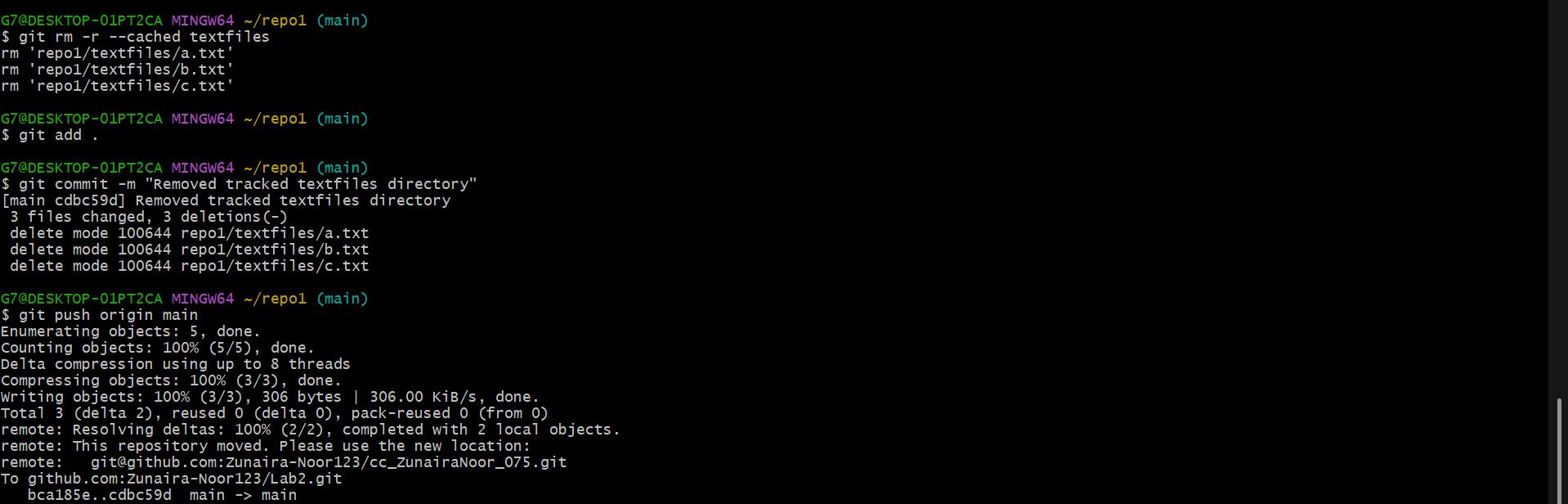
Now, create a .gitignore file in the root of your repository:

Add and commit the .gitignore file:

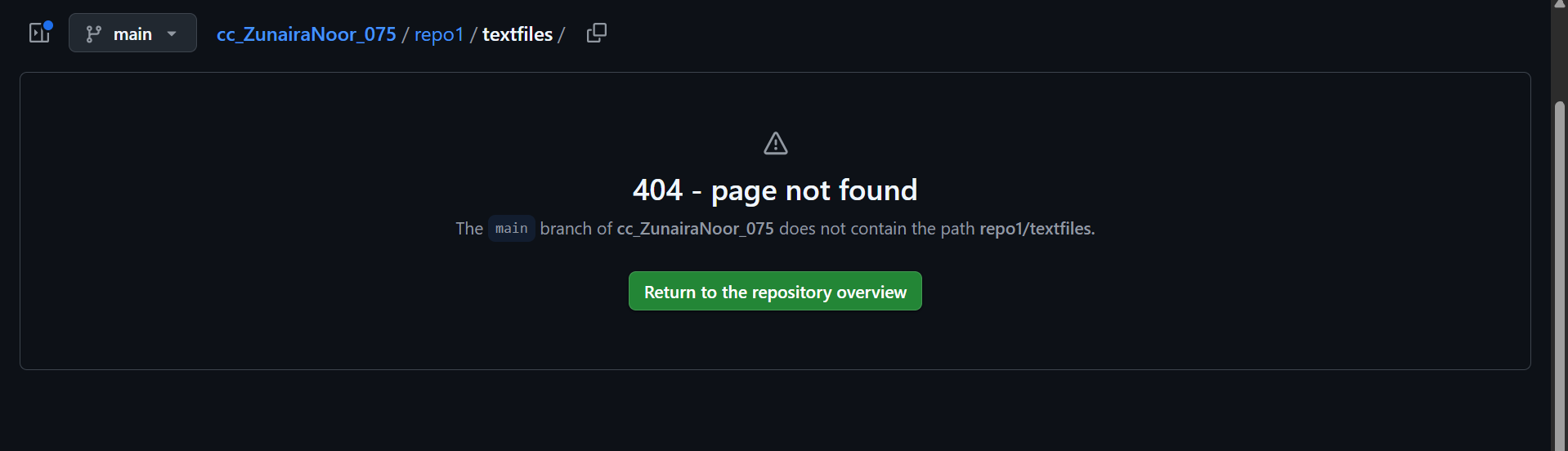


Go to **GitHub** and check your repository — notice that the textfiles directory is **still visible** on the remote.

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

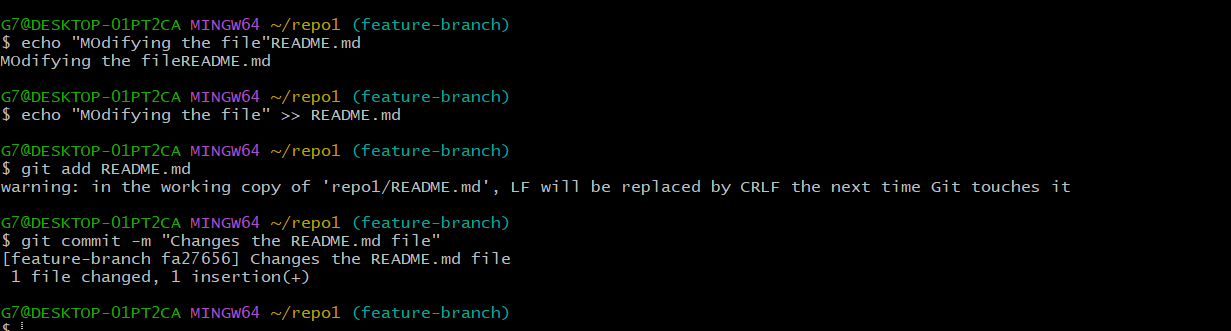


Check your GitHub repository again — the textfiles folder should now be **deleted remotely**.



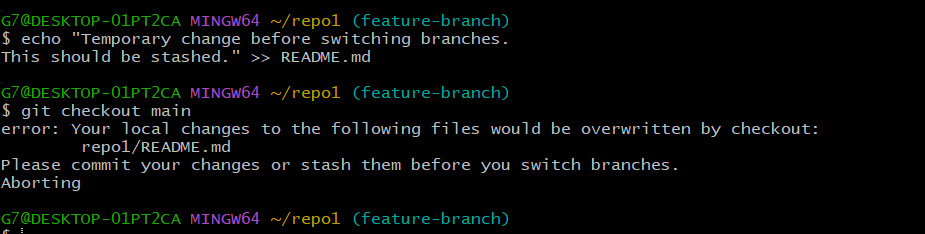
**Task 4 – Create Temporary Changes and Use git stash**

Modify any file (for example, README.md) by adding a few test lines.

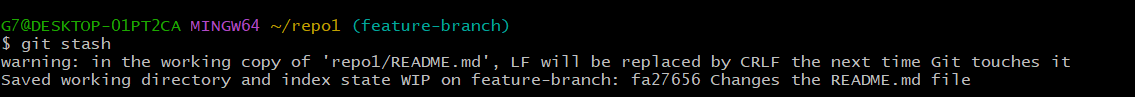


Without committing or stashing the changes, try to switch to another branch:

error message



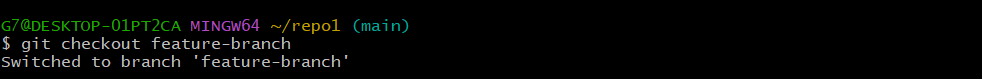
To fix this, **stash your changes**:



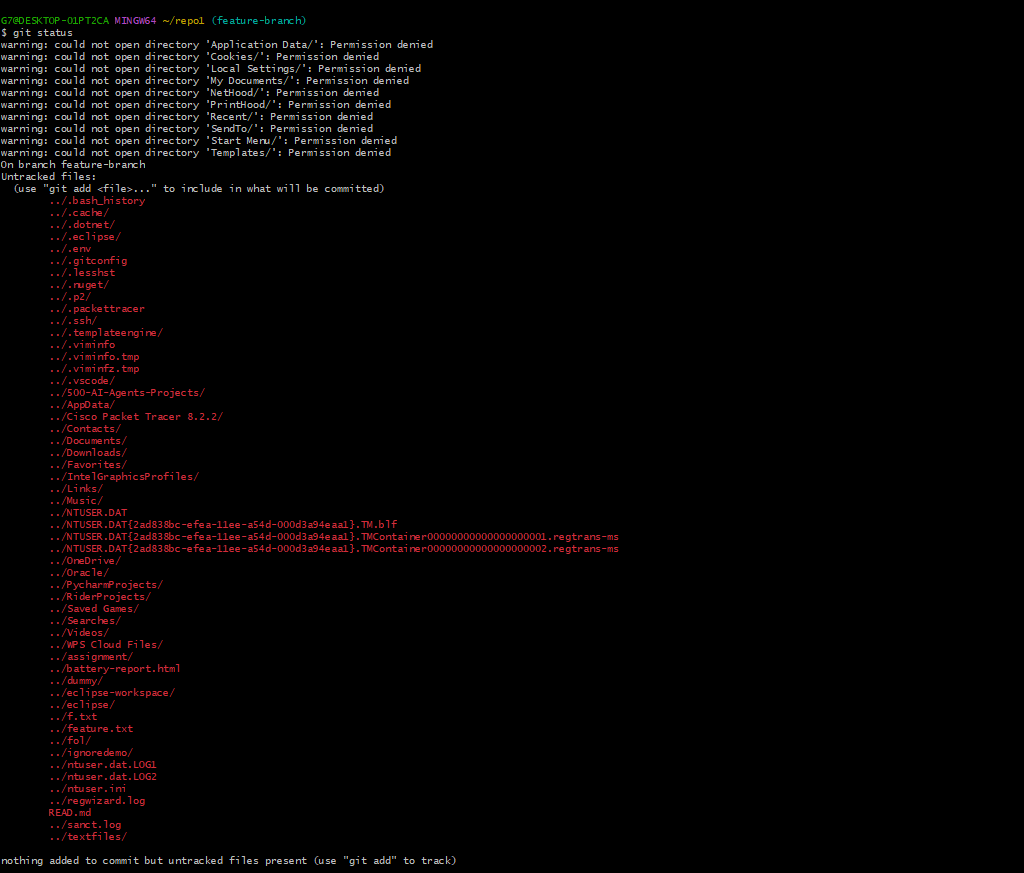
Try switching branches again — it should now work:



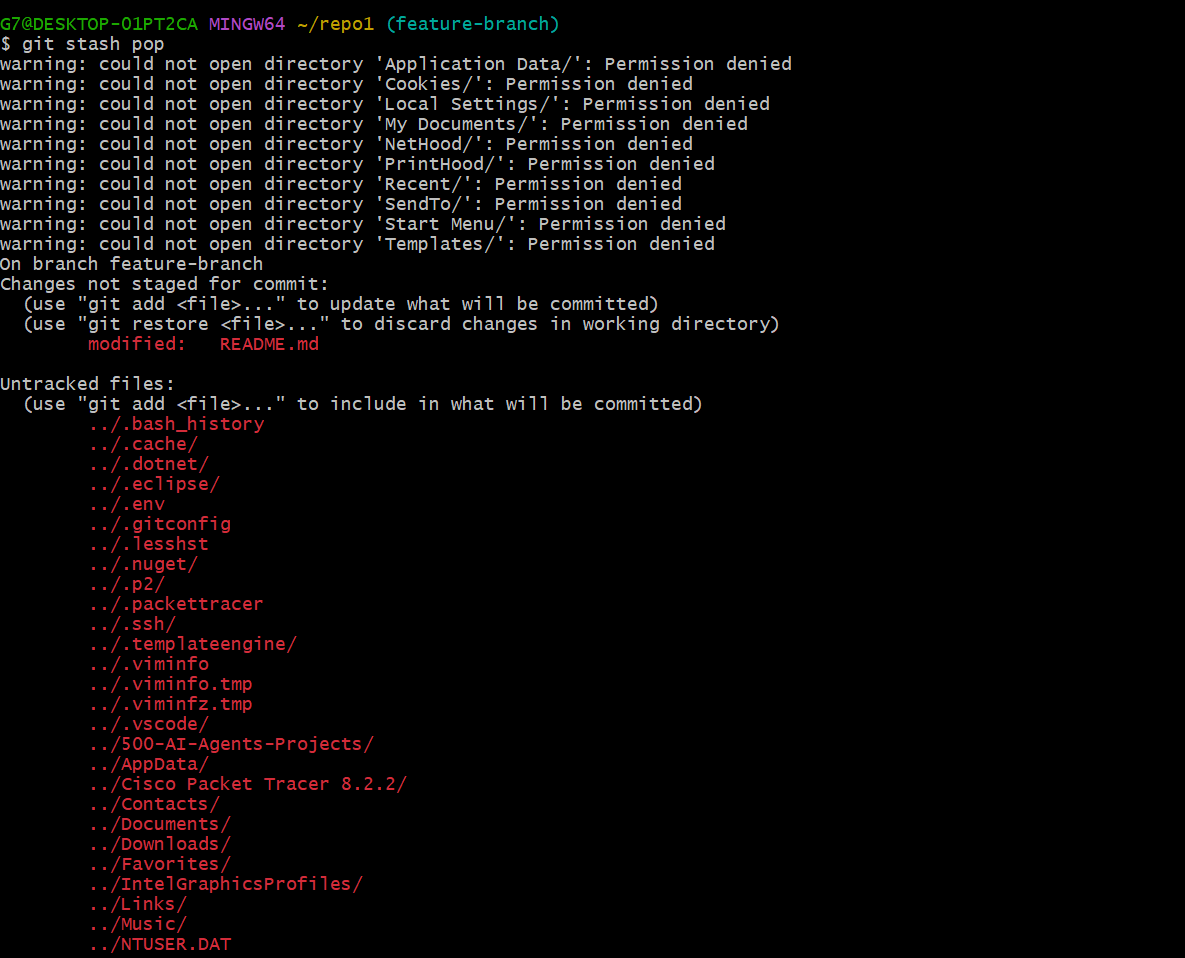
Return to your previous branch (for example, feature-branch):



