**Cloud Computing Lab**

**Lab 03 tasks**

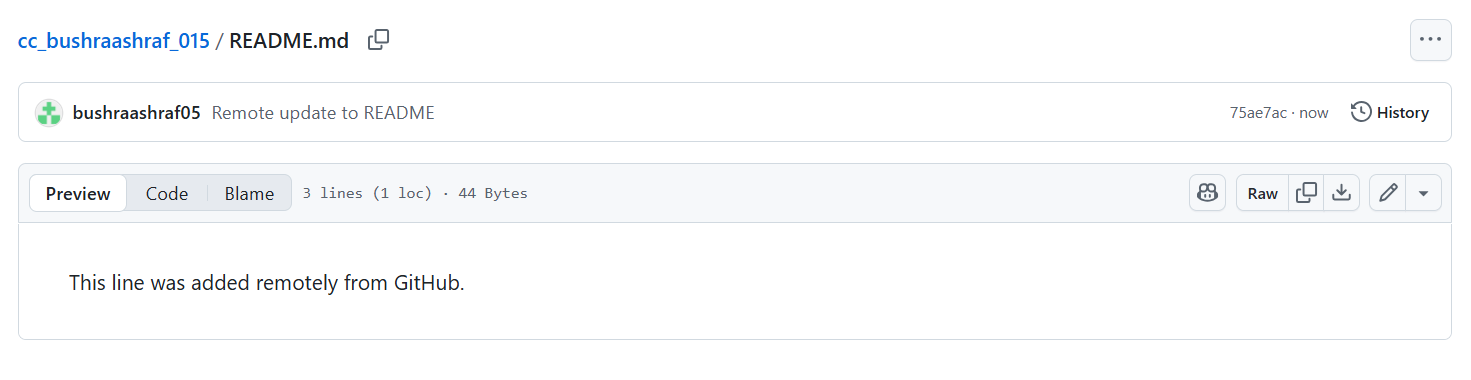
**Working with Git History, Stashing, and Reverting Commits**

**Submitted To:** **Muhammad Shoaib**

**Submitted From: Bushra Ashraf Bhatti**

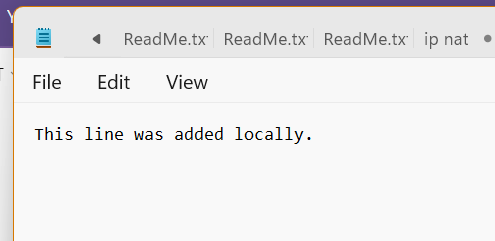
**Registration Number: 2023-BSE-015**

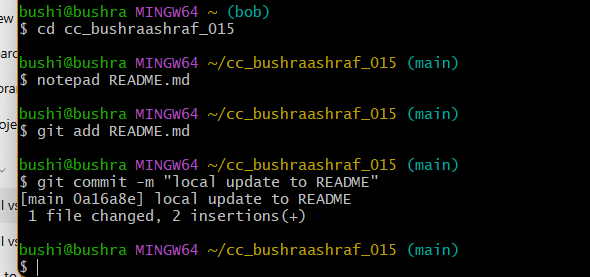
**Task 1 – Handling Local and Remote Commit Conflicts (Pull vs Pull --rebase)**

**Part A: Make a remote change on GitHub (browser):** 

We edited README.md **directly on GitHub** and created a **remote commit** with message Remote update to README. Screenshot saved as remote\_edit.png.

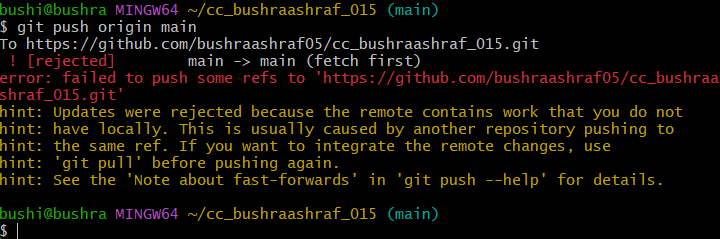
**Part-B: Make a conflicting local commit (terminal):**





We created a **local commit** that modifies the same part of README.md (message: Local update to README).

**Part C — Try to push (and capture the push rejection)**

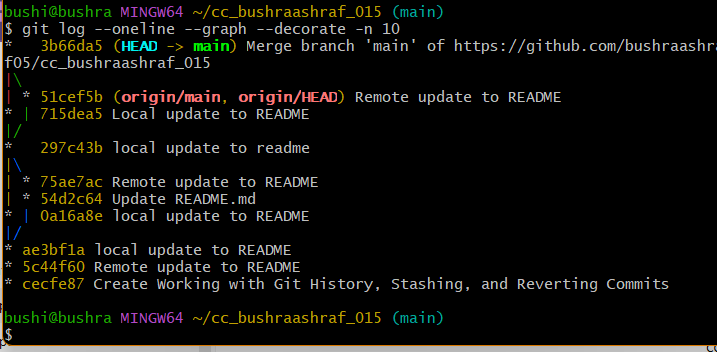
****

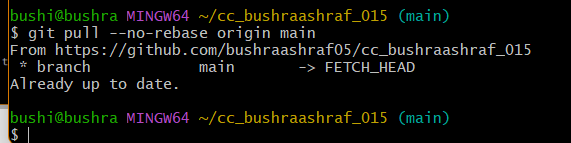
We attempted to push our local commit but **push was rejected** because the remote has a commit we don't have.

**Part D — Fix with a merge (default git pull / git pull --no-rebase)**

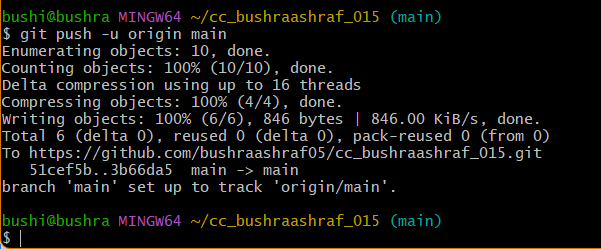
Fetch & merge remote into your local branch (explicit no-rebase):

git pull --no-rebase origin main





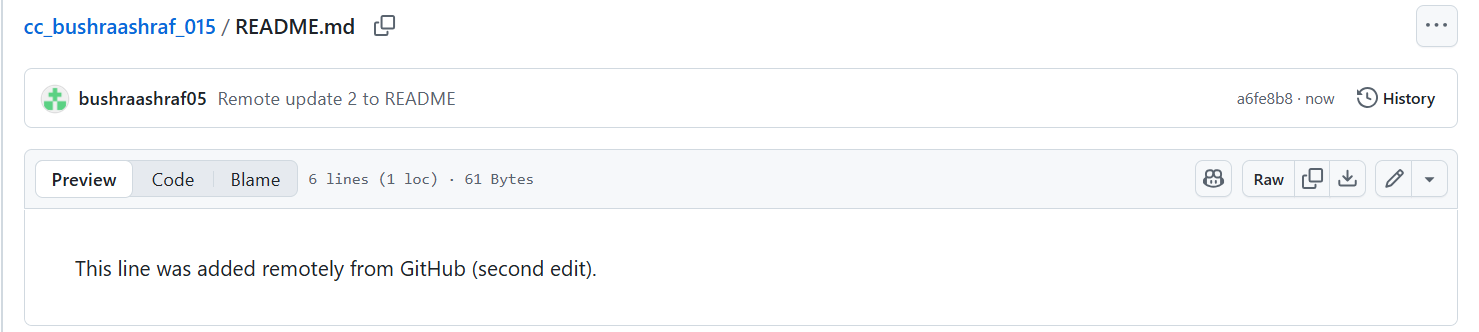
Now push the merged history:

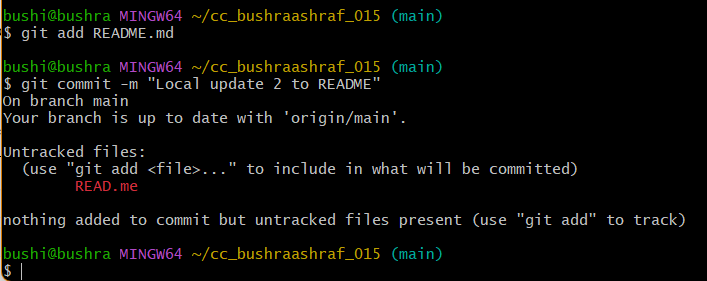


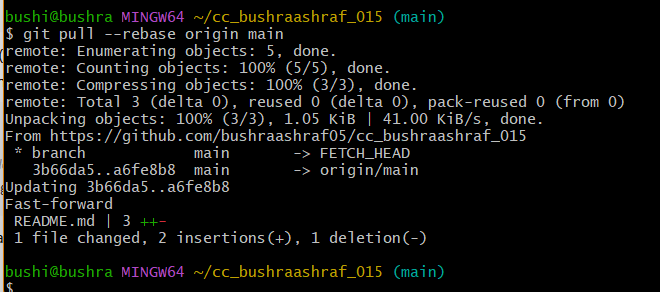
We pulled remote changes using a **merge** (git pull --no-rebase origin main). Git created a **merge commit** combining remote and local history. We then pushed the merged result.

