**COGNIZANT DIGITAL NURTURE PROGRAM 4.0**

**JAVA FSE DEEP SKILLING**

**SUPERSET ID : 6391159**

**NAME : SATHYA SHREE R**

**CLASS : ECE C**

**REG NO : 727822TUEC207**

**WEEK 8 HANDS ON EXERCISE – GIT HOL**

**Exercise 1 : Git Basic Setup and Notepad++ Integration Lab**

**PROBLEM STATEMENT :**

Set up Git, make Notepad++ the default editor, and push a file to a remote repository.

**CODE :**

**Git Setup :**

git --version

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

git config --list

**Integrate Notepad++ :**

# After adding Notepad++ path in Environment Variables

notepad++ # test if it opens

git config --global core.editor "notepad++ -multiInst -nosession"

git config --list --global # verify editor

**Create Repo & File**

mkdir GitDemo

cd GitDemo

git init

ls -a # view hidden .git folder

echo "Welcome to Git" > welcome.txt

ls # check file

cat welcome.txt

git status

**Track & Commit**

git add welcome.txt

git commit # opens Notepad++ for commit message

git status

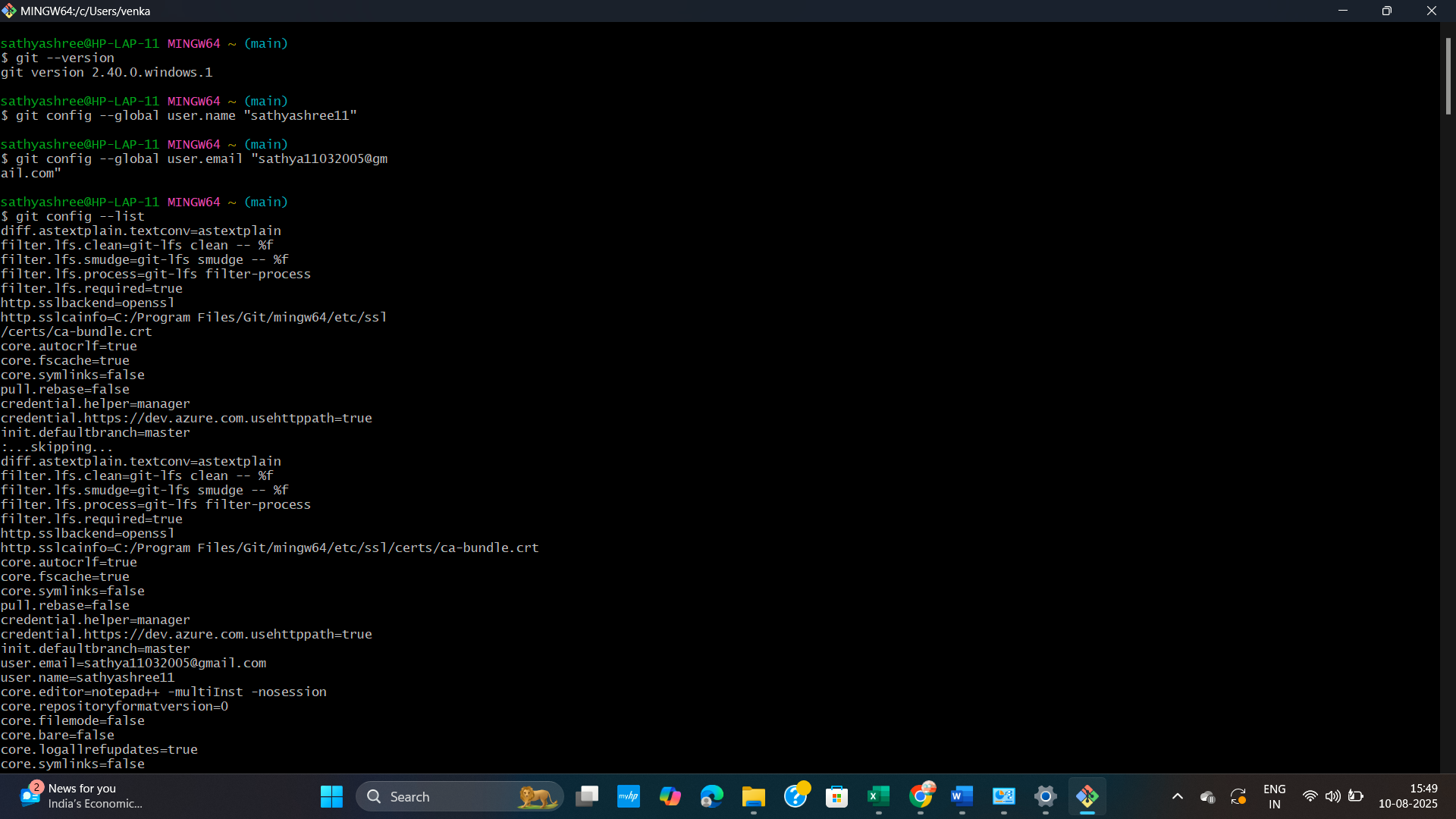
**Push to Remote**

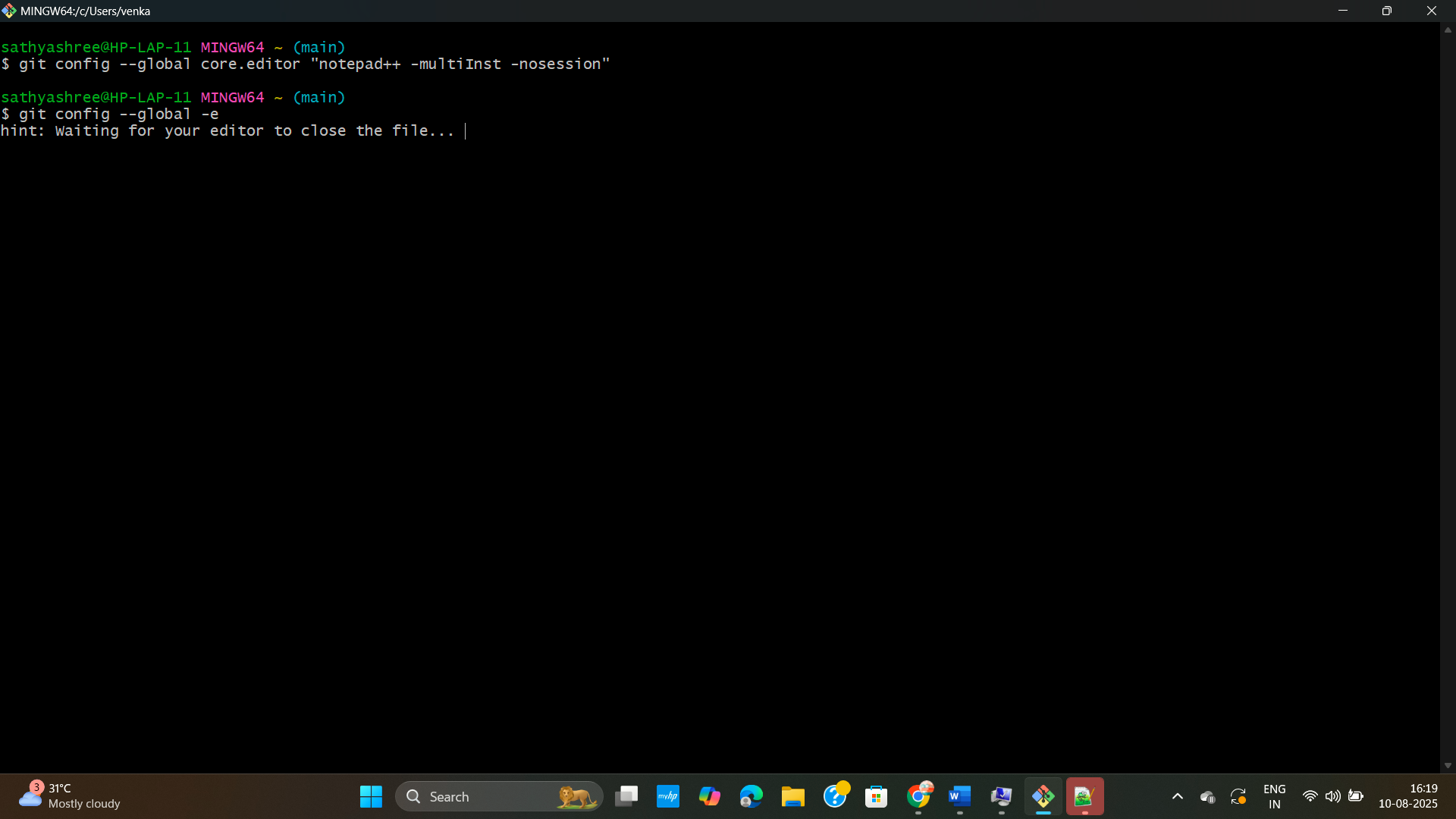
git remote add origin <repo-url>

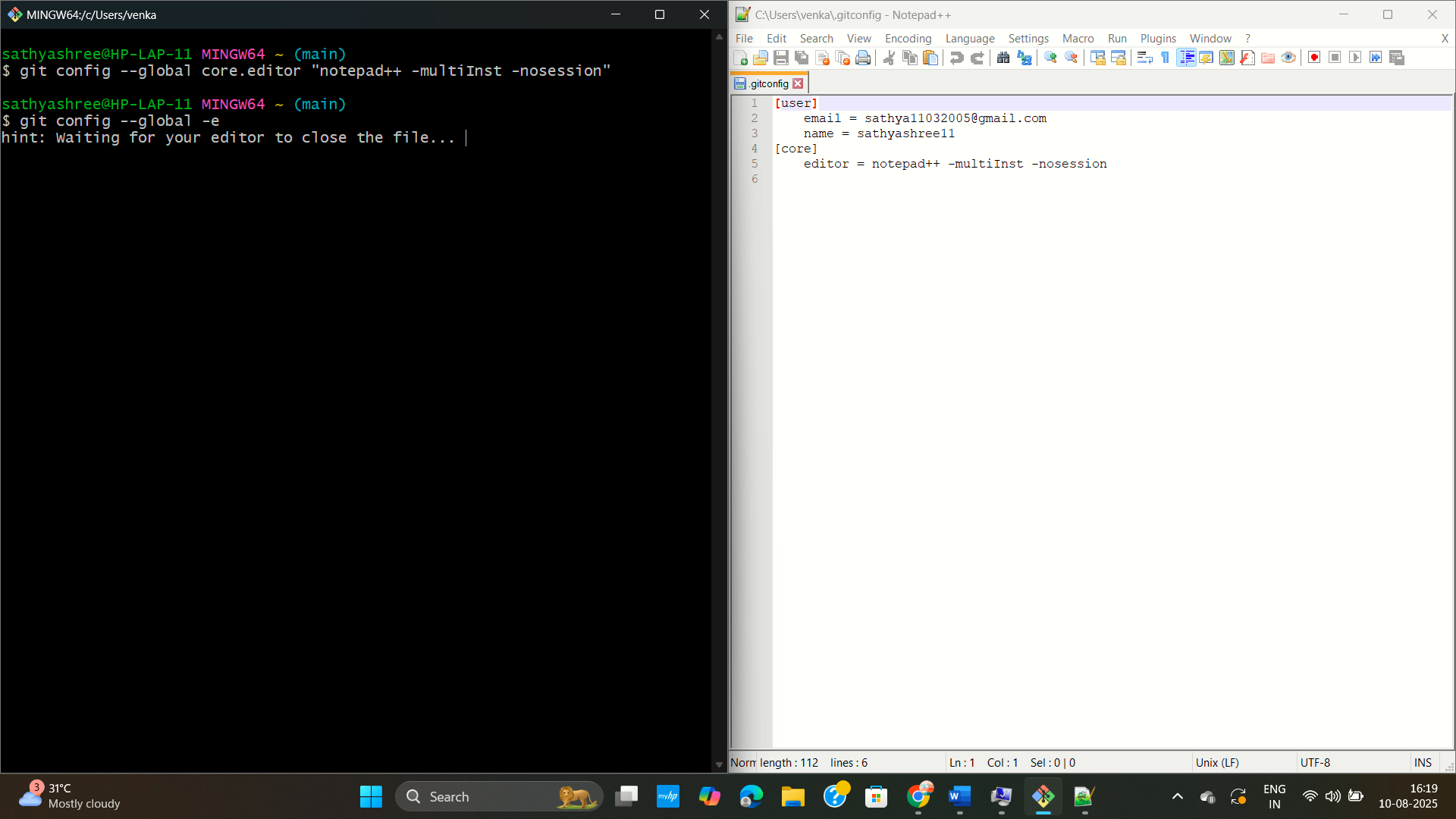
git pull origin master

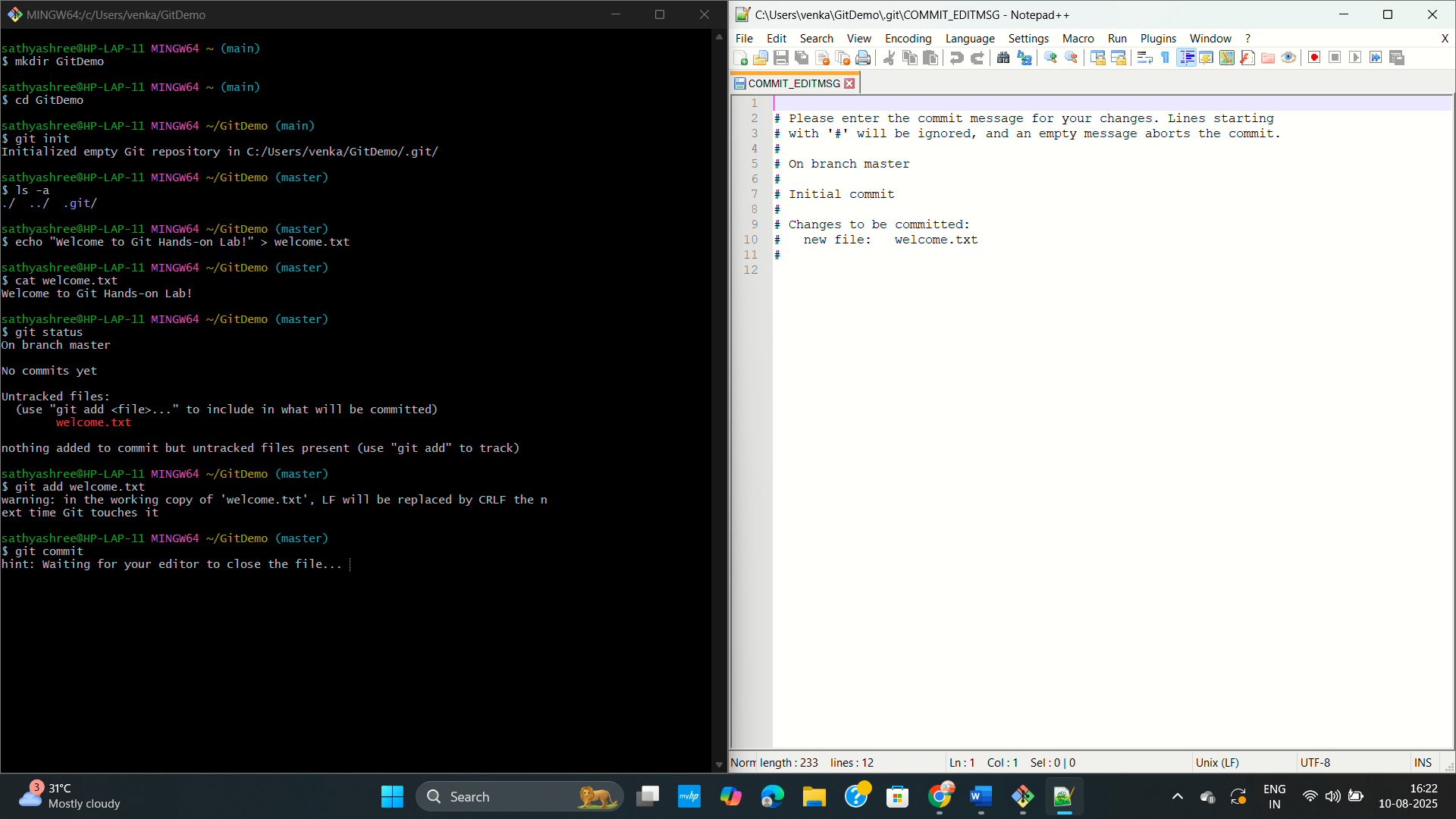
git push -u origin master

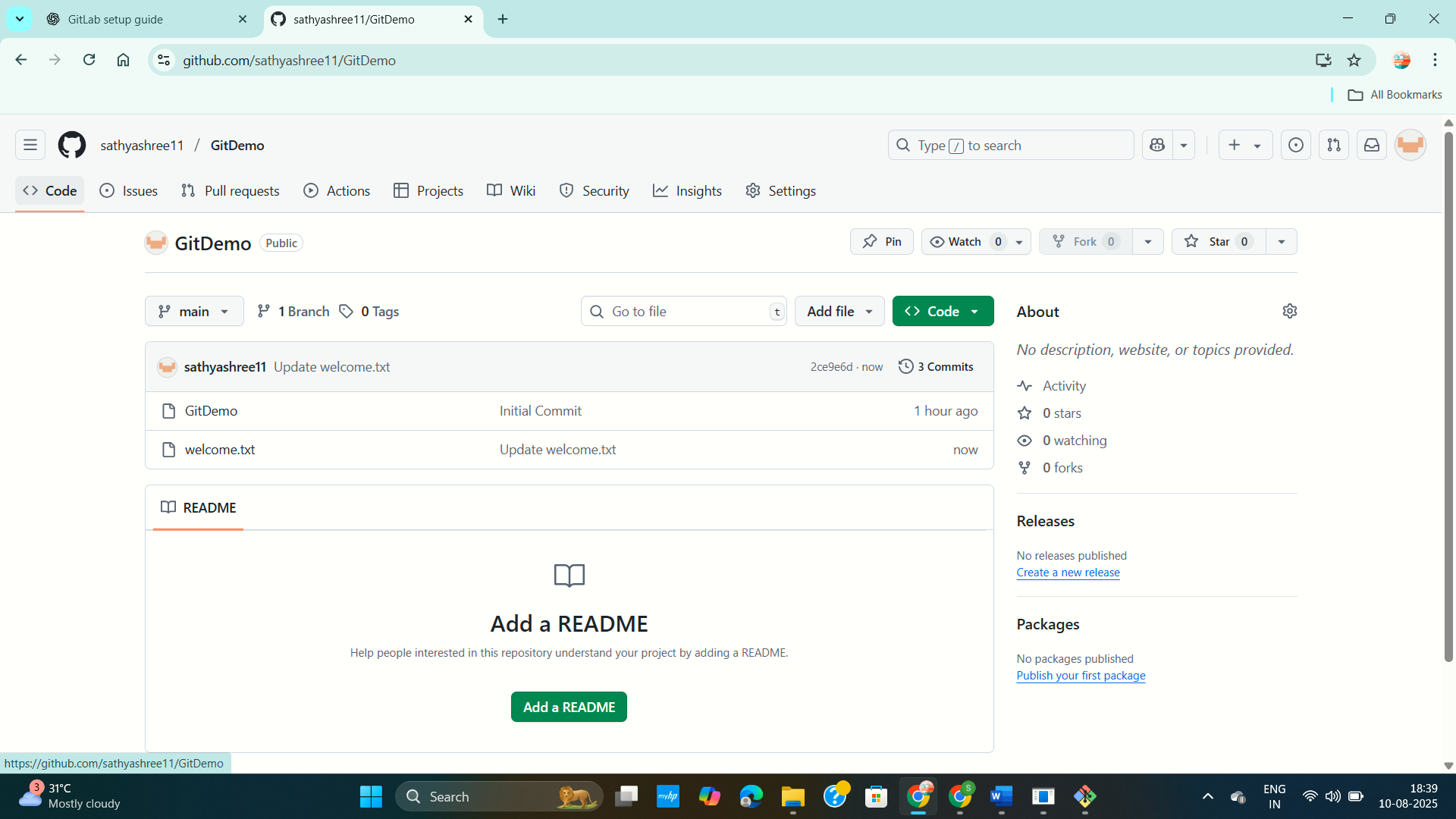
**OUTPUT :**











**Exercise 2 : Ignoring Unwanted Files in Git using .gitignore**

**PROBLEM STATEMENT :**

In a Git project, unwanted files like log files or log folders can clutter version control and cause unnecessary commits. You need to configure Git to ignore these files.

**CODE :**

**1. Go to your Git project folder**

cd "C:/Users/venka/Desktop/GitLabProject"

**2. Remove lock file if Git was stuck**

del .git\index.lock # (CMD)

# or in Git Bash:

rm -f .git/index.lock

**3. Create a .log file and logs folder**

echo Sample log content > error.log

mkdir logs

**4. Create or edit .gitignore**

# Add the following lines to it:

# \*.log

# logs/

notepad .gitignore # open in Notepad (Windows)