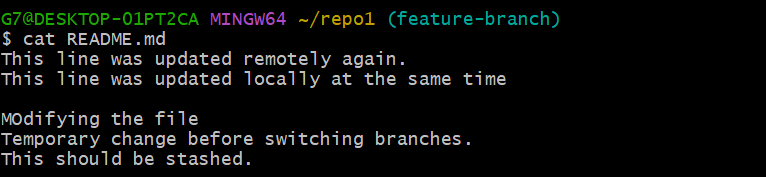
Check your working directory status:



Now restore your stashed changes:

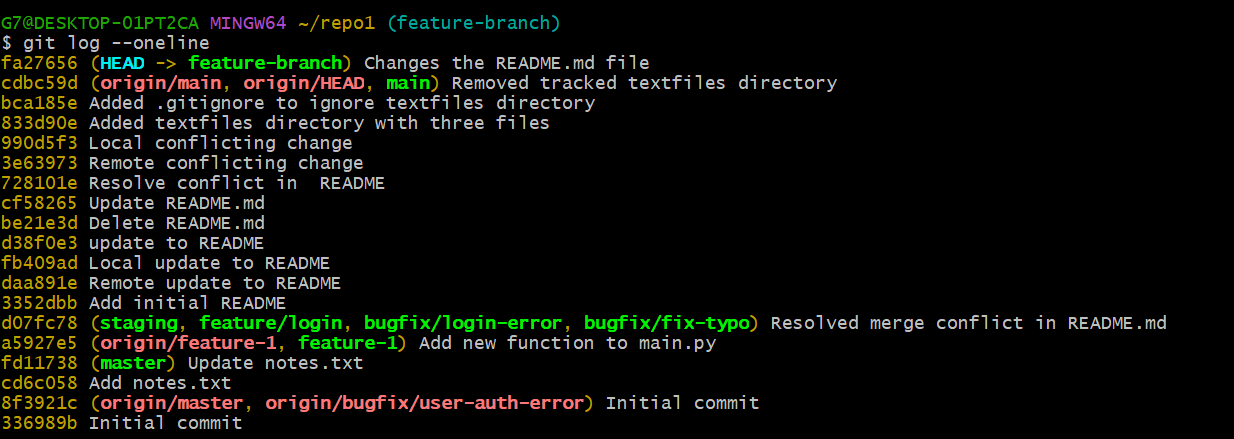


1. Confirm that your previous edits are back in the file.

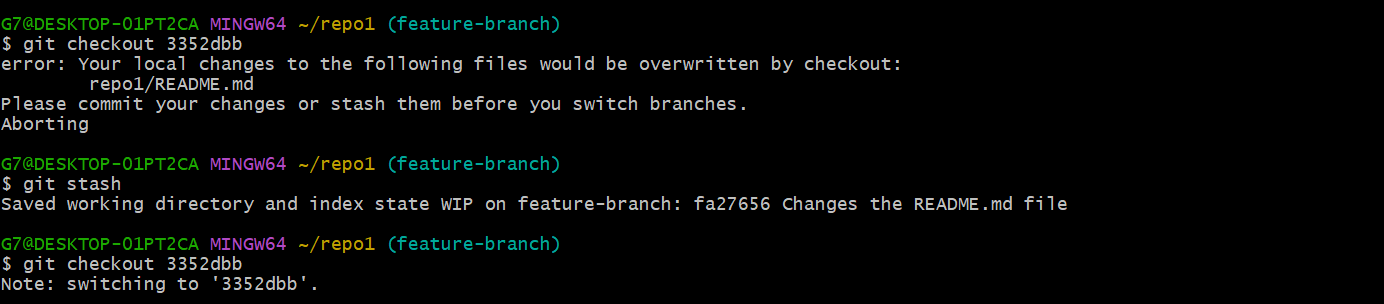


**Task 5 – Checkout a Specific Commit Using git log**

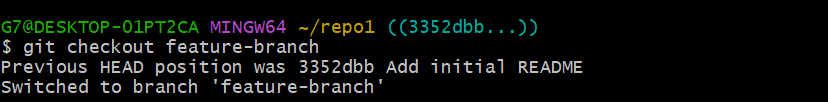
View commit history:



Checkout that commit:

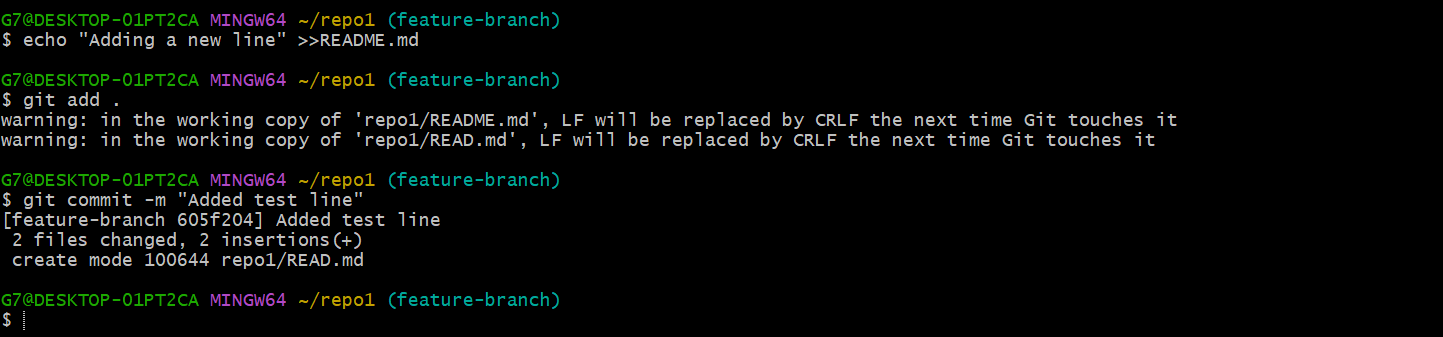


To return to your main branch:

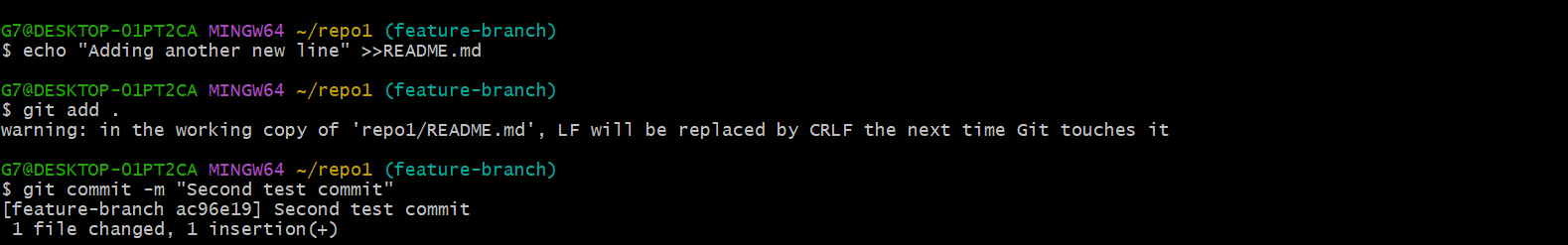


**Task 6 – Resetting Commits (Soft vs Hard Reset) (With Verification Steps)**

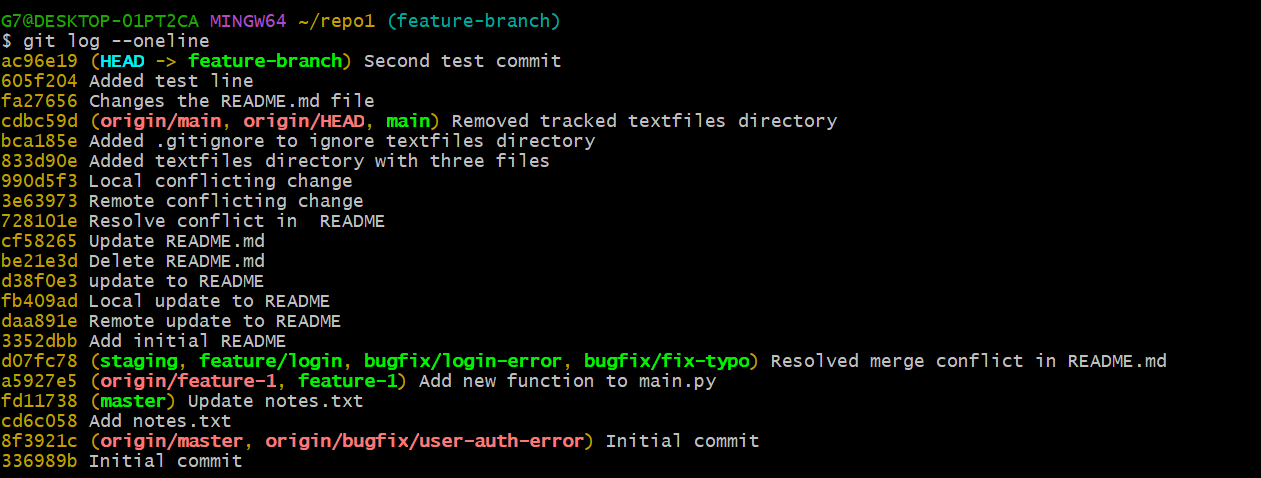
**Add a new line in any file and commit it:**



**Add another change and commit again:**



**View the history before reset:**



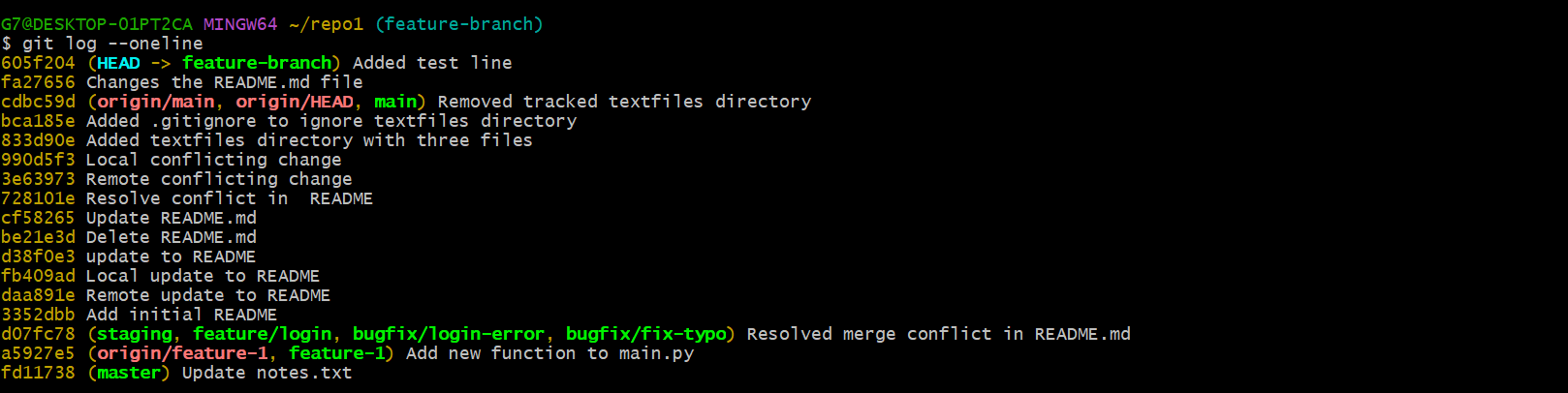
**Check file contents for both changes:**



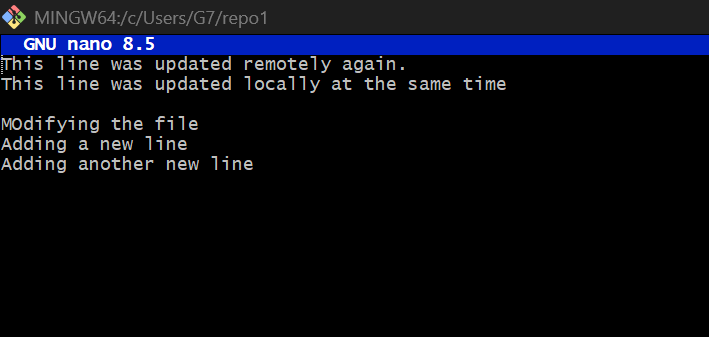
**Perform a soft reset (keeps changes in working directory):**



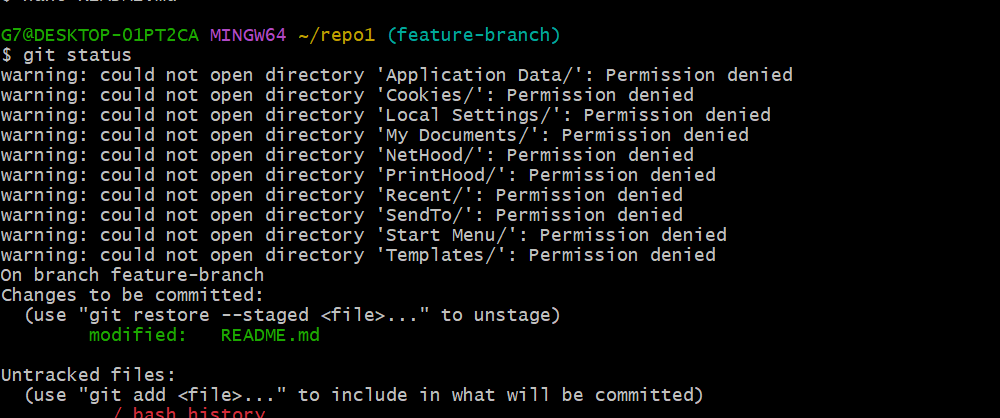
**Check commit history after soft reset:**