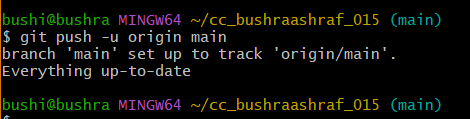
**Part E — Repeat the experiment but use rebase**

On GitHub, edit README.md again (a new change). Example line:









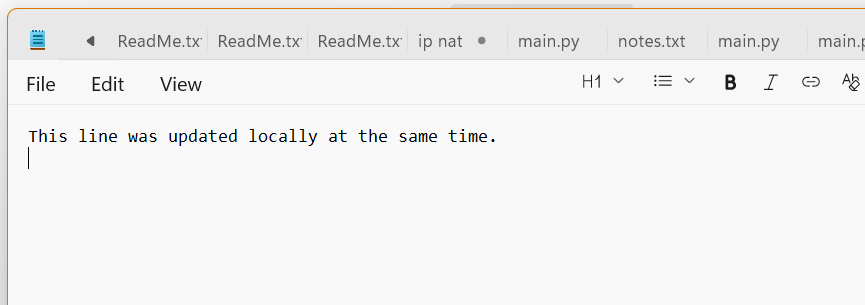
## Task 2 – Creating and Resolving Merge Conflicts Manually

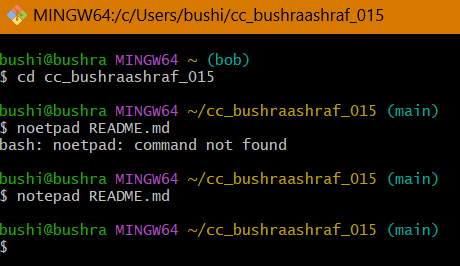
## Part A — Make the conflicting change on GitHub (remote):

## 

We created a **remote commit** on GitHub that changes README.md to say This line was updated remotely again.

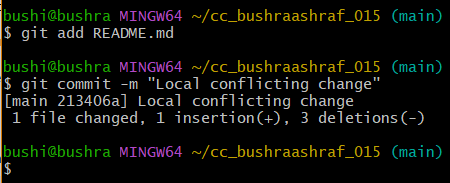
**Part B — Make a different conflicting change locally:**

****

****

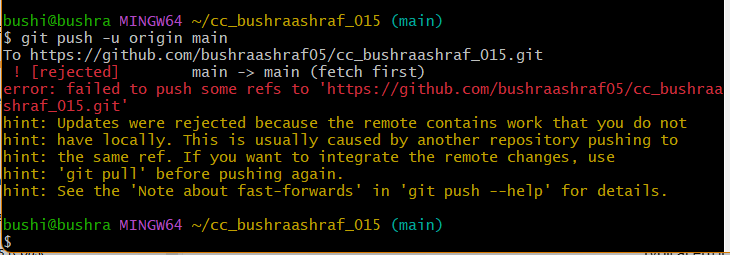
We edited the same line locally (different content) and saved the file.

**Part C — Stage and commit the local change:**

****

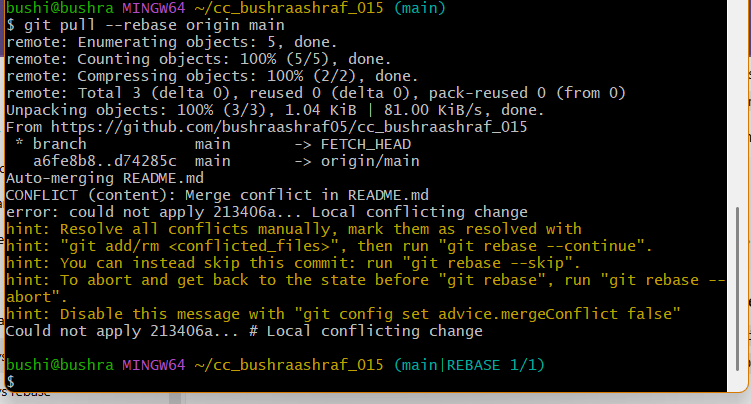
We created a **local commit** (Local conflicting change) that conflicts with the remote commit.

**Part D — Try to push (you’ll get rejected):**

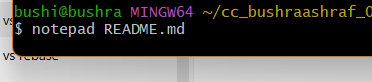
****

We attempted to push, and Git rejected the push because the remote contains a conflicting commit.

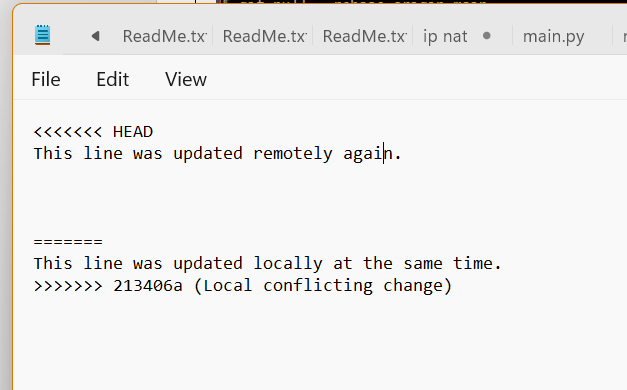
**Part E — Pull with rebase (this will pause and show the conflict):**



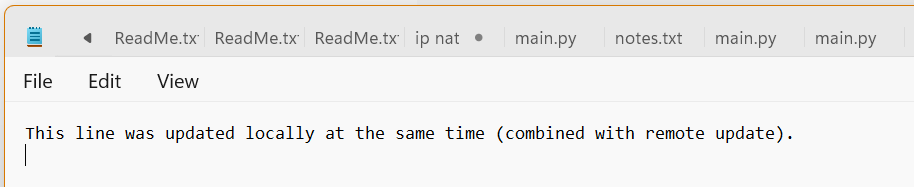
**Part F — Open the file and resolve the conflict manually:**

****

We will see a conflict:

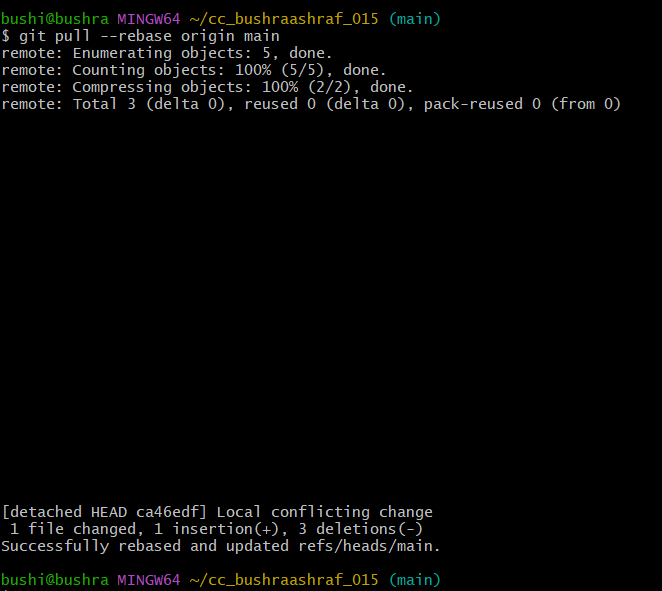


Edit the file to produce the final content you want.