**5. Remove files from tracking if already tracked**

git rm --cached error.log

git rm --cached -r logs

**6. Commit and push changes**

git add .gitignore

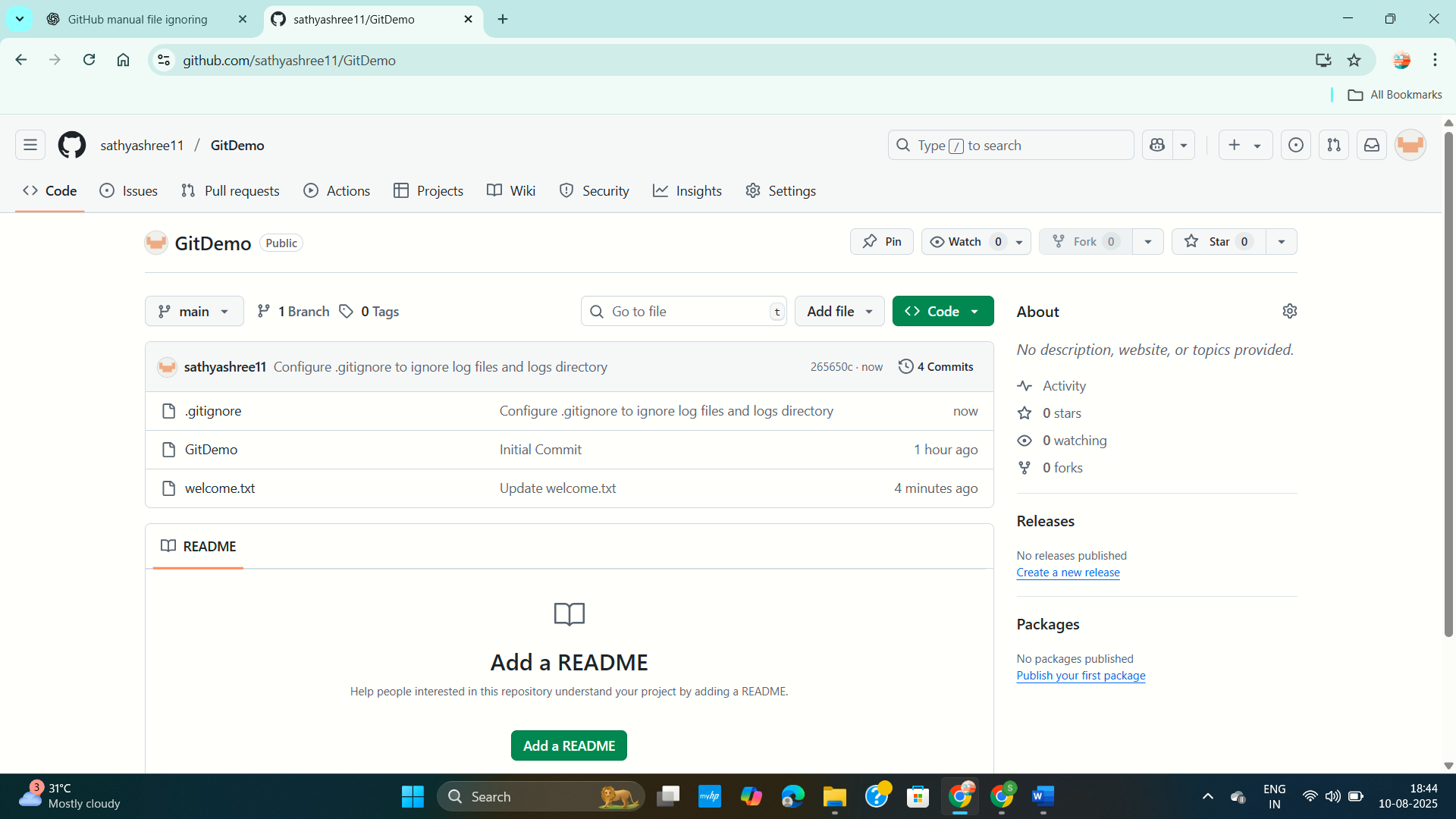
git commit -m "Ignore .log files and logs folder"