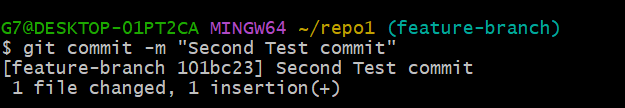
**Verify changes in the file:**



**Check git status:**



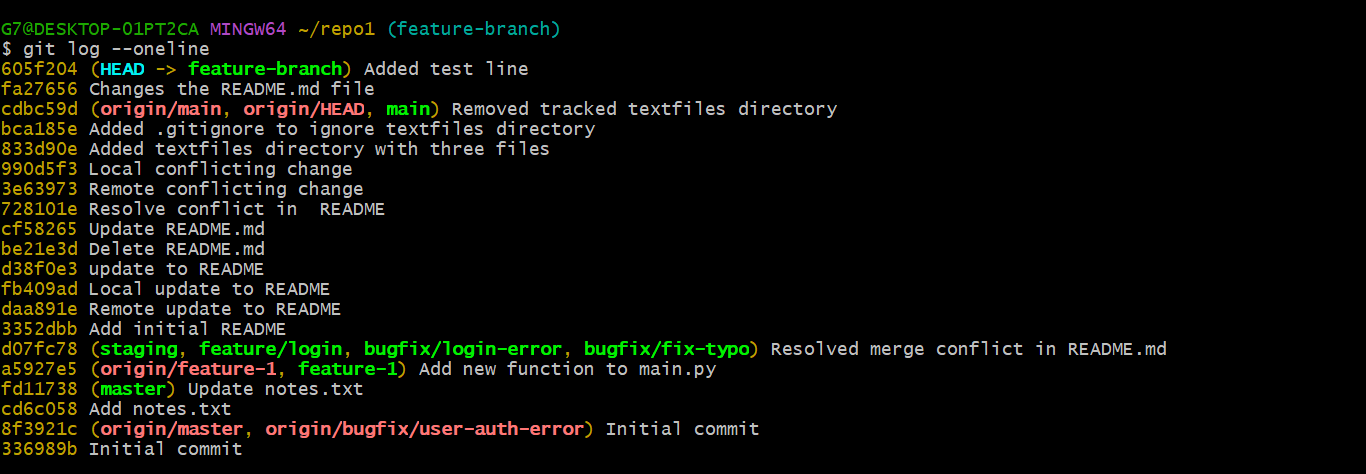
**Perform commit**



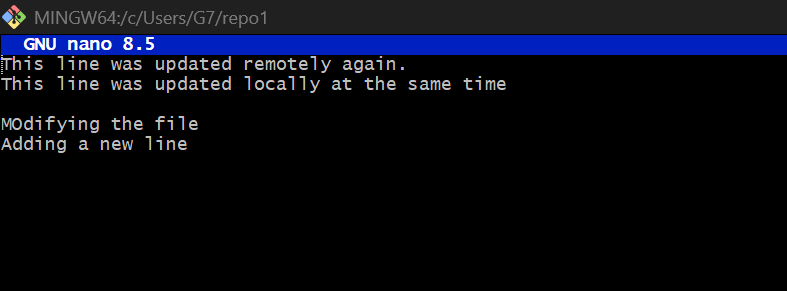
1. **Perform a hard reset (discards changes completely):**



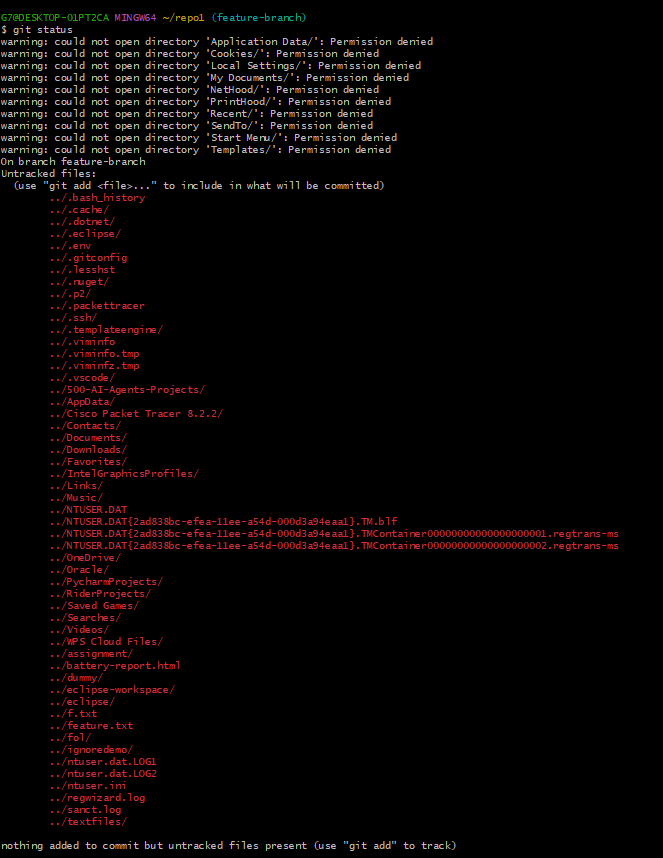
**Check commit history after hard reset:**



**Verify changes in the file:**

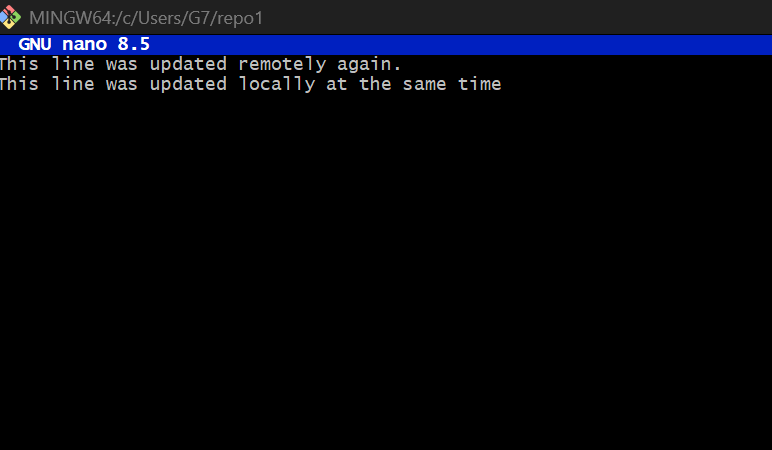


**Check git status:**

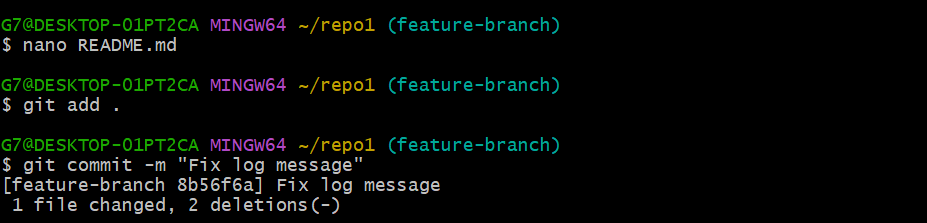


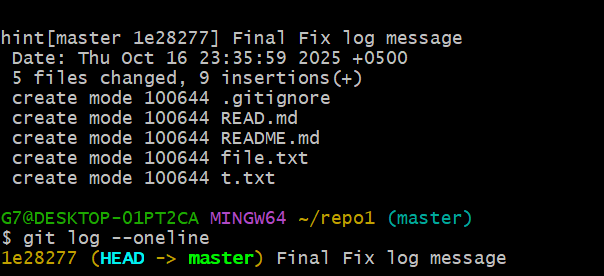
**Task 7 – Amending the Last Commit**

Make a small change in any file.



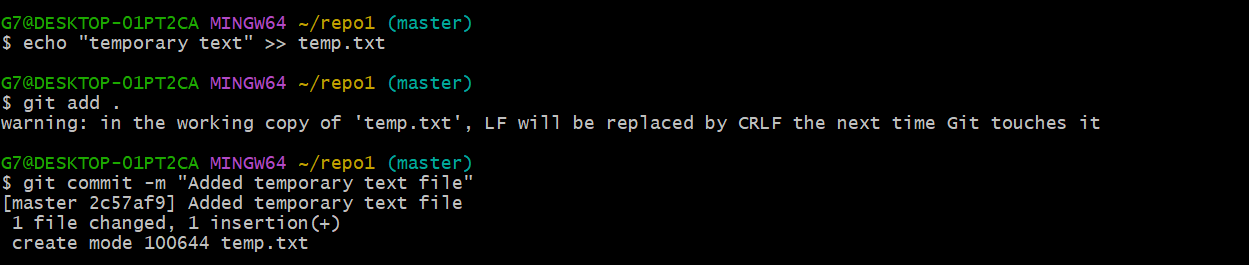
Stage it and commit:



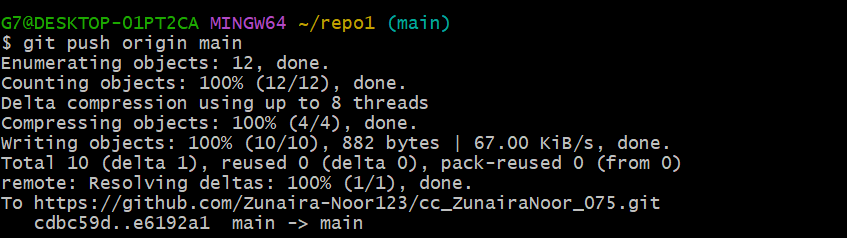


**Task 8 – Reverting a Commit (Safe Undo on Remote Branch)**

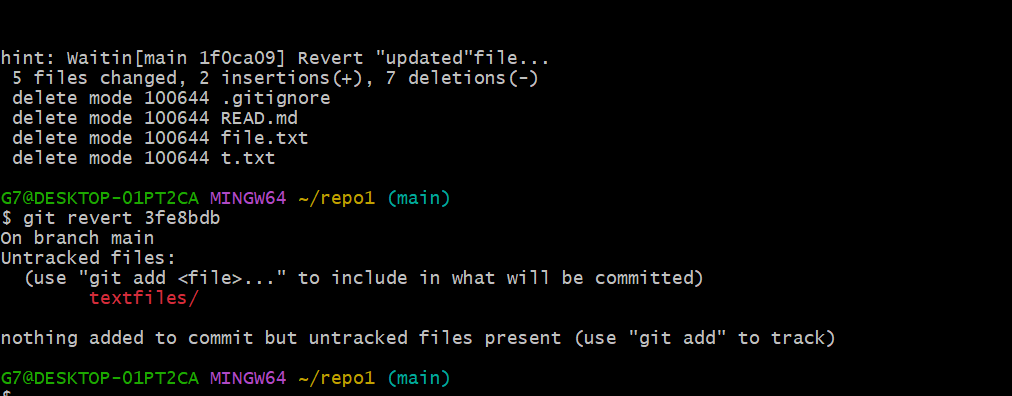
Make a change and commit it:



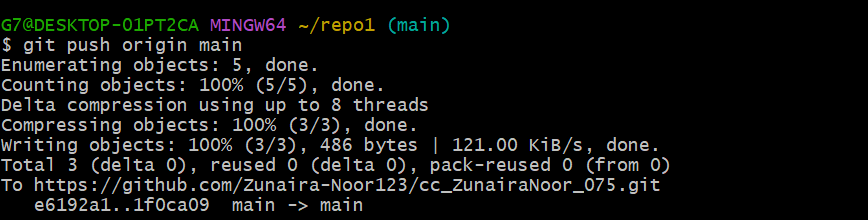
Push the revert commit:



Now revert that commit safely (do not delete it):



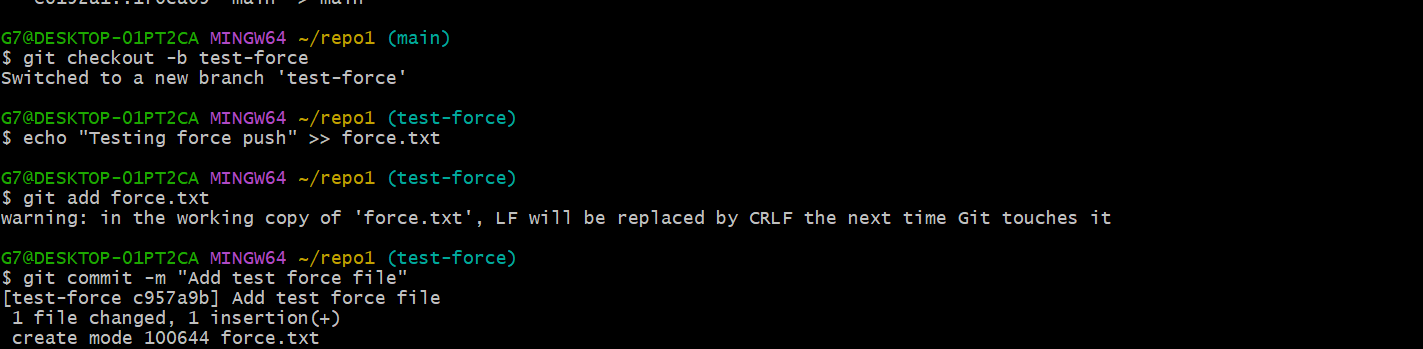
Push the revert commit:



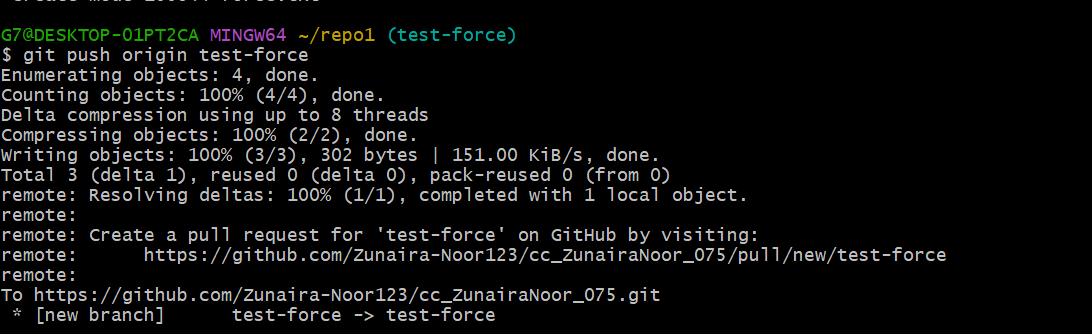
**Task 9 – Force Push (With Caution)**

**Create a new branch:**

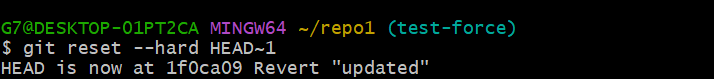
**Make and commit a small change.**



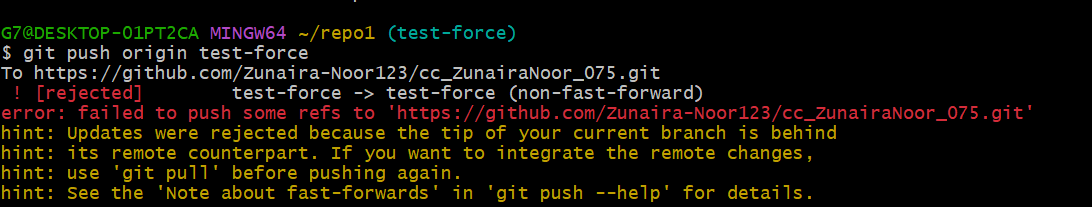
Push it:



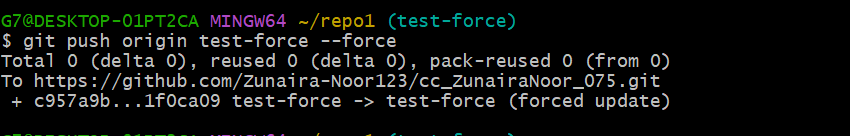
Perform a hard reset:



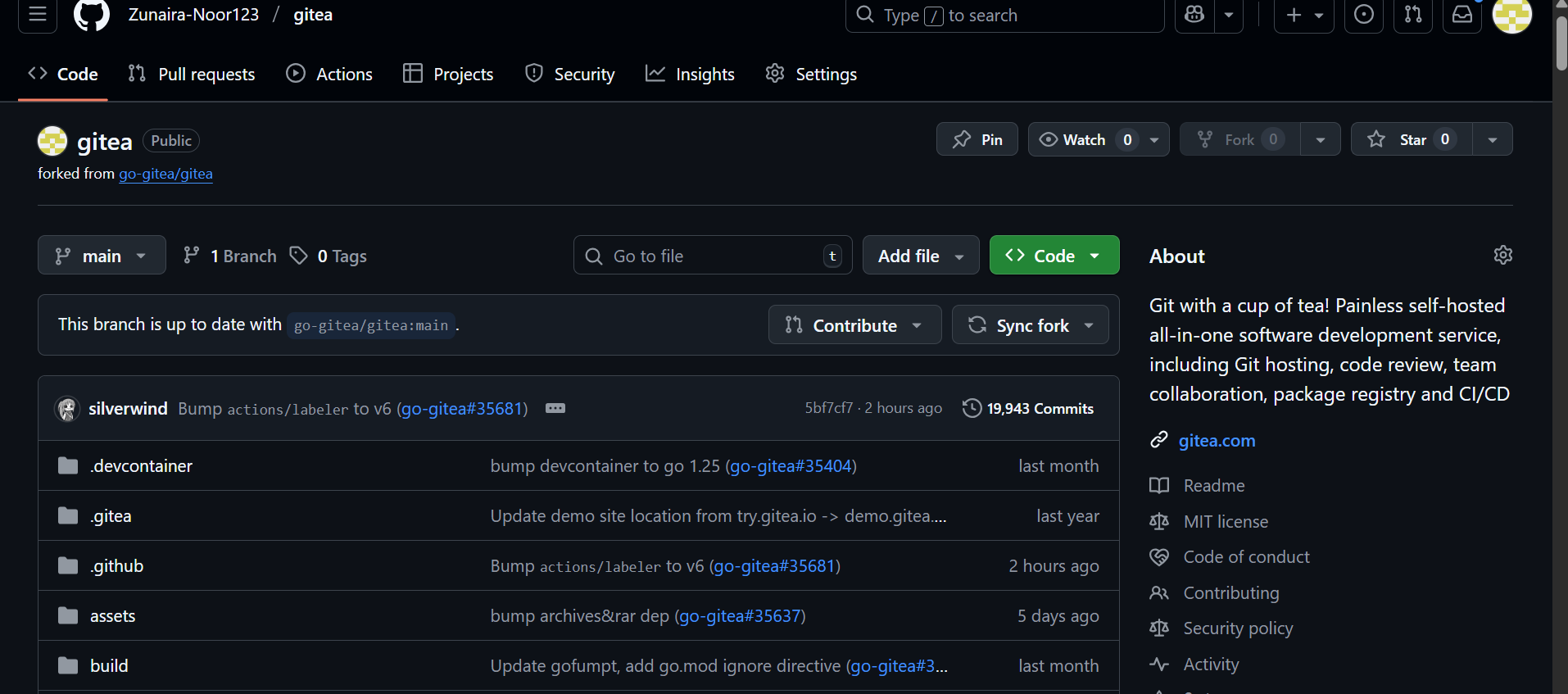
Push again



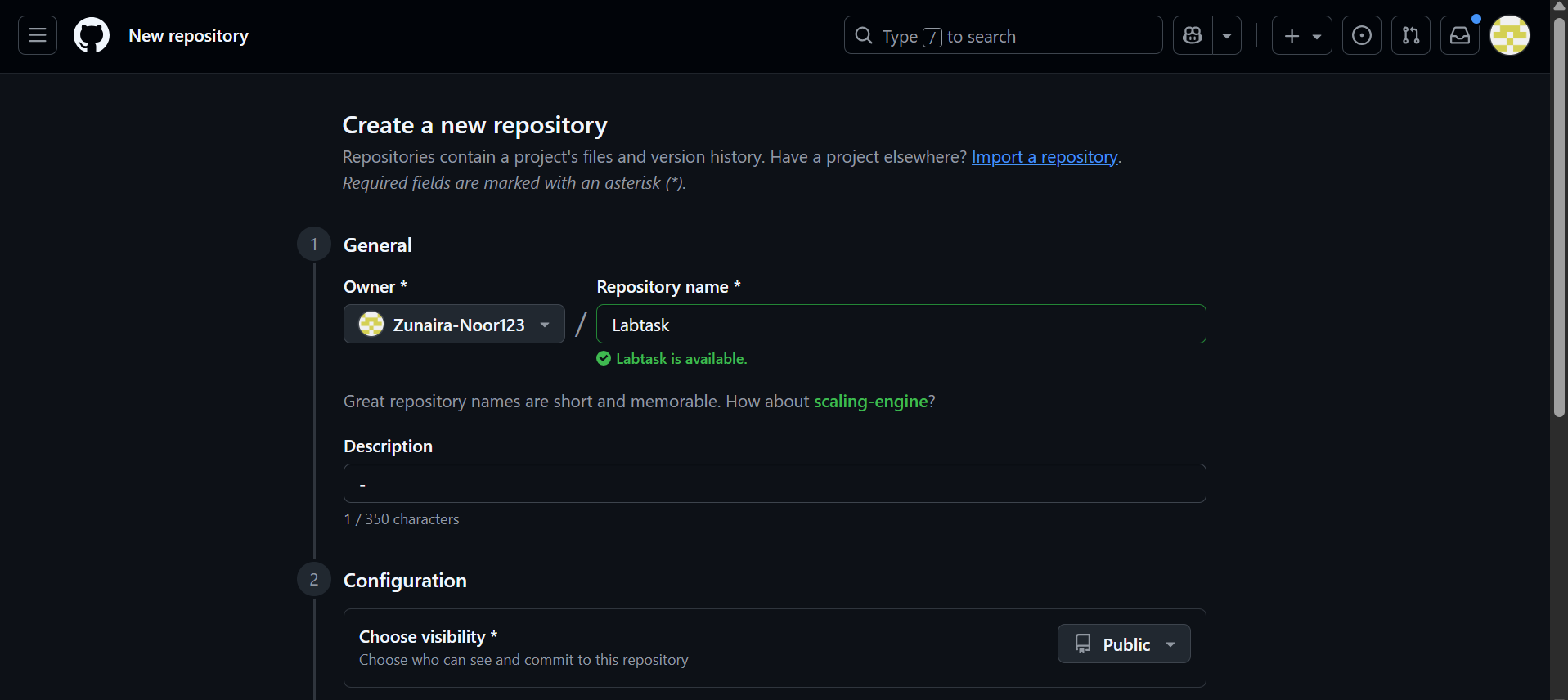
Now force-push to remote:

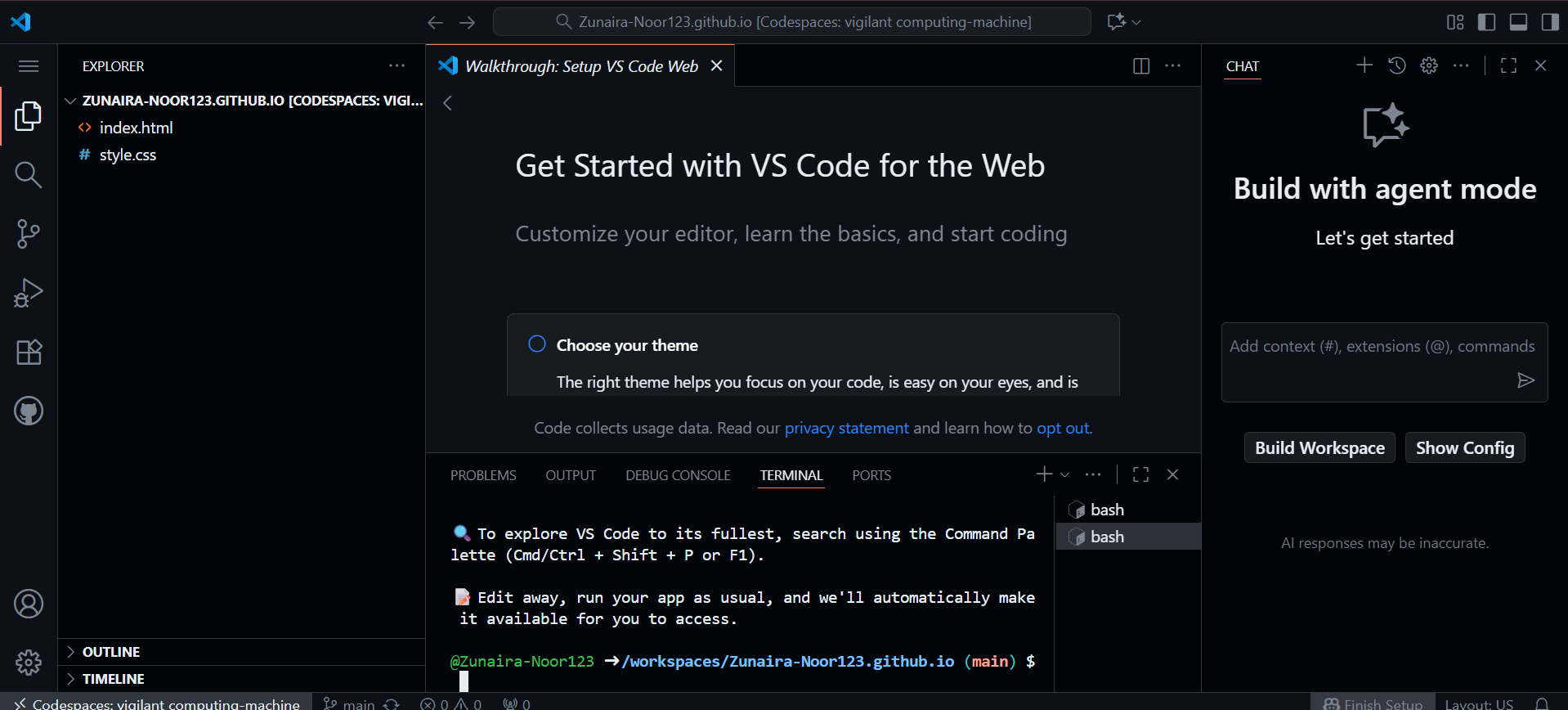


**Task 10 – Running Gitea in GitHub Codespaces via Docker Compose**

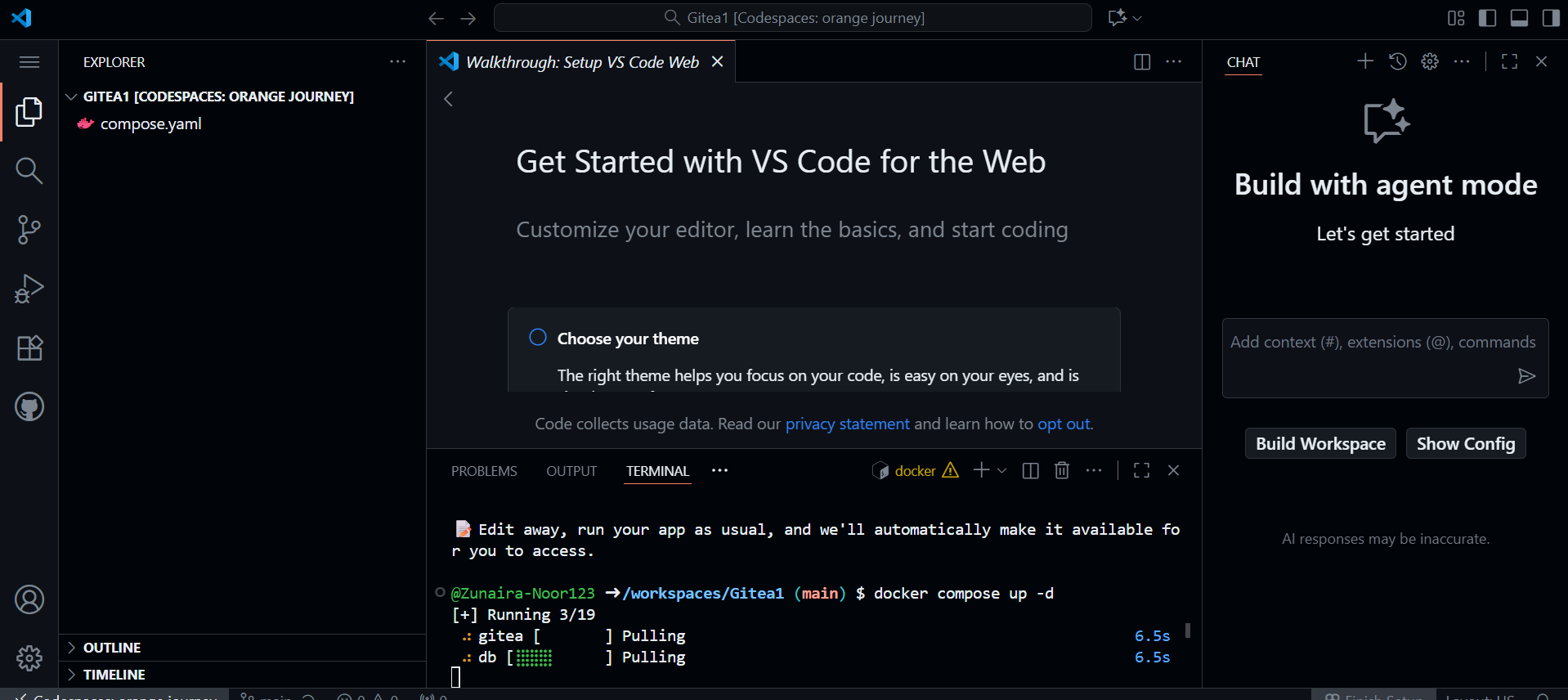


**Create a New Repository in Gitea**

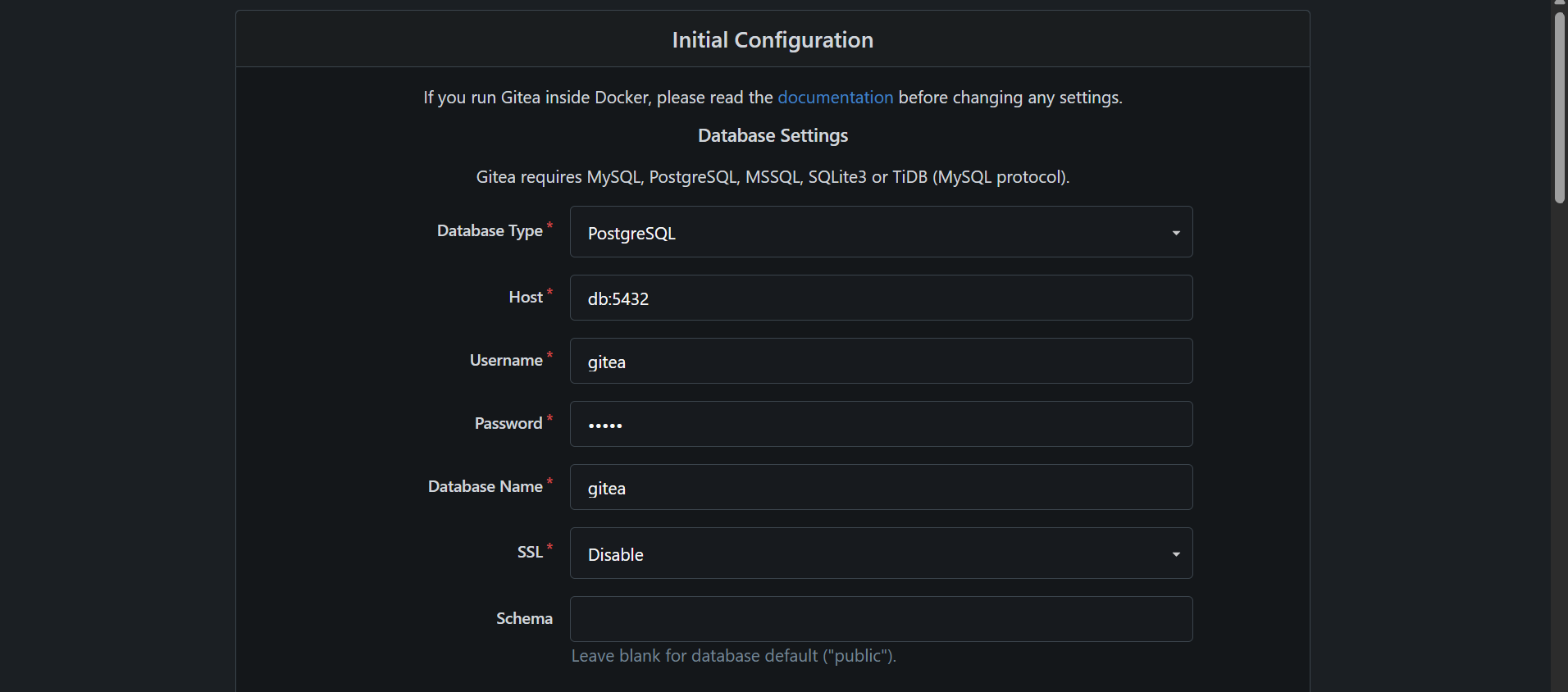




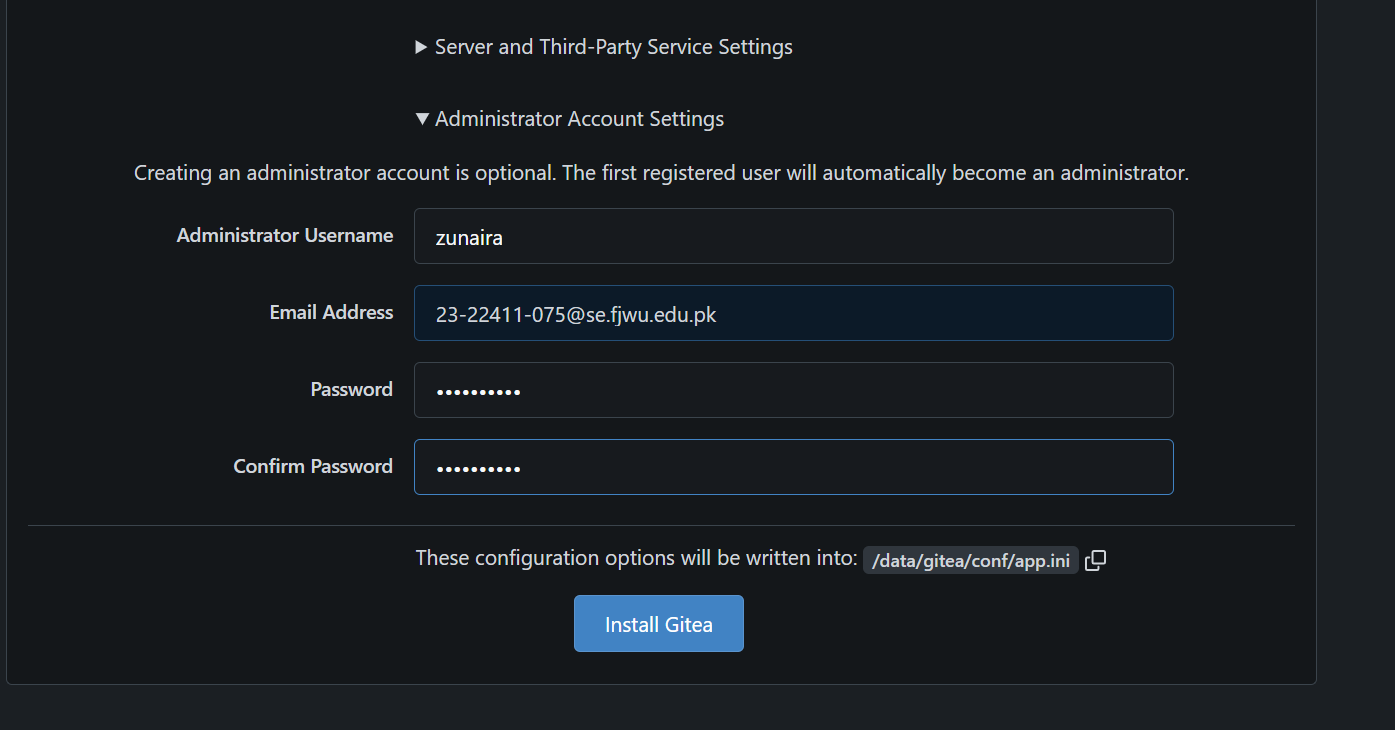
**Start Gitea with Docker Compose**



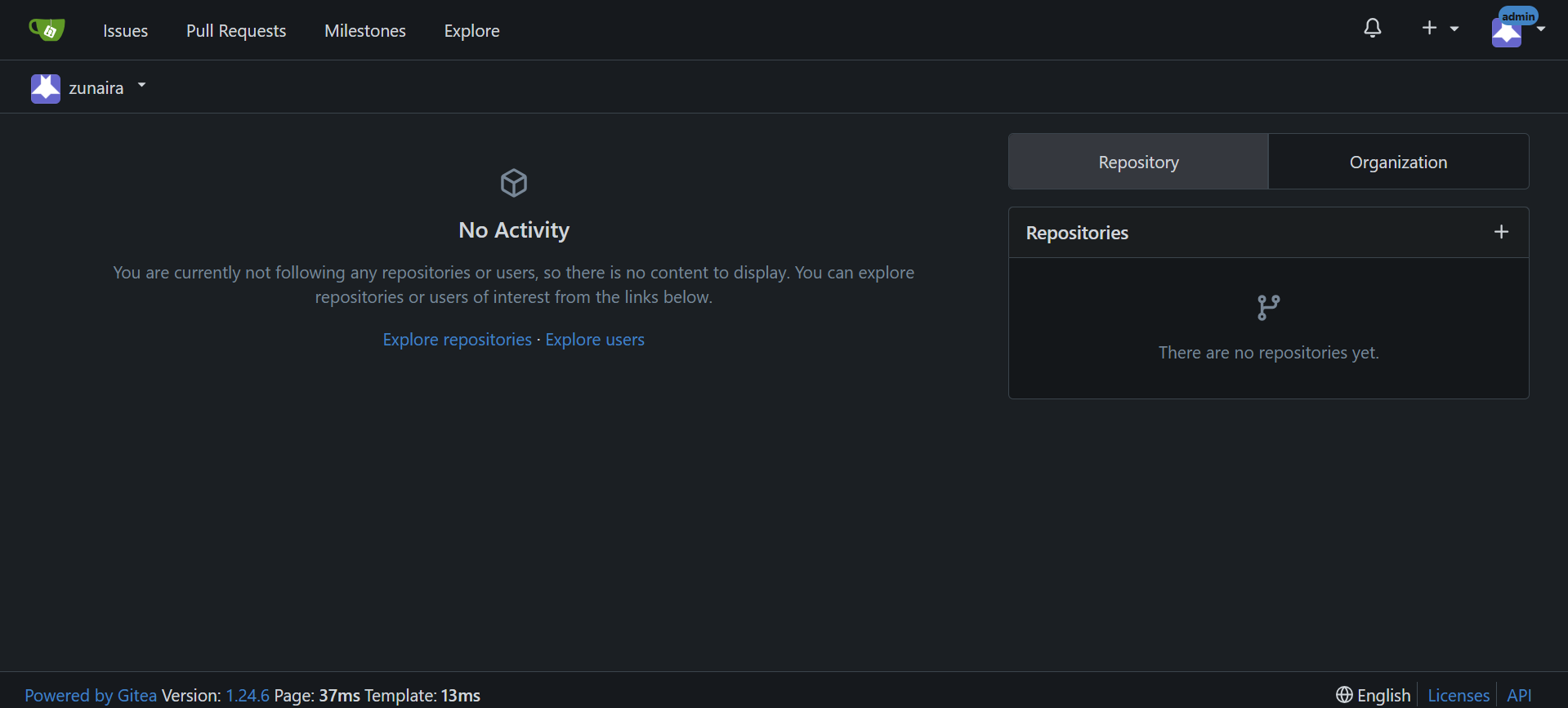
**Access Gitea Web Interface**



**Install Gitea**



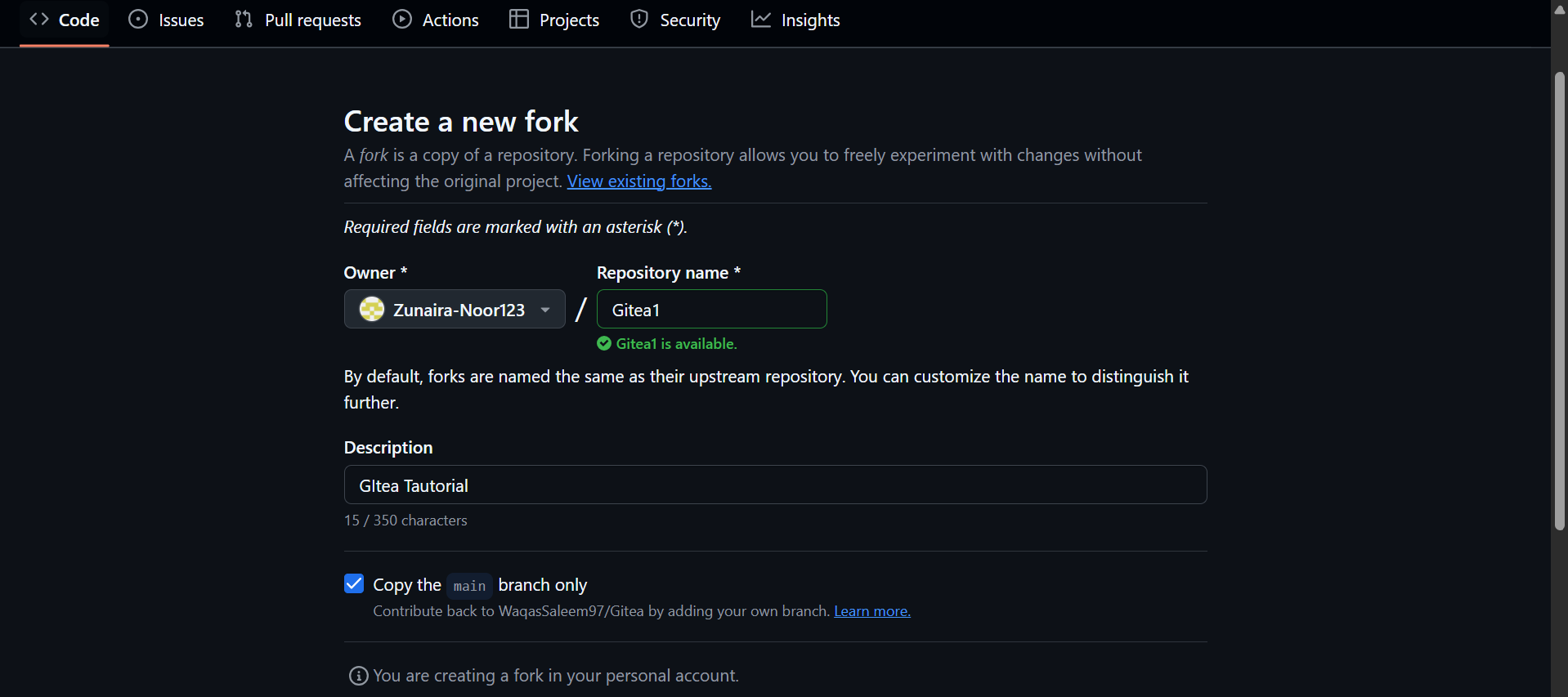
**Log In to Gitea**



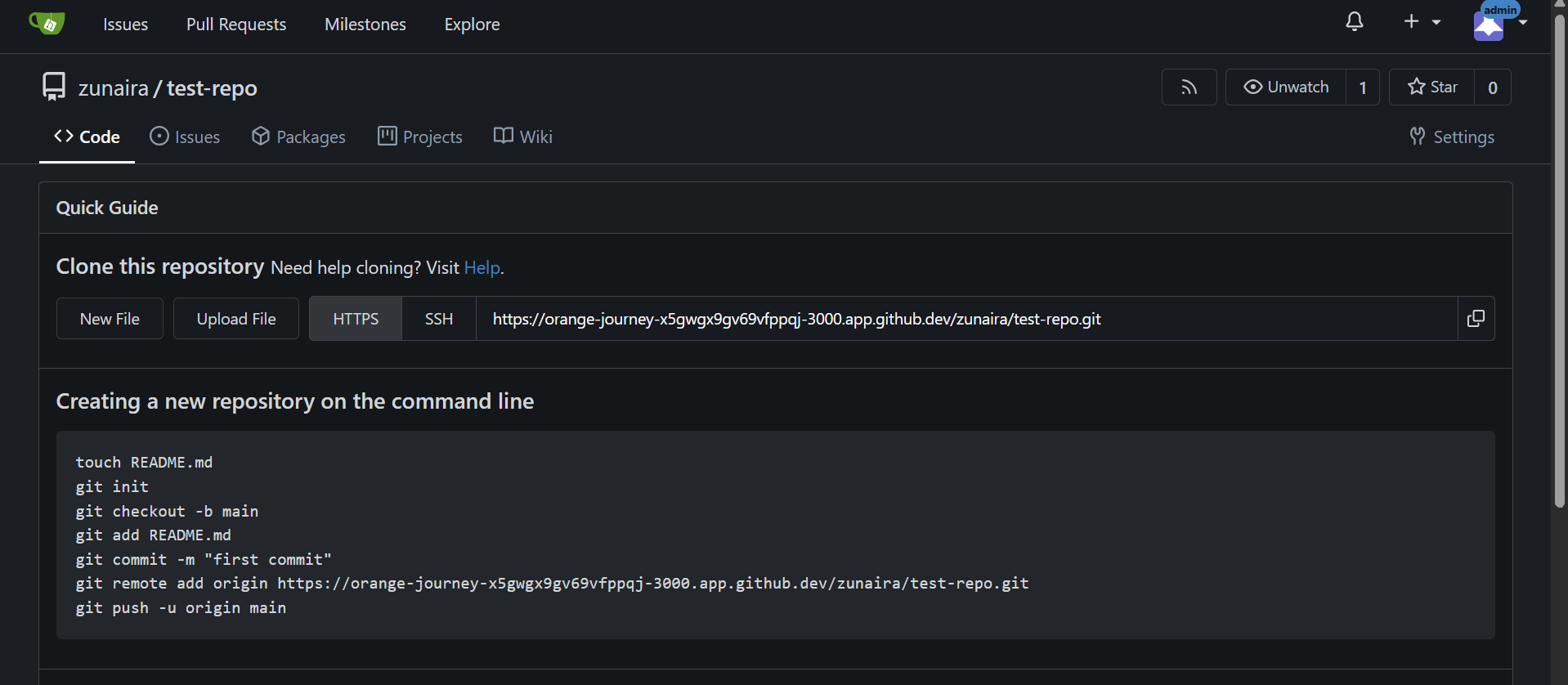
**Create a New Repository in Gitea**

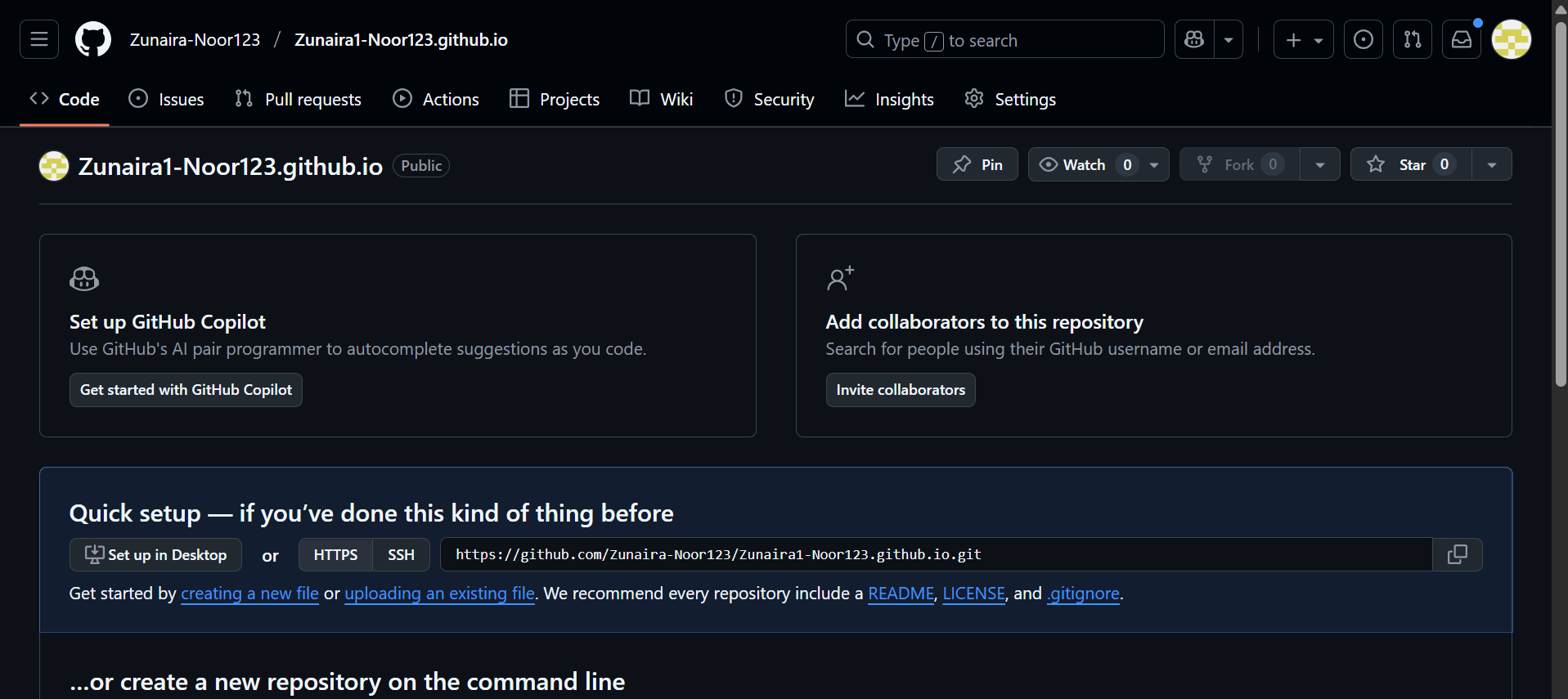