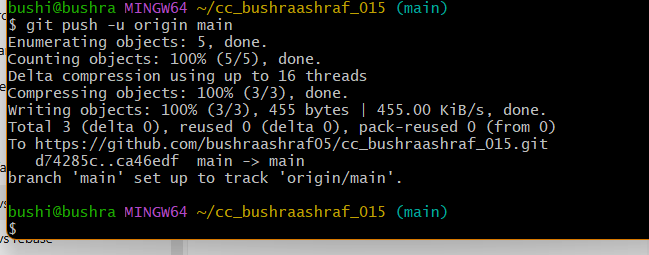
What we have done: We manually resolved the conflict inside README.md by editing the file and removing conflict markers.

**Part G — Mark resolved and continue the rebase**

****

We marked README.md resolved (git add) and continued the rebase (git rebase --continue), finishing the rebase so our local commits are replayed on top of the remote commits.

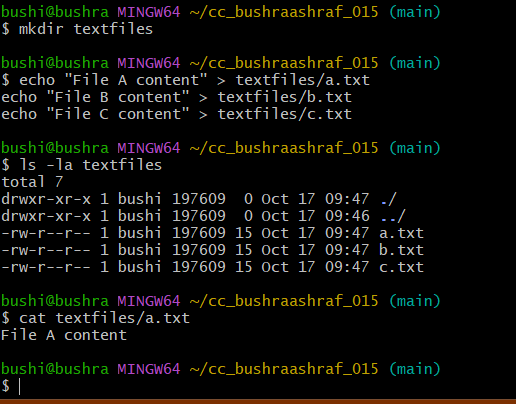
**Part H — Push the resolved result to remote**

****

We pushed the rebased, conflict-resolved branch to GitHub.

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

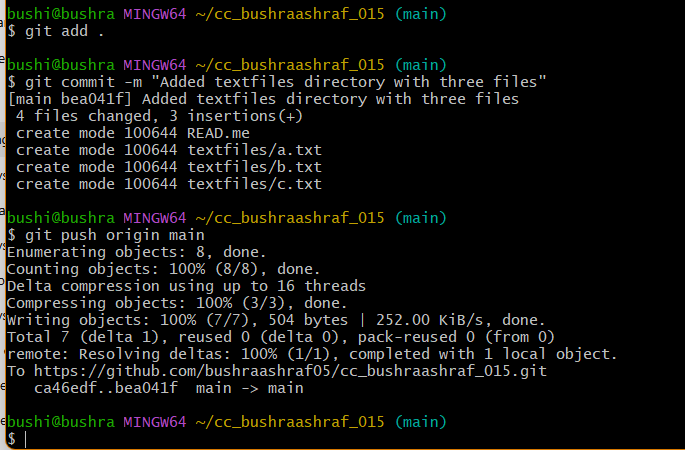
**Part A — Create the textfiles folder and the three files:**

****

**What we did:** Created a folder textfiles and three local text files (a.txt, b.txt, c.txt).

**Why:** Prepare files to demonstrate how .gitignore works and how to untrack already tracked files.

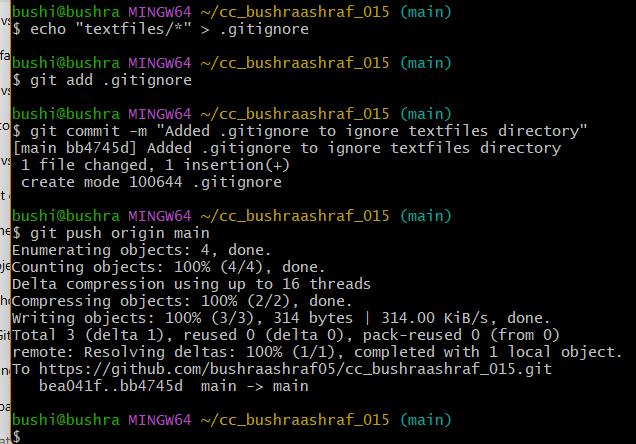
**Part B — Add, commit and push the new textfiles directory (make files tracked):**



**What we did:** Staged, committed, and pushed the textfiles folder so the files are now tracked by Git and present on the remote repository.

**Why:** Demonstrates that once committed and pushed, files appear on the remote and are tracked, which is necessary to show that .gitignore alone won’t remove them.

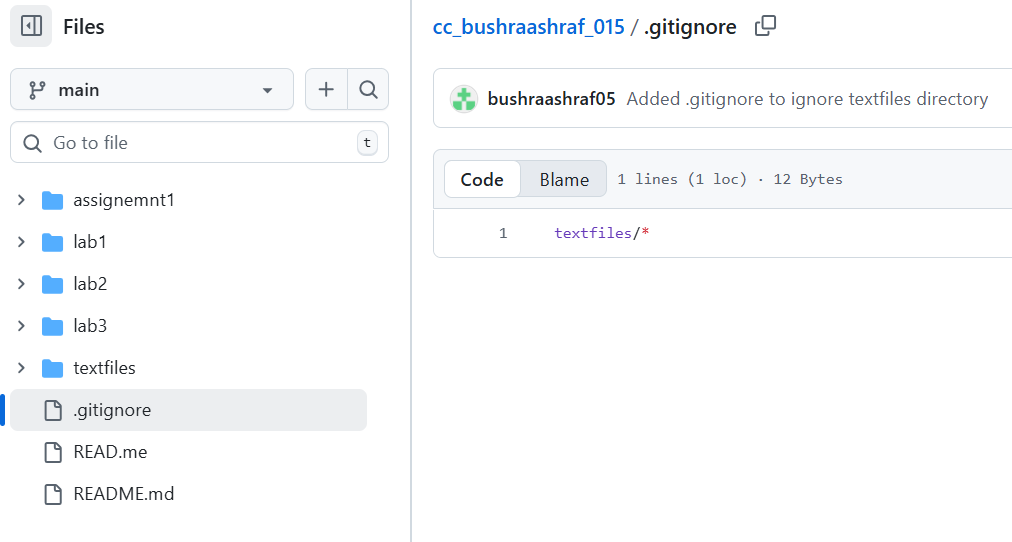
**Part C — Create and commit .gitignore**



**What we did:** Added .gitignore with textfiles/\* and pushed it.

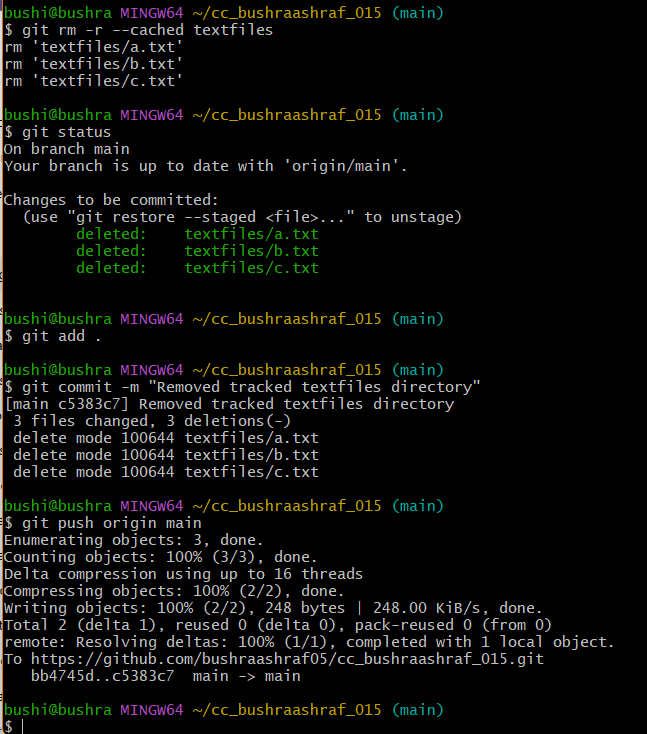
**Why:** Prevents **new** untracked files inside textfiles from being added in future. It does **not** automatically untrack files that are already committed.