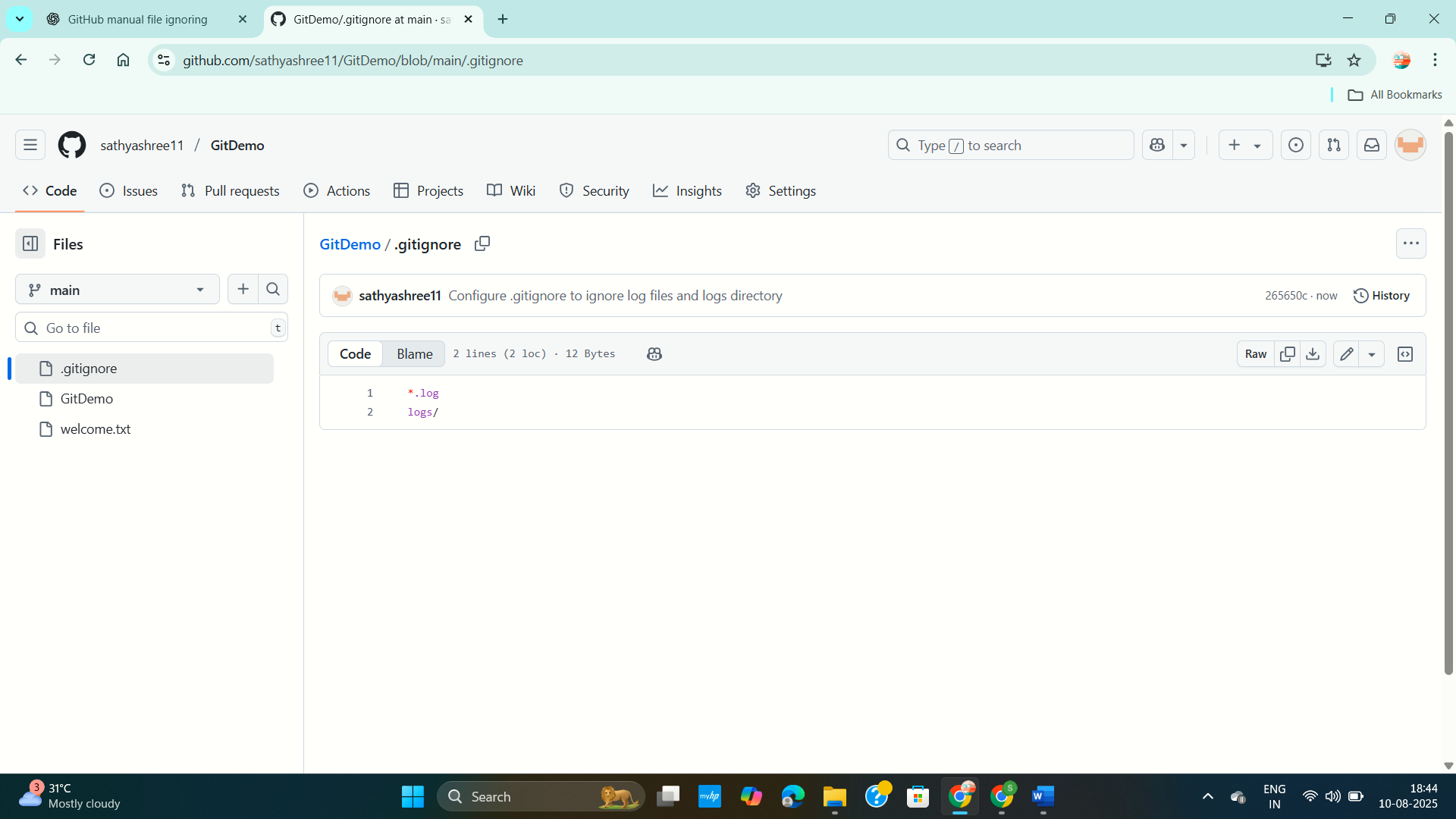
git push origin main

**7. Verify that files are ignored**

git status

**OUTPUT :**





**Exercise 3 : Branching and Merging in Git with GitLab Merge Requests**

**PROBLEM STATEMENT :**

To understand and practice branching and merging in Git, and to learn how to create branch and merge requests in GitLab, using both CLI and a visual merge tool (P4Merge).

**CODE :**

**Branching :**

**1. Create a new branch**

git branch GitNewBranch

**2. List local and remote branches**

git branch -a

**3. Switch to the new branch**

git checkout GitNewBranch

**4. Add new files and contents**

echo "Hello from GitNewBranch" > file1.txt

git add file1.txt

**5. Commit changes**

git commit -m "Added file1 in GitNewBranch"