**Task 11 – Creating a GitHub Pages Portfolio Site**

**Create a GitHub Pages Repository**

****

**Open the Forked Repo in GitHub Codespaces**

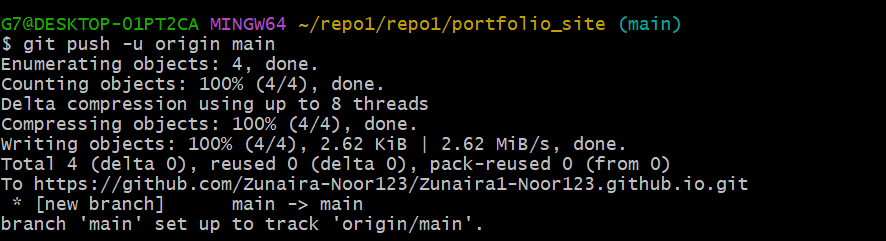
****



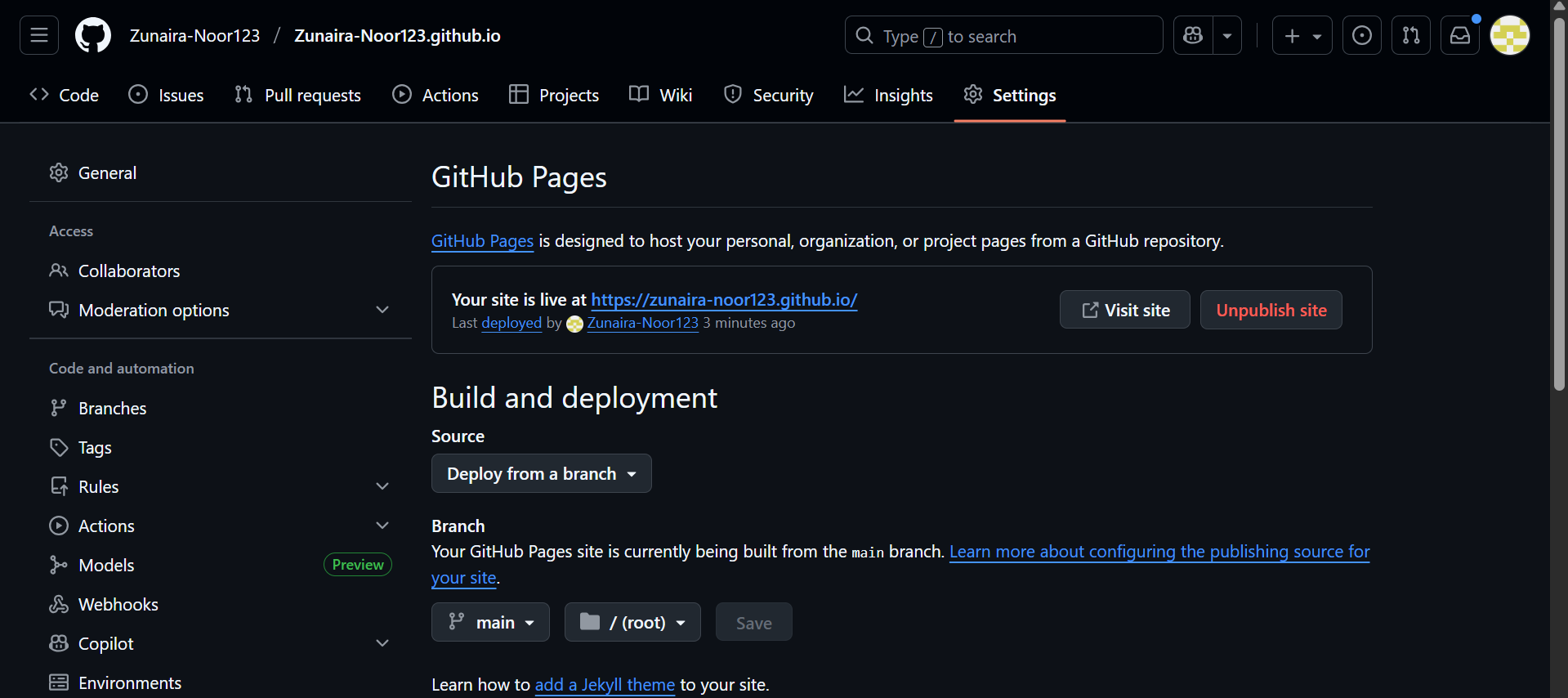
**Add Static Website Code**



**Push the Files to GitHub**



**Check GitHub Pages Settings**



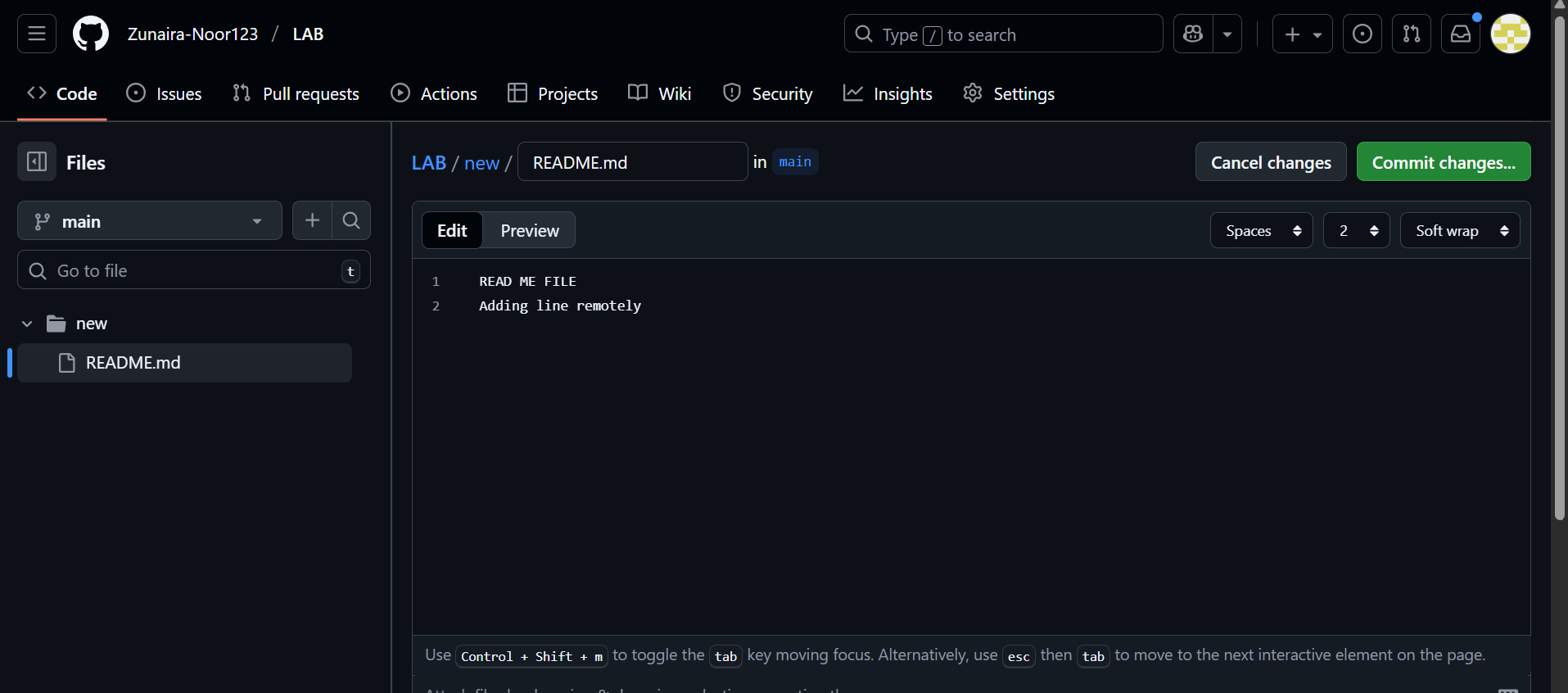
**Visit Your Live Site**



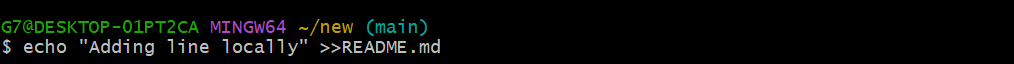
**Exam Evaluation Questions**

1. **Local vs Remote Conflict Resolution**

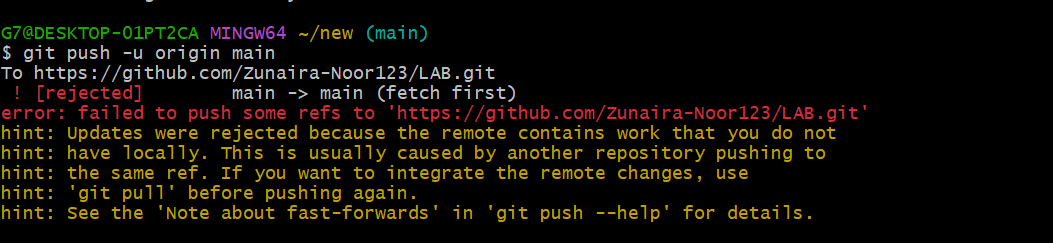
**On GitHub, edit a file (e.g., README.md) and commit the change**