**Part D — Inspect remote repository (confirm textfiles is still visible):**

****

**What we did:** Verified the remote still contains textfiles.

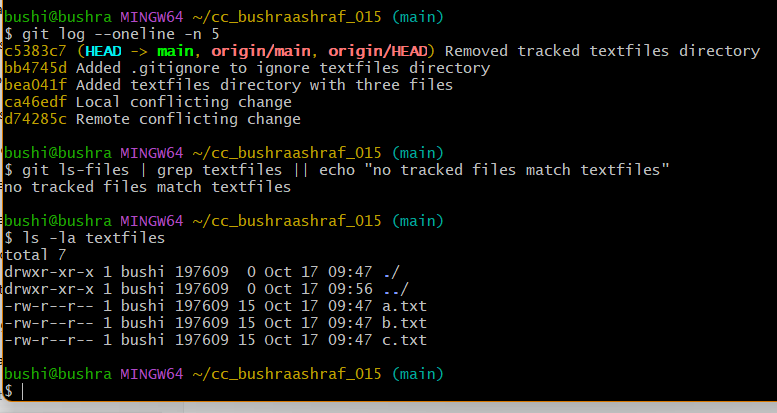
**Part E — Remove the files from Git tracking but keep them locally**

****

**What we did:** Ran git rm -r --cached textfiles to untrack the files, then committed and pushed that change.

**Why:** This removes the files from the repository (and from the remote) but keeps them on your local machine. Now .gitignore will prevent them from being re-added accidentally.

**Part F — Confirm remote no longer has textfiles and final checks**

****

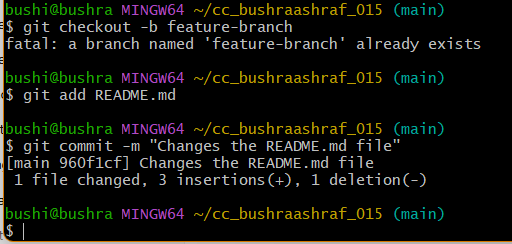
**What we did:** Verified the remote does not contain textfiles anymore and that the .gitignore prevents re-adding those files by accident.

**Why:** Final confirmation: the files are untracked in Git but still available locally — this is the desired behavior when you want files present on your machine but not in the remote repo.

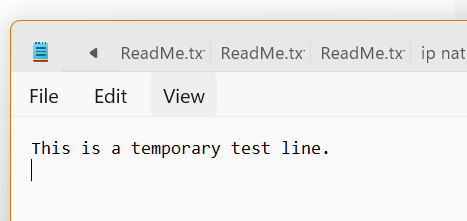
## Task 4 – Create Temporary Changes and Use git stash

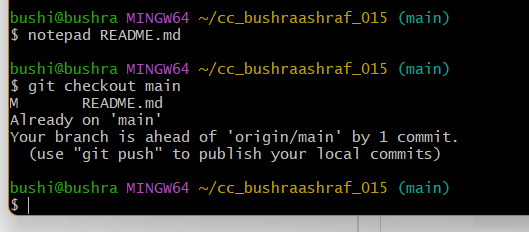
**Step A — Create a new feature branch**

**Step B — Stage and commit a small change**

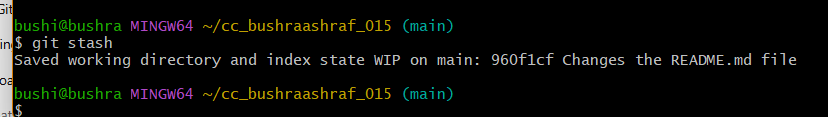
****

**Step C — Edit the file but don’t commit**

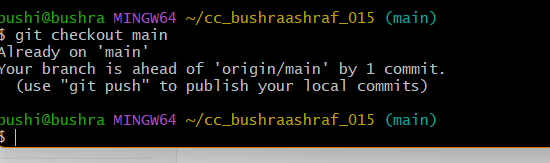
****

****

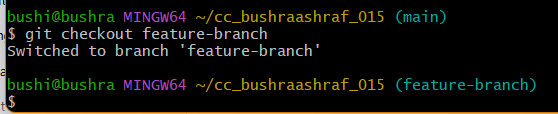
**Step D — Save your unfinished work with stash**

****

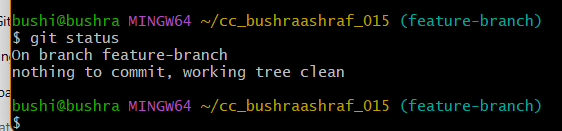
**Step E — Switch branches safely**

****

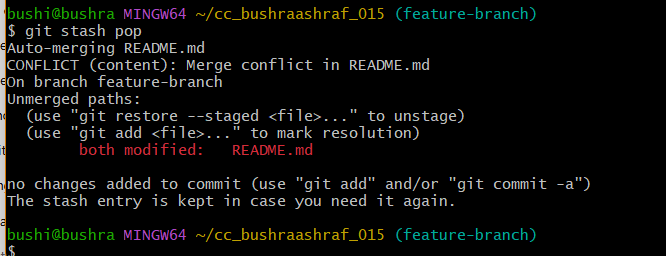
**Step F — Return to your feature branch**

****

**Step G — Check your current status**

****

**Step H — Restore your stashed changes**



## Task 5 – Checkout a Specific Commit Using git log:

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

## 

## 

## Part B: ****Add another change and commit again:****

## **Edit the same file again** and **add a different/new line** (e.g., Second test line.). Save the file.

## 

## 

## Part C — View history before any reset

## 

## Part D — Confirm file currently has both changes

## 

## Part E — Perform a soft reset (keeps changes in working directory / staging)

## 

## Part F — Check commit history after soft reset

## 

## ****PART G: Verify changes in the file:****

## 

## Part H: ****Check git status:****

## 

## PART I: ****Perform commit****

## 

## Part J: ****Perform a hard reset (discards changes completely):****

## 

## ****PART H: Check commit history after hard reset:****

## 

## ****PART I: Verify changes in the file:****

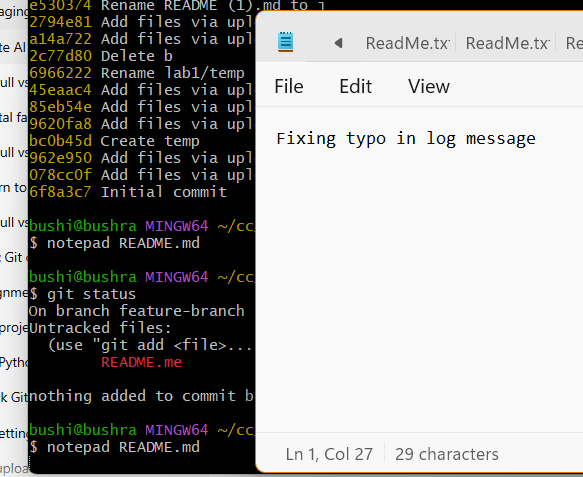
## 

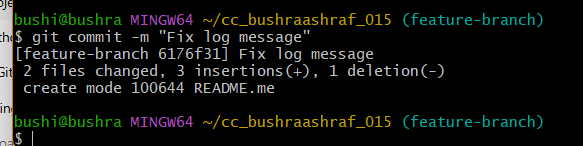
## ****Check git status:****

## 

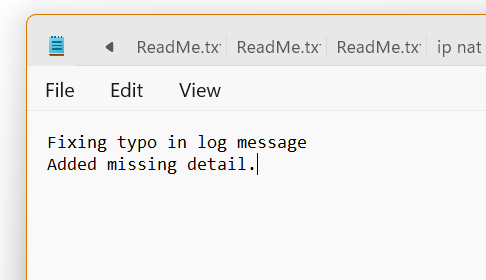
## Task 7 – Amending the Last Commit

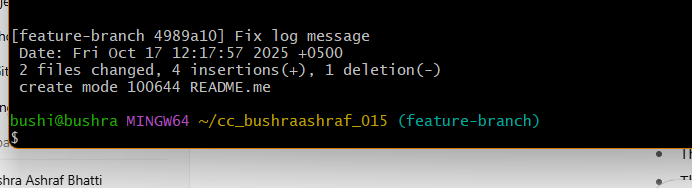
1. **Make a small change in any file.**

****

****

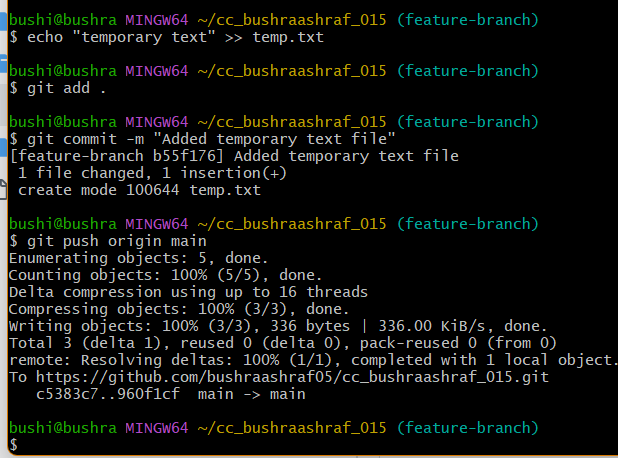
**Realize you forgot another file and fix it. Update it now:**

****

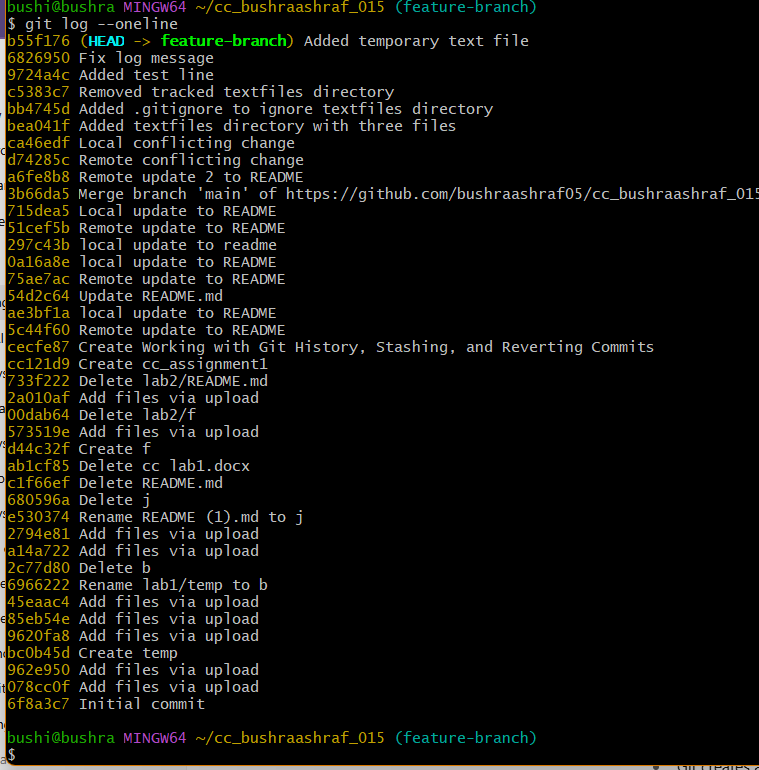
****

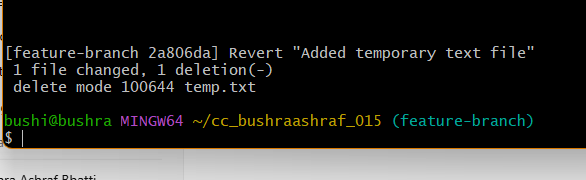
## Task 8 – Reverting a Commit (Safe Undo on Remote Branch)