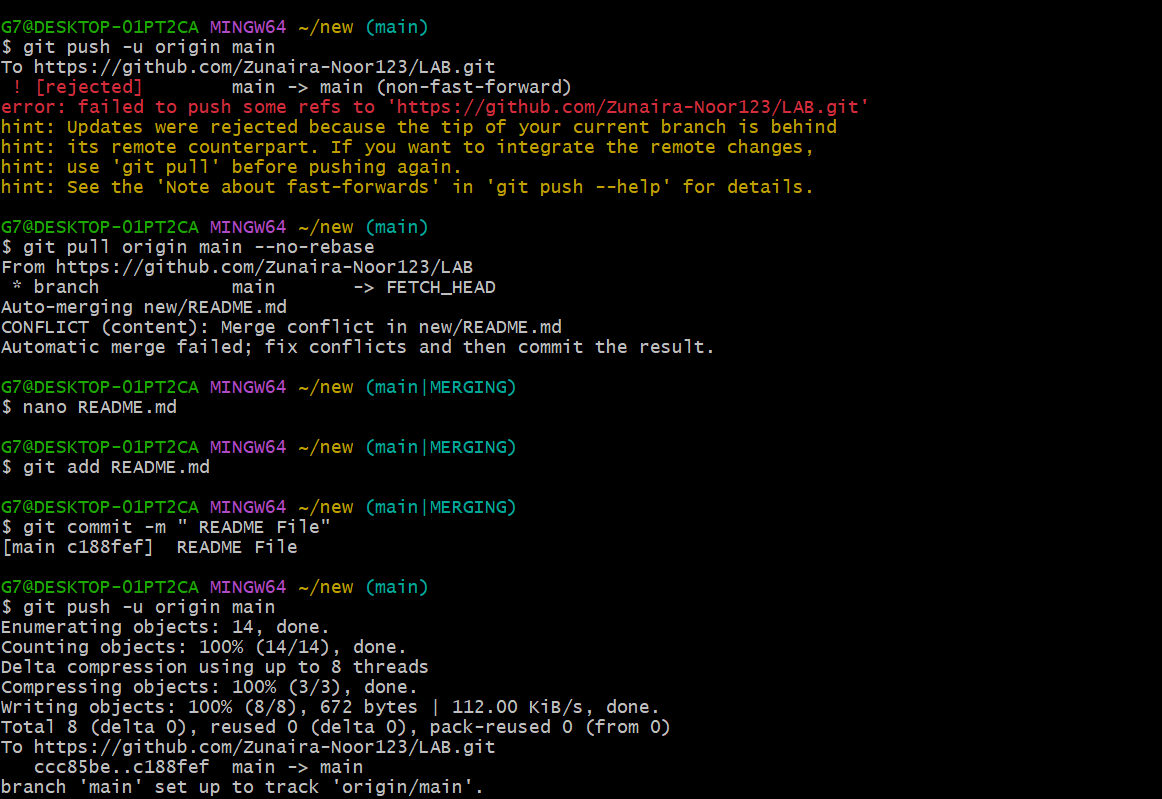
On your local machine, edit the same file differently (Avoid Conflict) and commit.



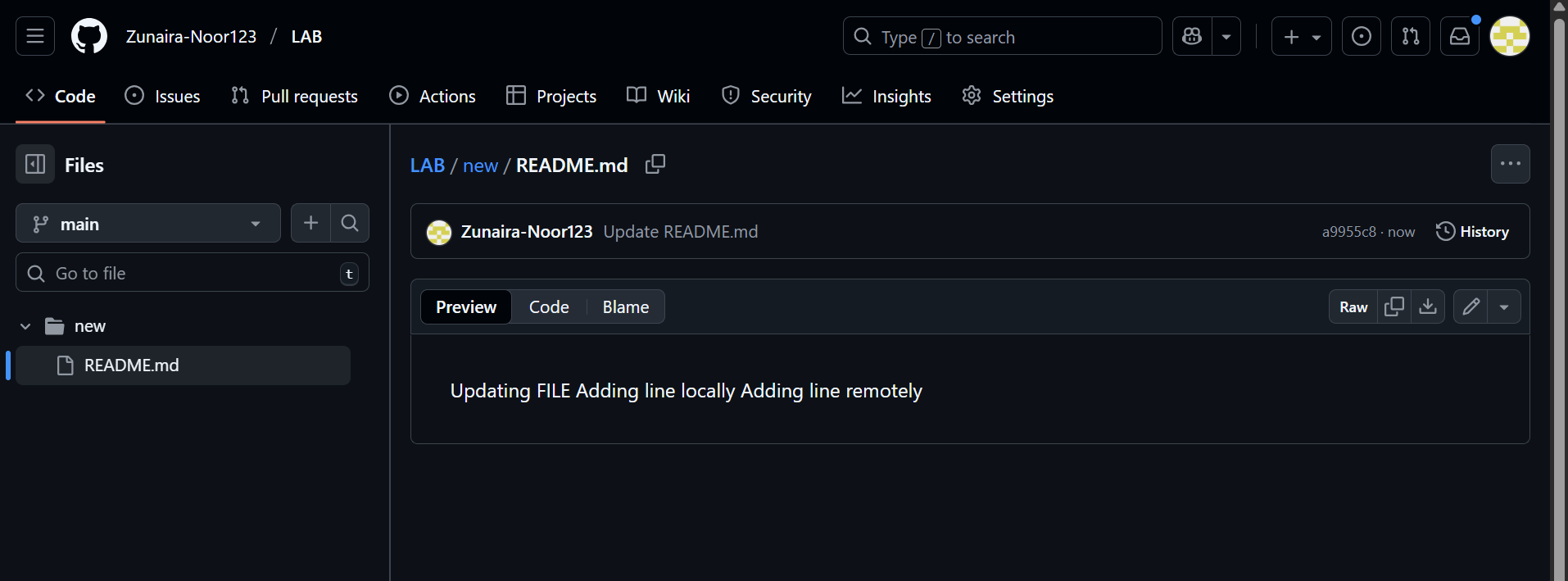
Try to push your local commit and observe the error



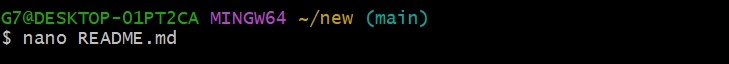
**2. Manual Merge Conflict Handling**



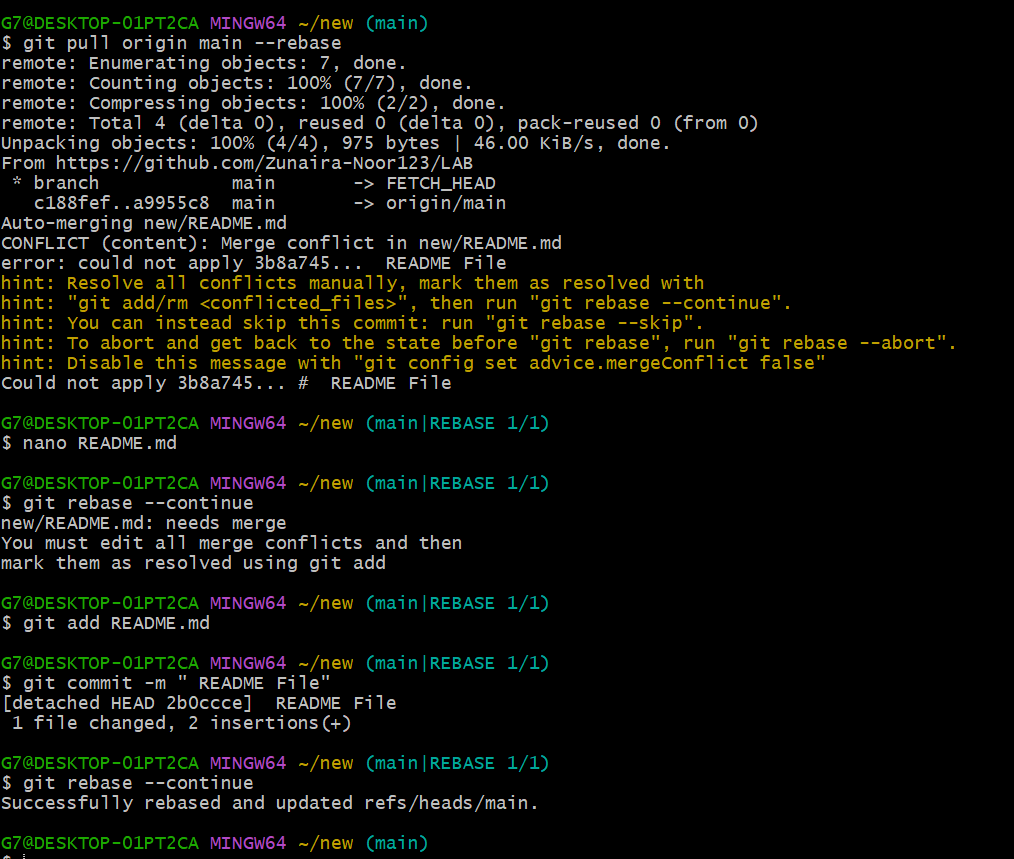
On GitHub, change a specific line in a file and commit.

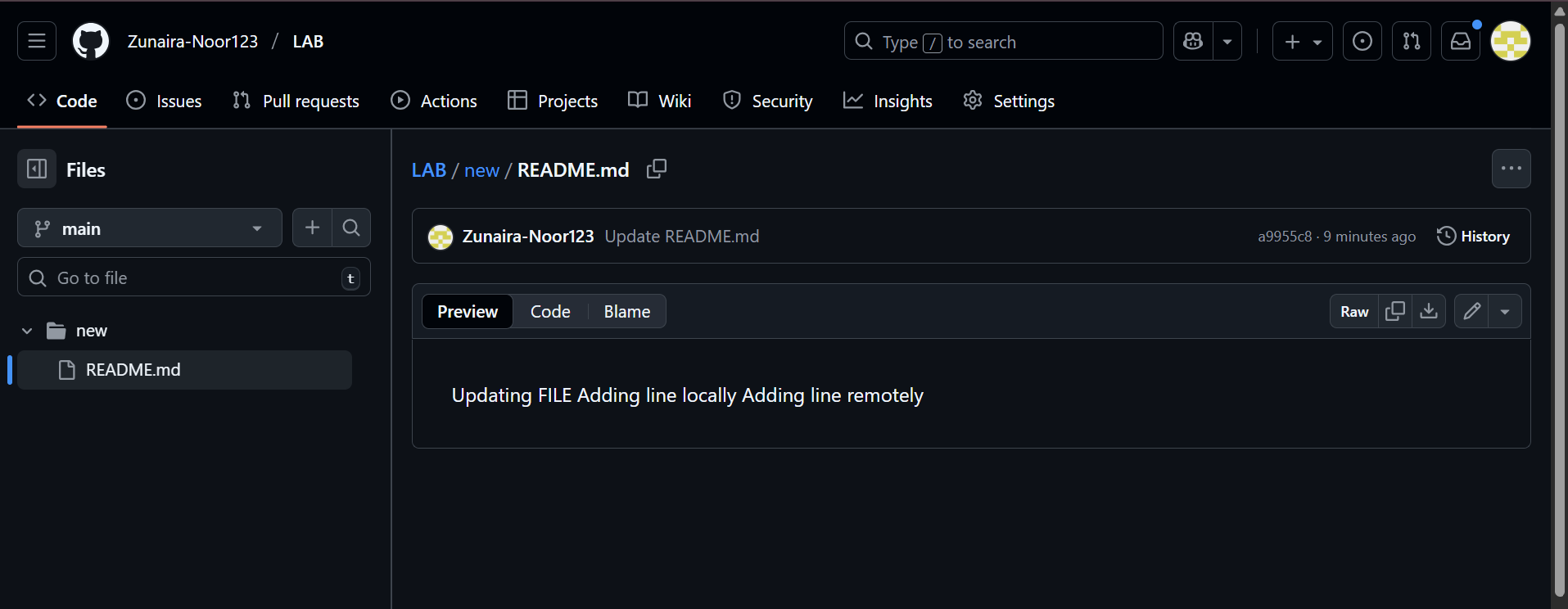
|

Locally, change the same line differently and commit.

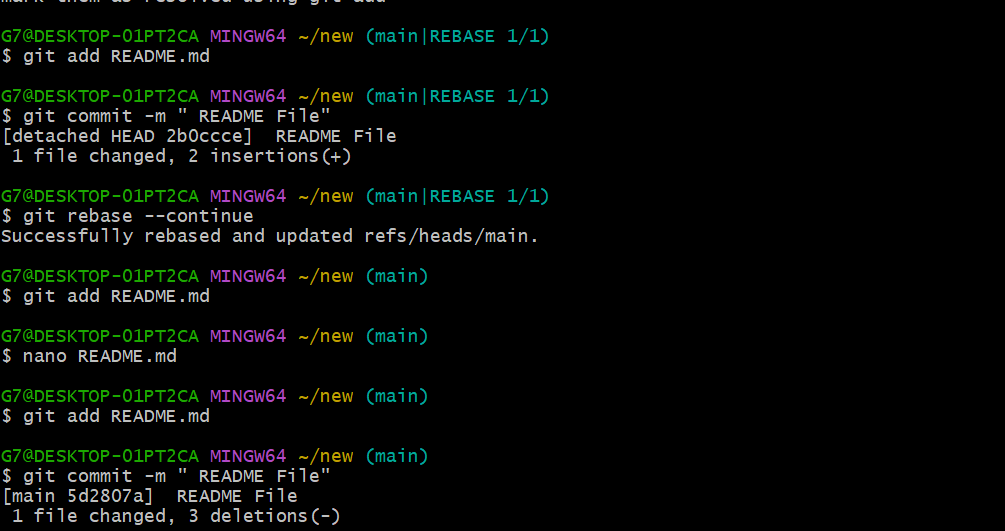


Use git pull --rebase to fetch changes and trigger the conflict.