**Make a change and commit it:**

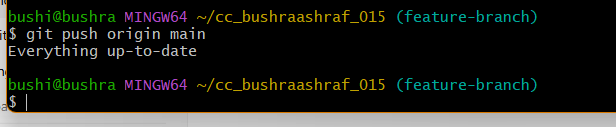
****

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

****

****

**Push the revert commit to remote**

****

## Task 9 – Force Push (With Caution)

## Part A — Create a new branch

## 

## Part B — Make and commit a small change:

## 

## Part C — Push the new branch to remote

## 

## Part D — Perform a hard reset (locally):

## 

## Part E — Try to push normally (will be rejected)

## 

## Part F — Force-push to overwrite remote history (use with caution)

## 

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

## Fork and open codepsace:

## 

## 

## ****Start Gitea with Docker Compose:****

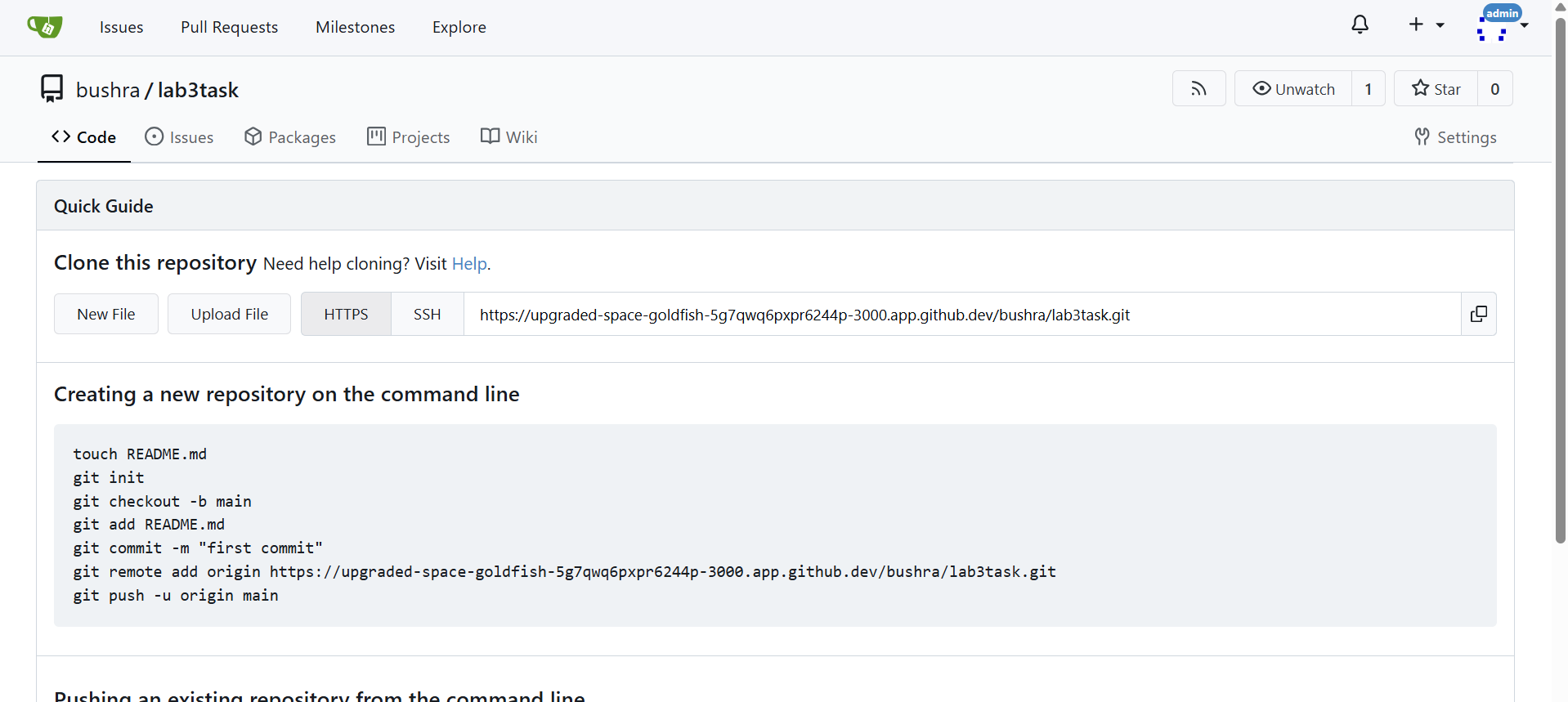
## 

## 

## 

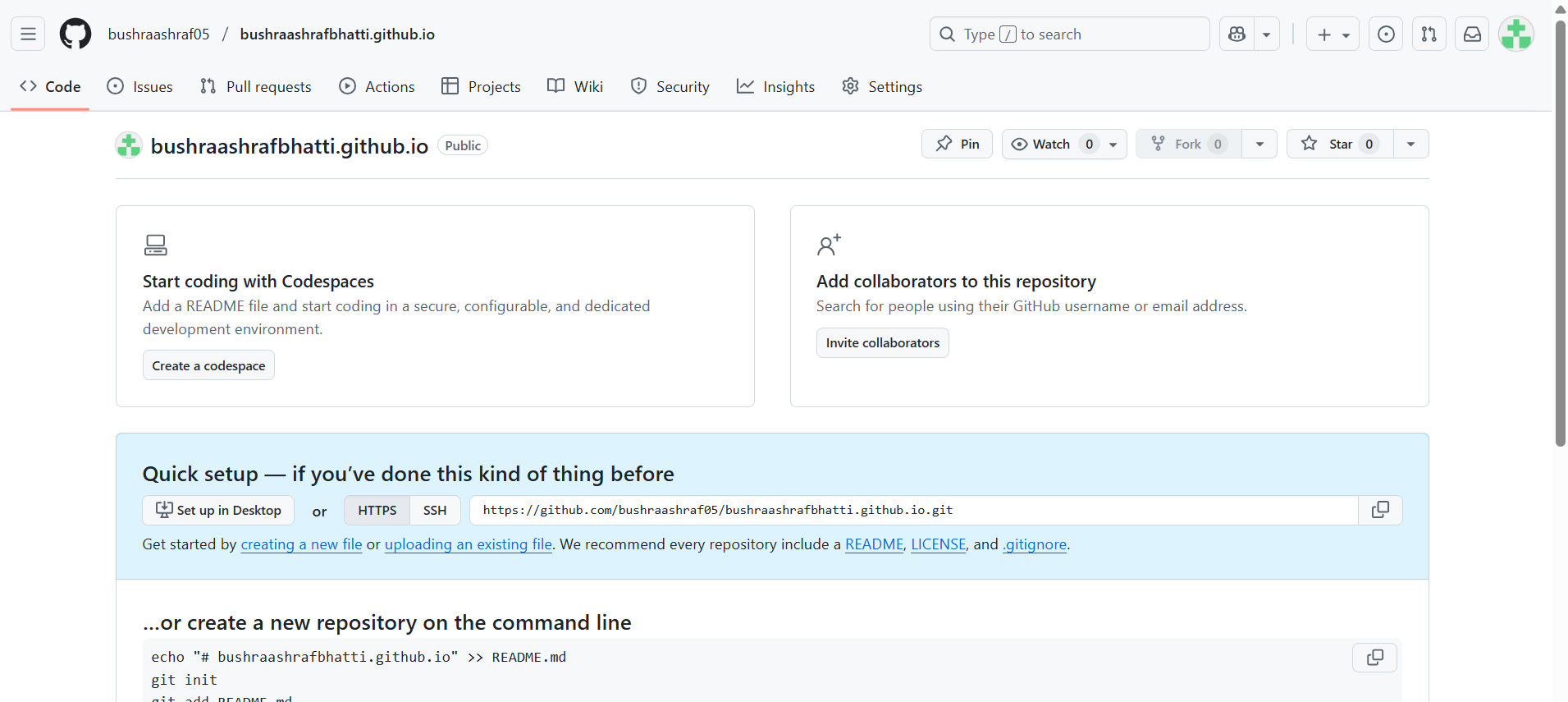
## 

## ****Create a New Repository in Gitea:****

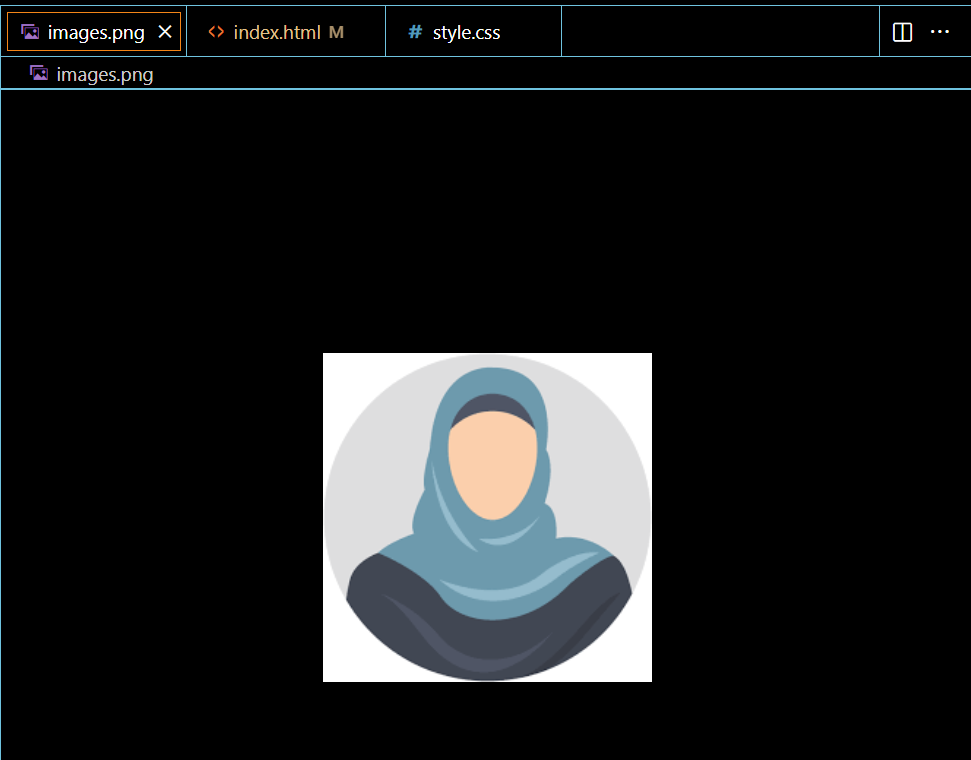


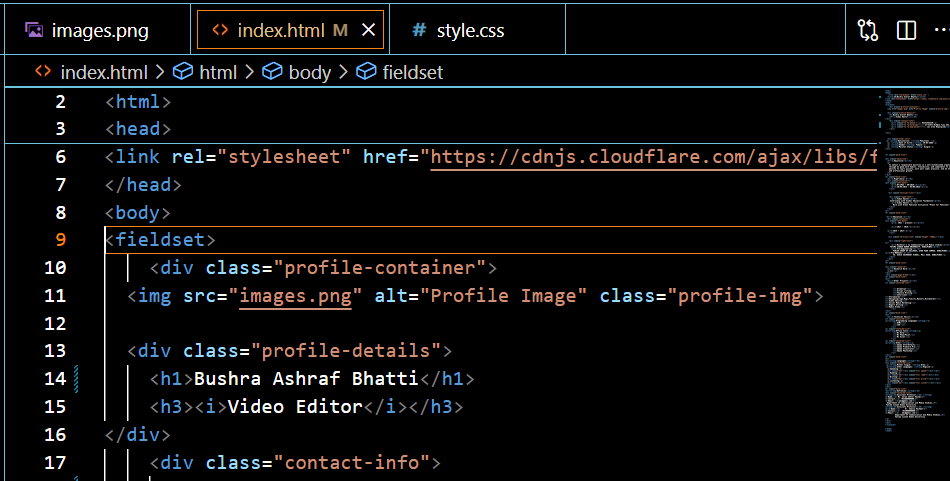
## Task 11 – Creating a GitHub Pages Portfolio Site

**step 1: create the public repository with the name “bushraashrafbhatti.githib.io:**



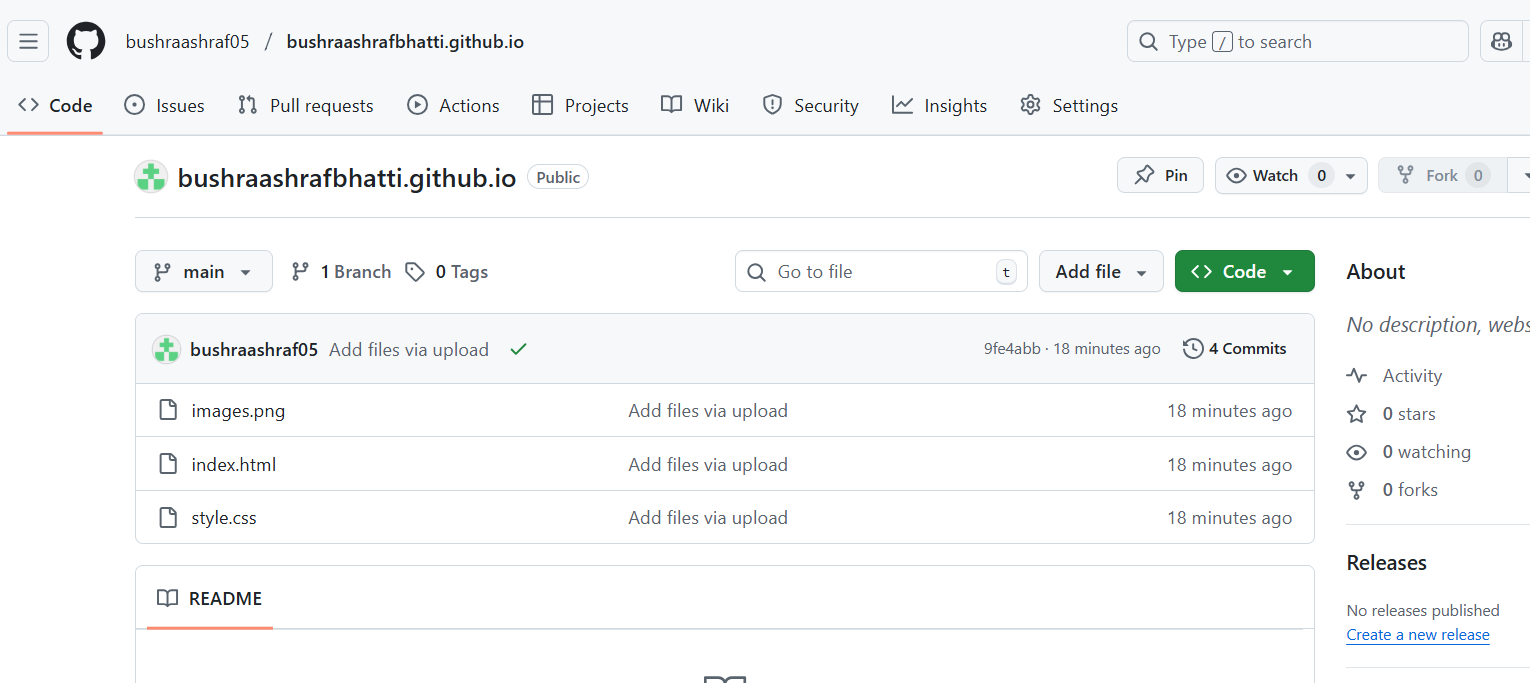
**Three vs code files:**



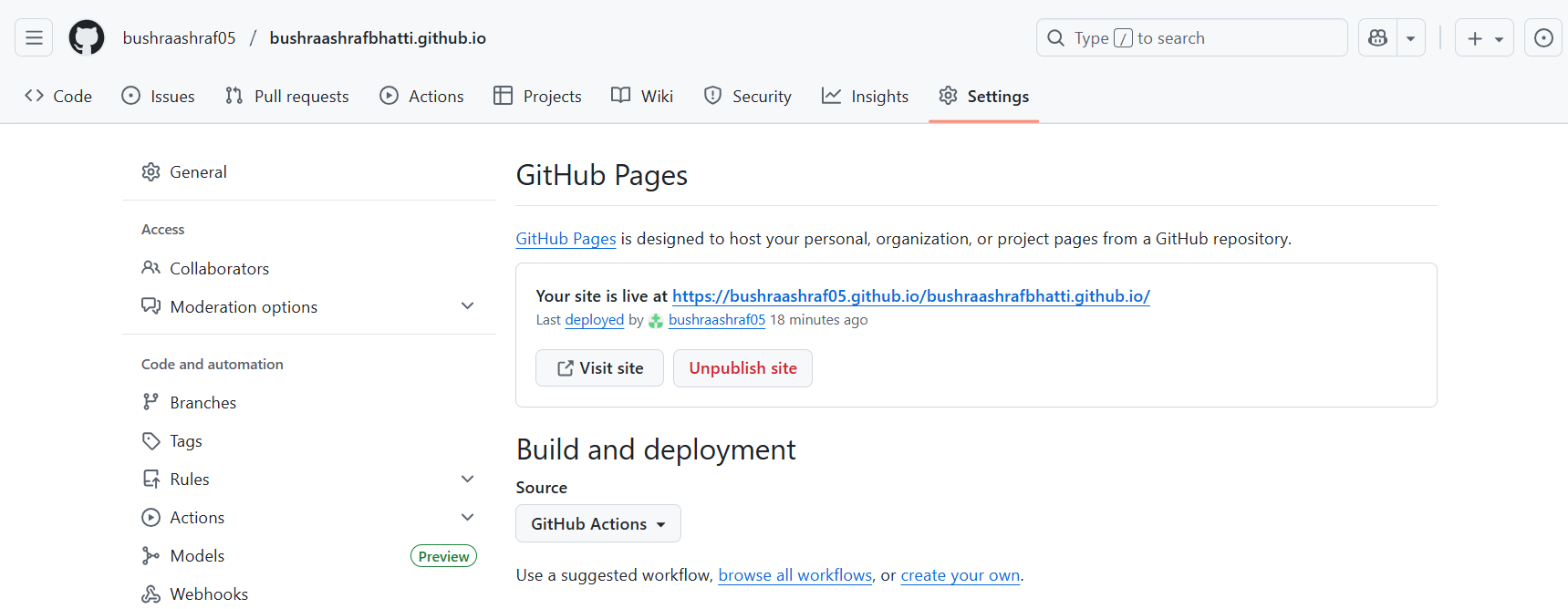




**Step 2: Push the files to the repository through codespace:**



**Step 3: Enable GitHub Pages in your repository settings if not automatically enabled. Publish your site and share the link.**