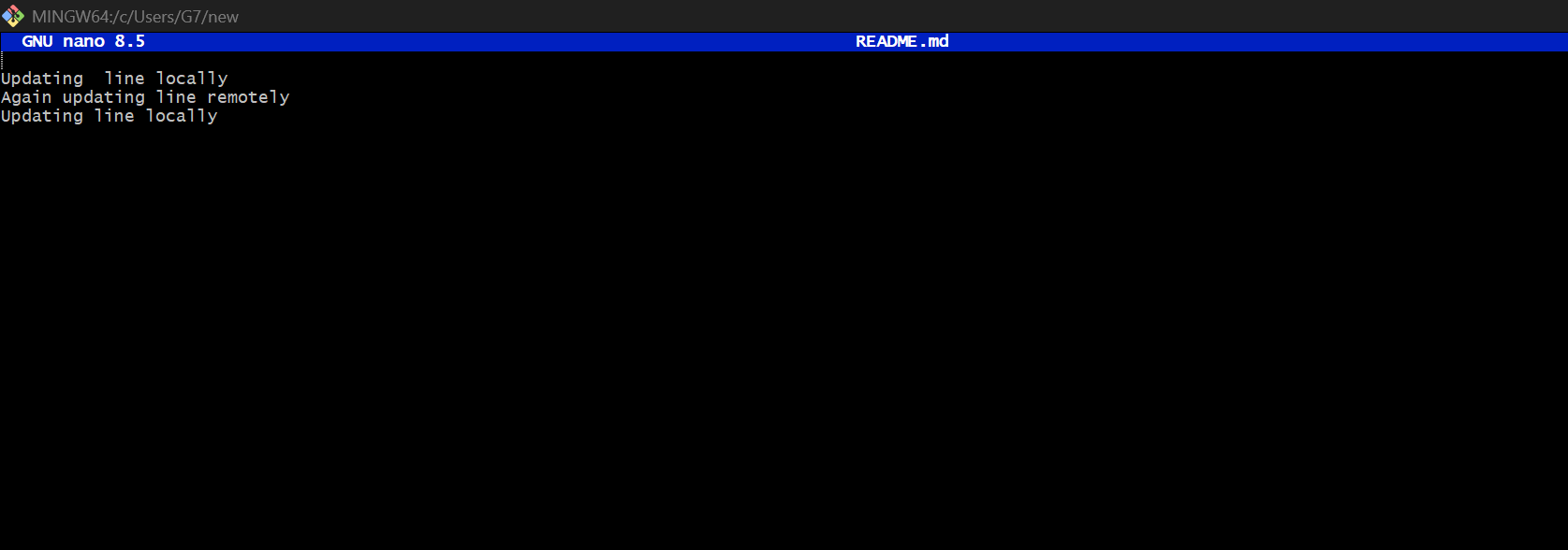




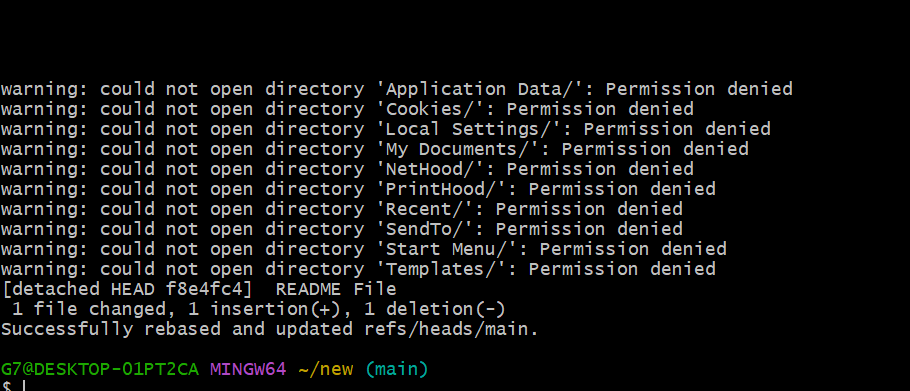
Edit the conflicted file to resolve the conflict manually



Edit the conflicted file to resolve the conflict manually



Mark the conflict resolved (git add <file>, git rebase --continue) and push

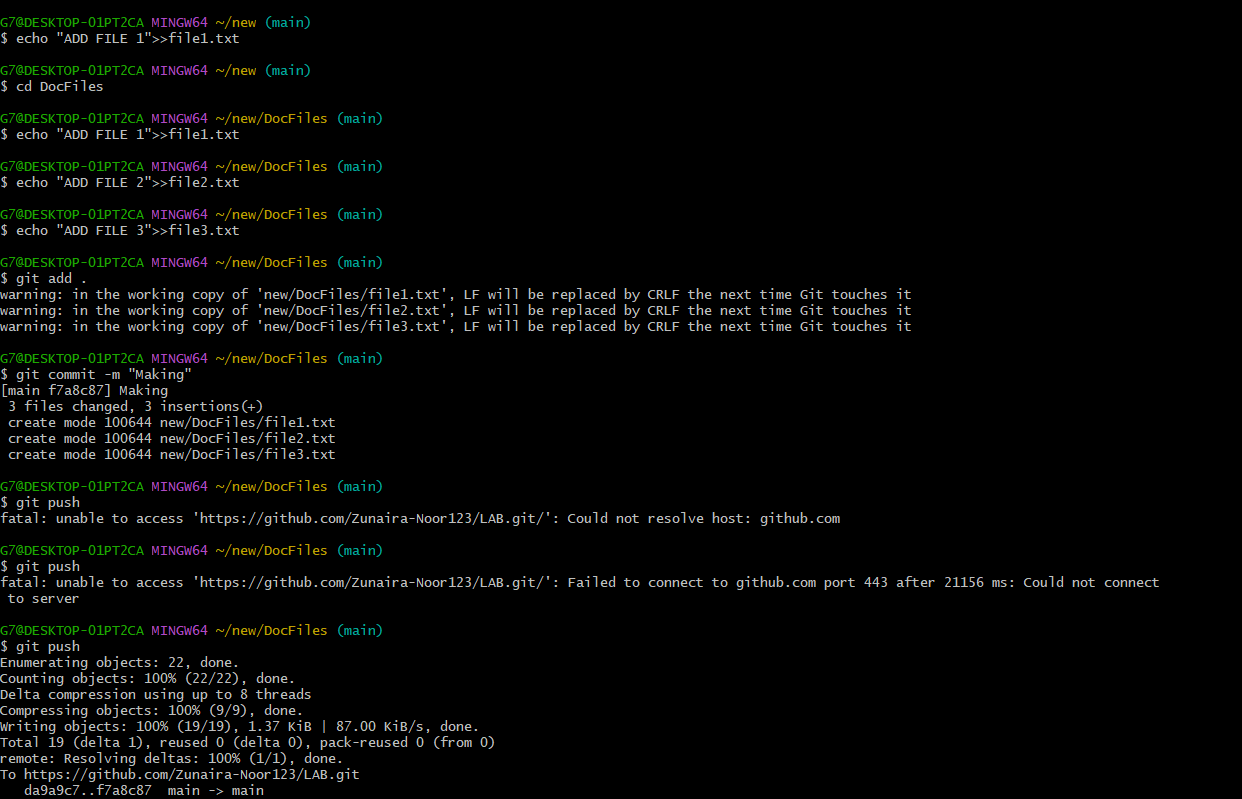


1. **Managing Ignored and Tracked Files**

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

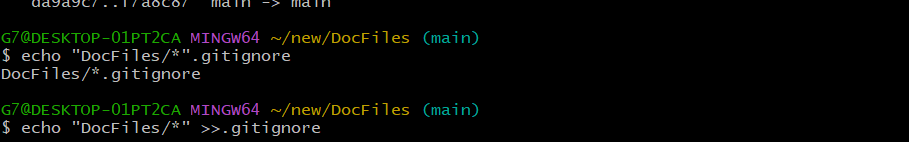


Commit and push the folder/files to GitHub

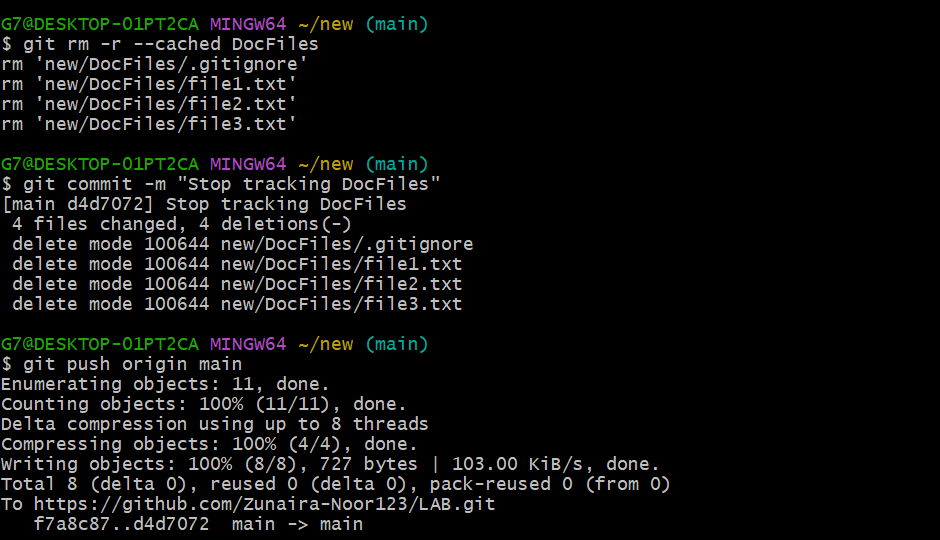


Add the folder to your .gitignore file

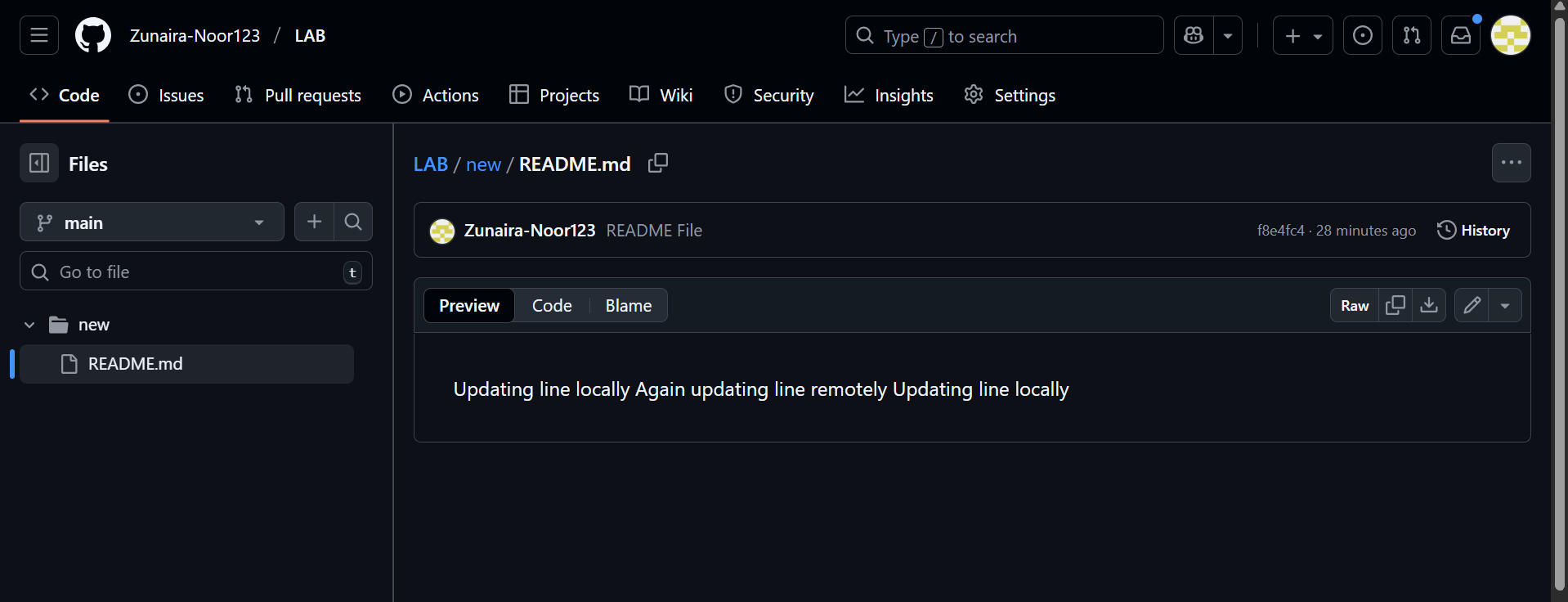
Commit and push the .gitignore update



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

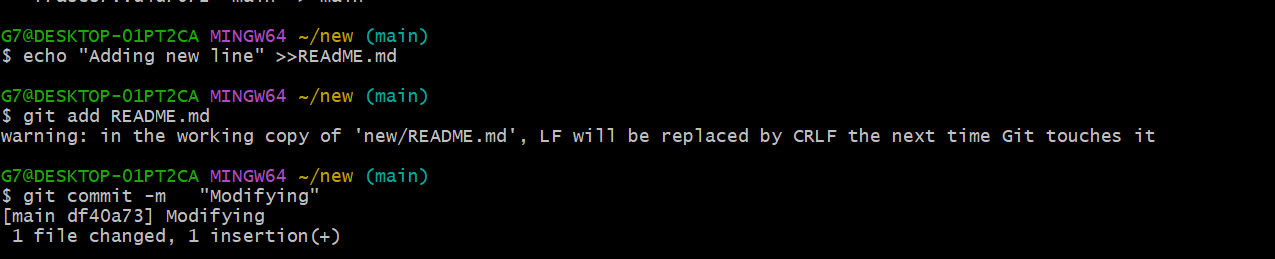


Commit and push the change, then verify the folder is no longer tracked on GitHub

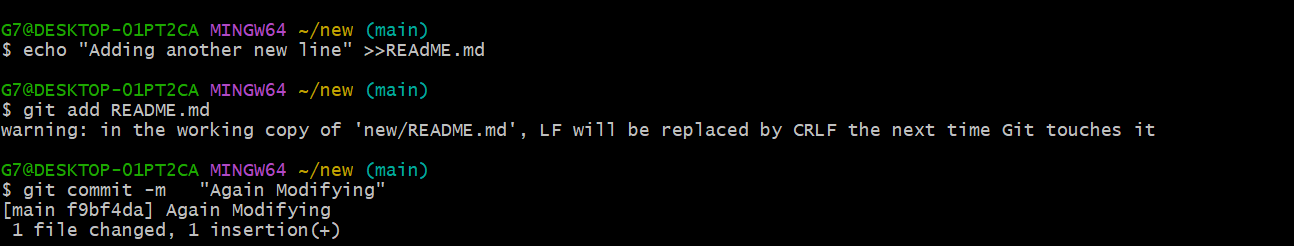


**4. Commit History Manipulation and Recovery**

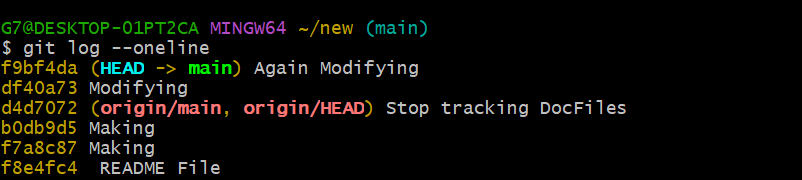
Make a change and commit it.



Make another change and commit again.



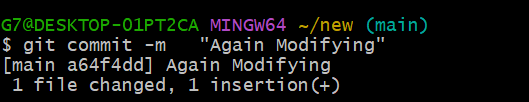
View your commit history



Perform a soft reset (git reset --soft HEAD~1) and observe your file and history



Make commit again.



Perform a hard reset (git reset --hard HEAD~1) and observe the changes