**Step 4:**

Link: <https://bushraashraf05.github.io/bushraashrafbhatti.github.io/>



## Exam Evaluation Questions:

## ****Question 1 – Local vs Remote Conflict Resolution:****

## 

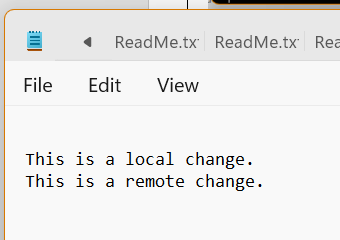
## 

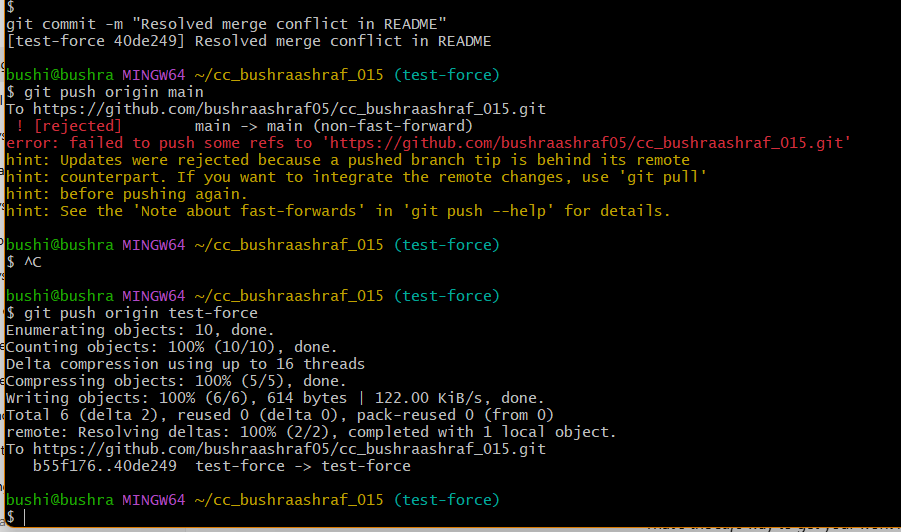
## 

## 

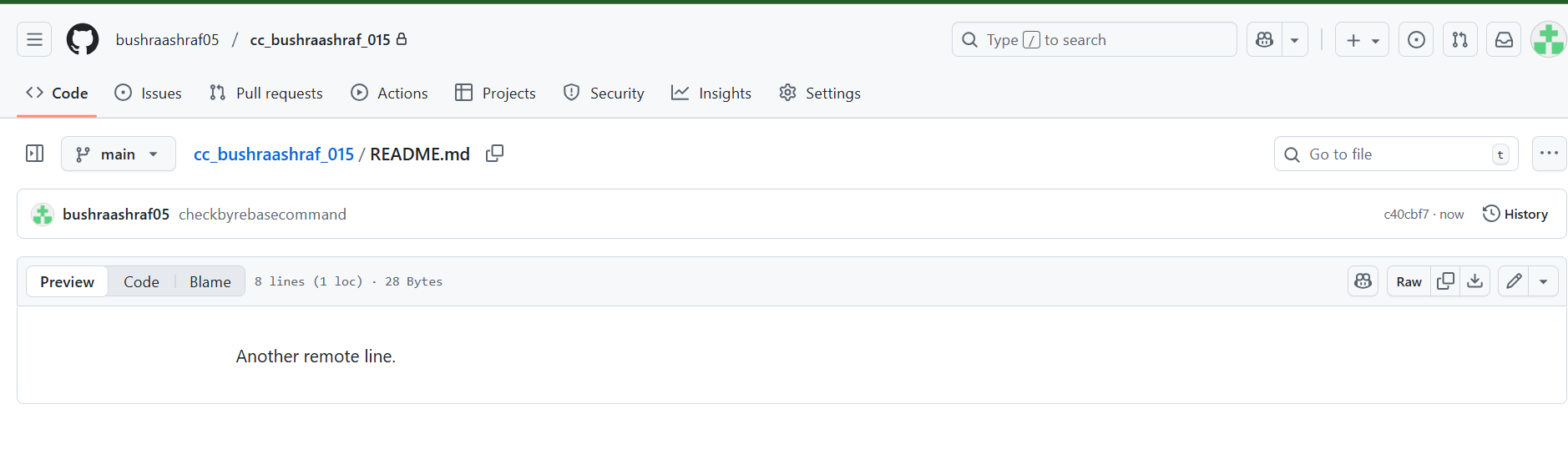
## 

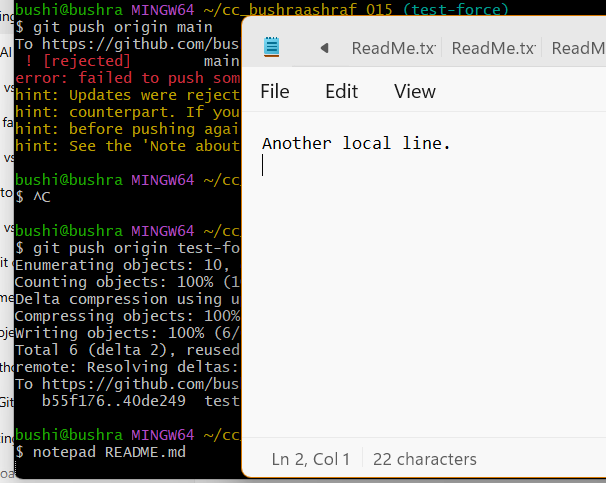
## 

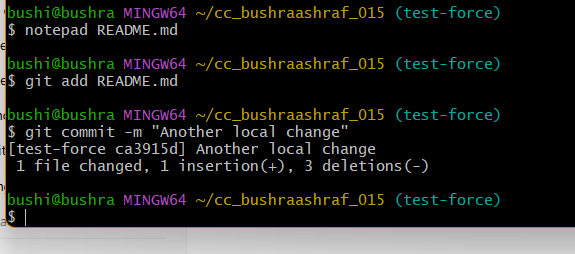


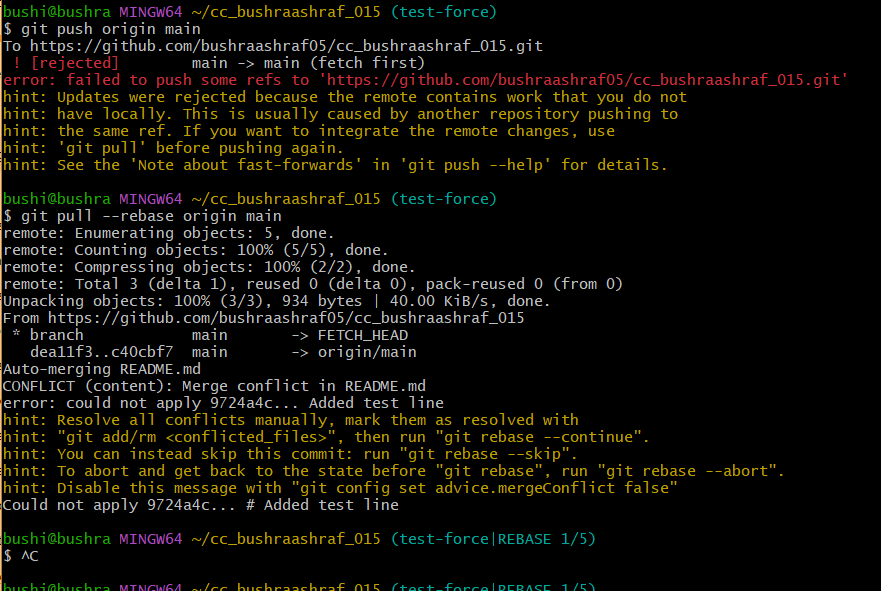


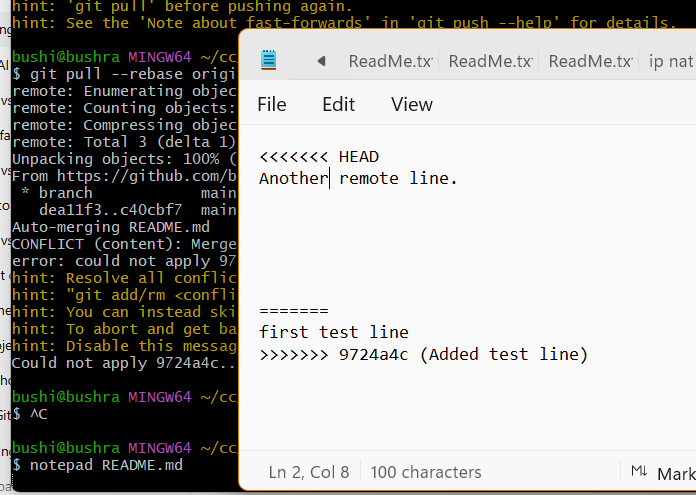
Resolve Using git pull --rebase (REBASE)



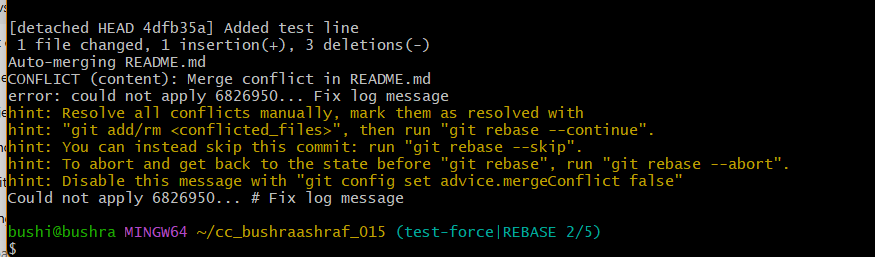








Manually fix it.



### 

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*