**6. Check status**

git status

**Merging :**

**1. Switch to master**

git checkout master

**2. Show CLI differences**

git diff master GitNewBranch

**3. Show visual differences in P4Merge**

git difftool master GitNewBranch

**4. Merge branch into master**

git merge GitNewBranch

**5. View merge history**

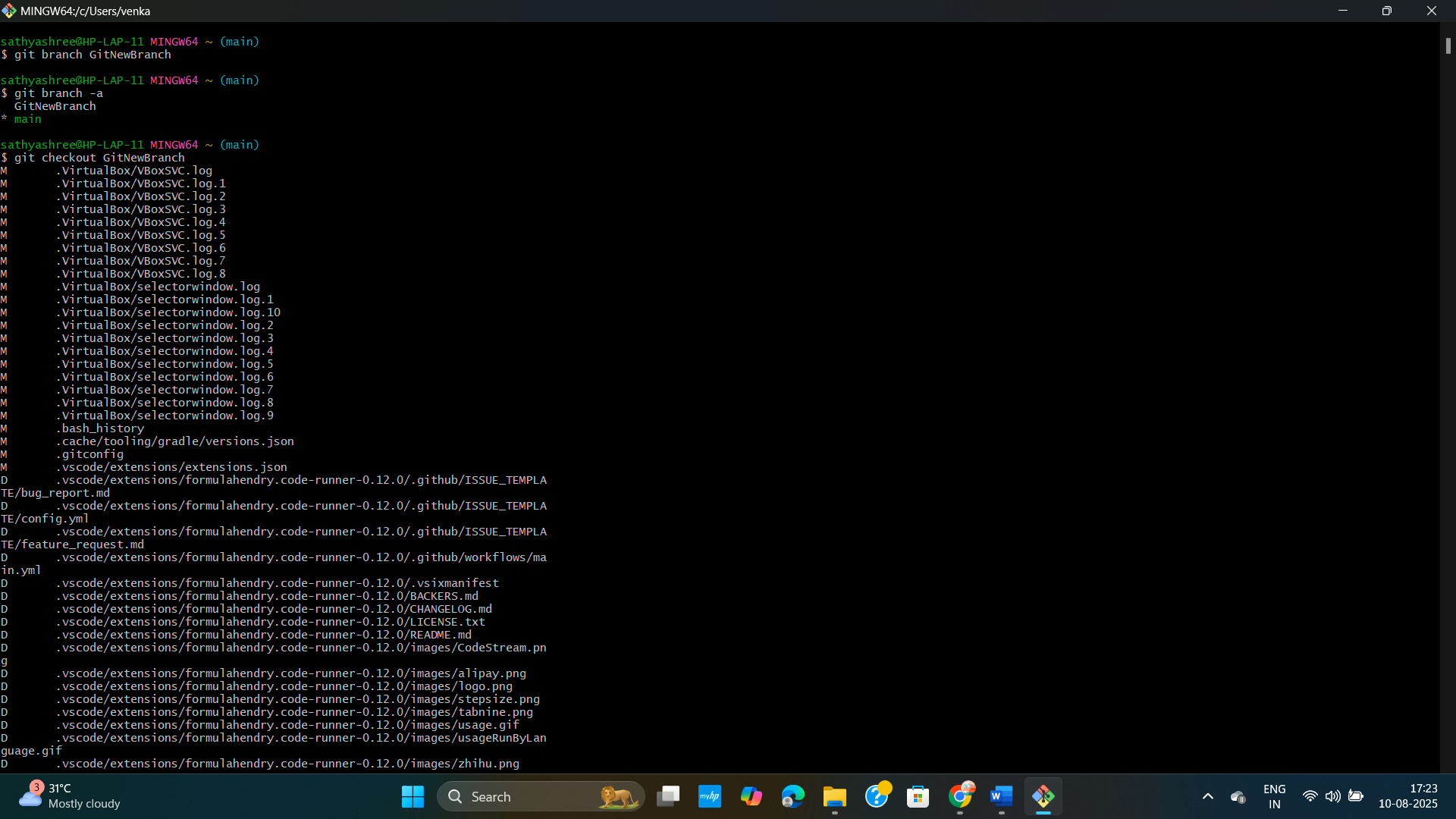
git log --oneline --graph --decorate

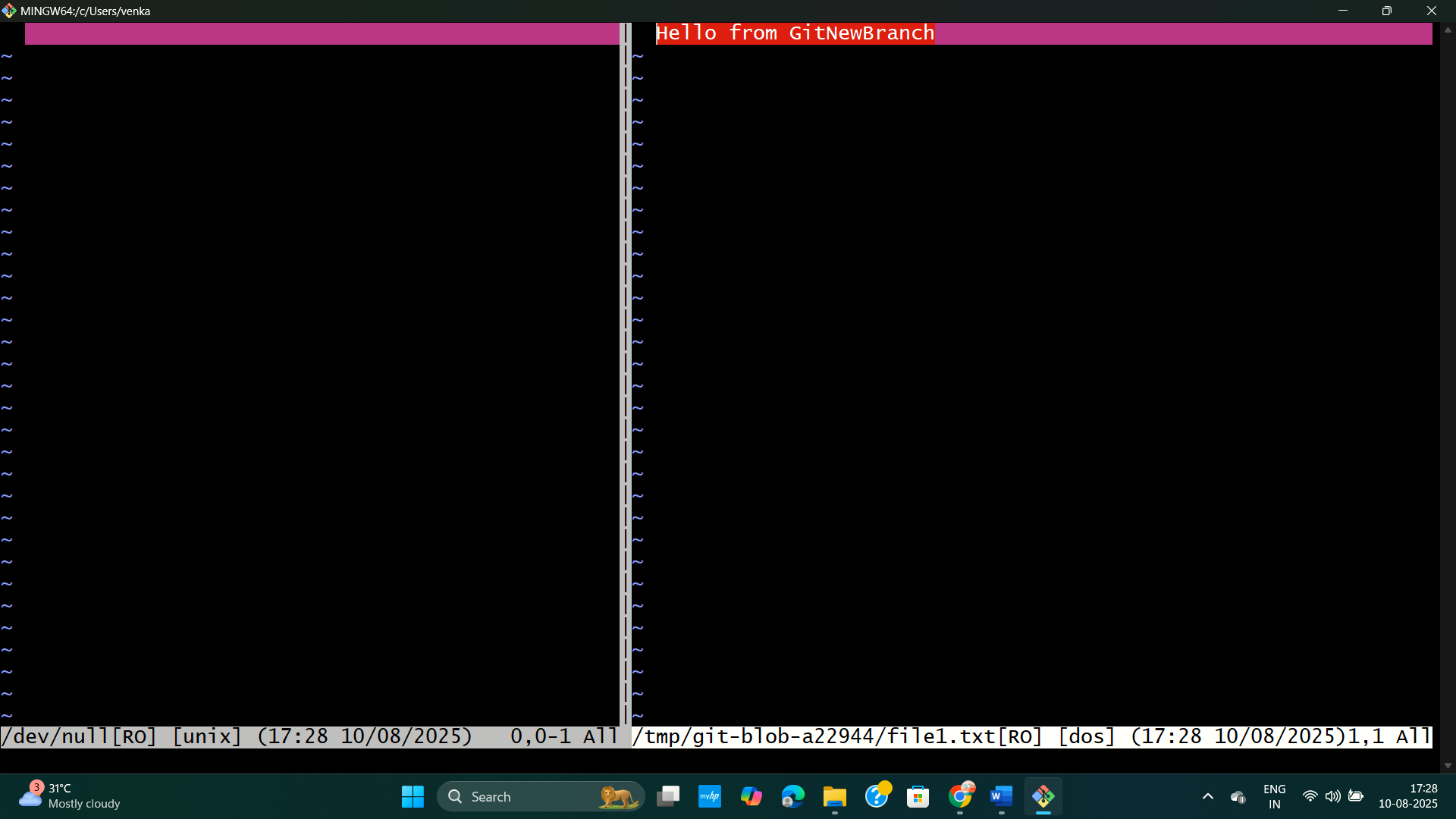
**6. Delete merged branch**

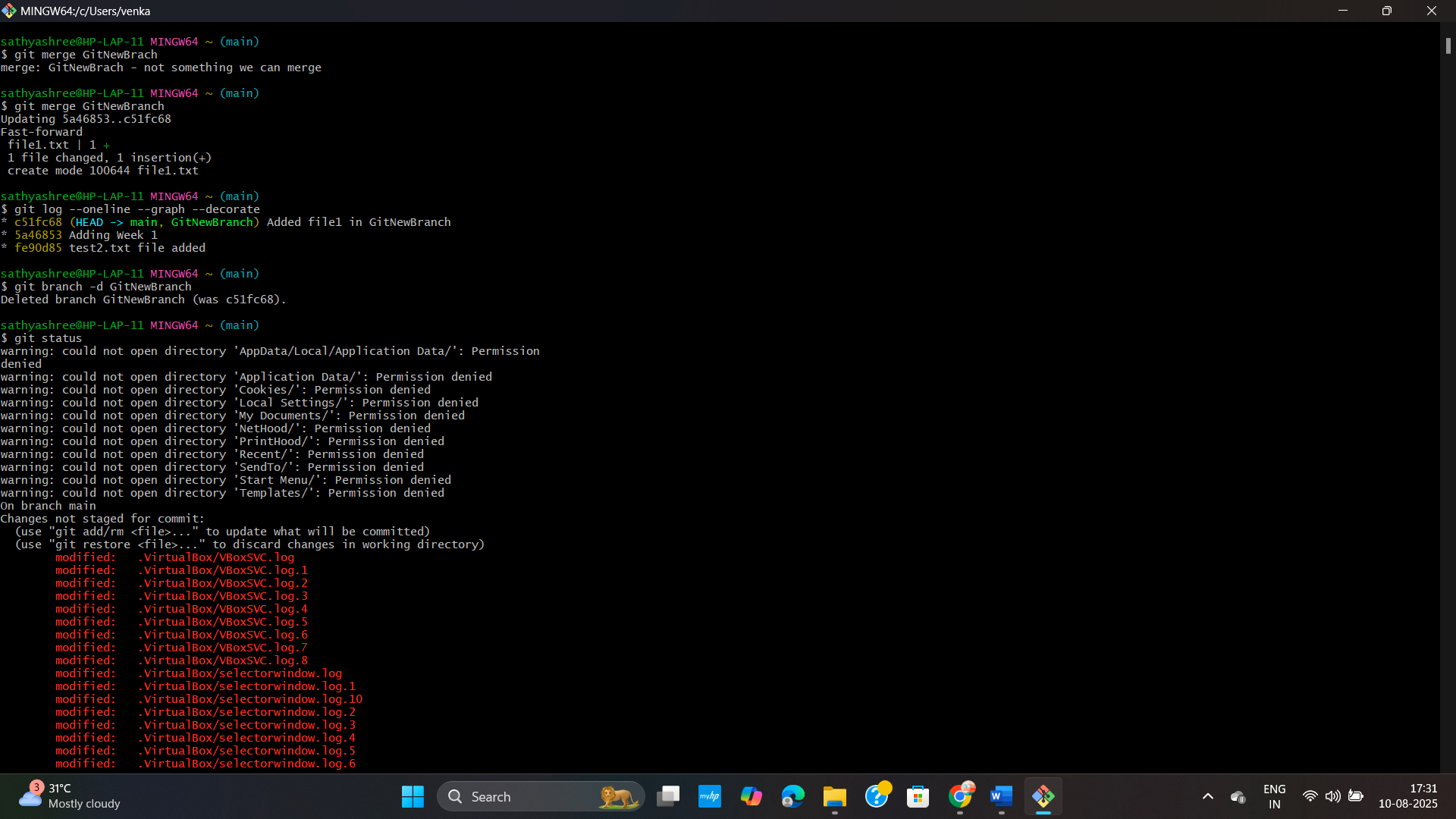
git branch -d GitNewBranch

git status

**OUTPUT :**







**Exercise 4 : Simulating and Resolving Merge Conflicts in Git**

**PROBLEM STATEMENT :**

When multiple developers work on the same file in different branches, merging them into the main branch (master) may lead to merge conflicts.  
 Simulate a scenario where two branches modify the same file differently and learn how to identify, resolve, and commit the conflict.

**CODE :**

**Resolving Merge Conflicts in Git :**

**1. Verify master is clean**

git status

**2. Create a new branch**

git checkout -b GitWork

**3. Add file hello.xml in branch**

echo "<greeting>Hello from GitWork branch</greeting>" > hello.xml

git add hello.xml

git commit -m "Add hello.xml in GitWork branch"

**# 4. Switch to master**

git checkout master

**# 5. Add different hello.xml in master**

echo "<greeting>Hello from Master branch</greeting>" > hello.xml

git add hello.xml

git commit-m "Add hello.xml in master branch"

**6. View branch history**

git log --oneline --graph --decorate --all

**7. Compare differences**

git diff GitWork

**# (Optional) Use P4Merge for visual diff if installed**

git mergetool

**8. Merge GitWork into master (will cause conflict)**

git merge GitWork

**9. Open hello.xml and resolve conflict manually:**

# Keep desired changes, remove conflict markers (<<<<<<, ======, >>>>>>)

**10. Mark file as resolved**

git add hello.xml

**11. Commit merge**

git commit -m "Merge branch 'GitWork' into master with conflict resolution"

**12. Add backup files to .gitignore (if any created by merge tool)**

echo "\*.orig" >> .gitignore

git add .gitignore

git commit -m "Add .gitignore for merge backup files"

**13. List all branches**

git branch

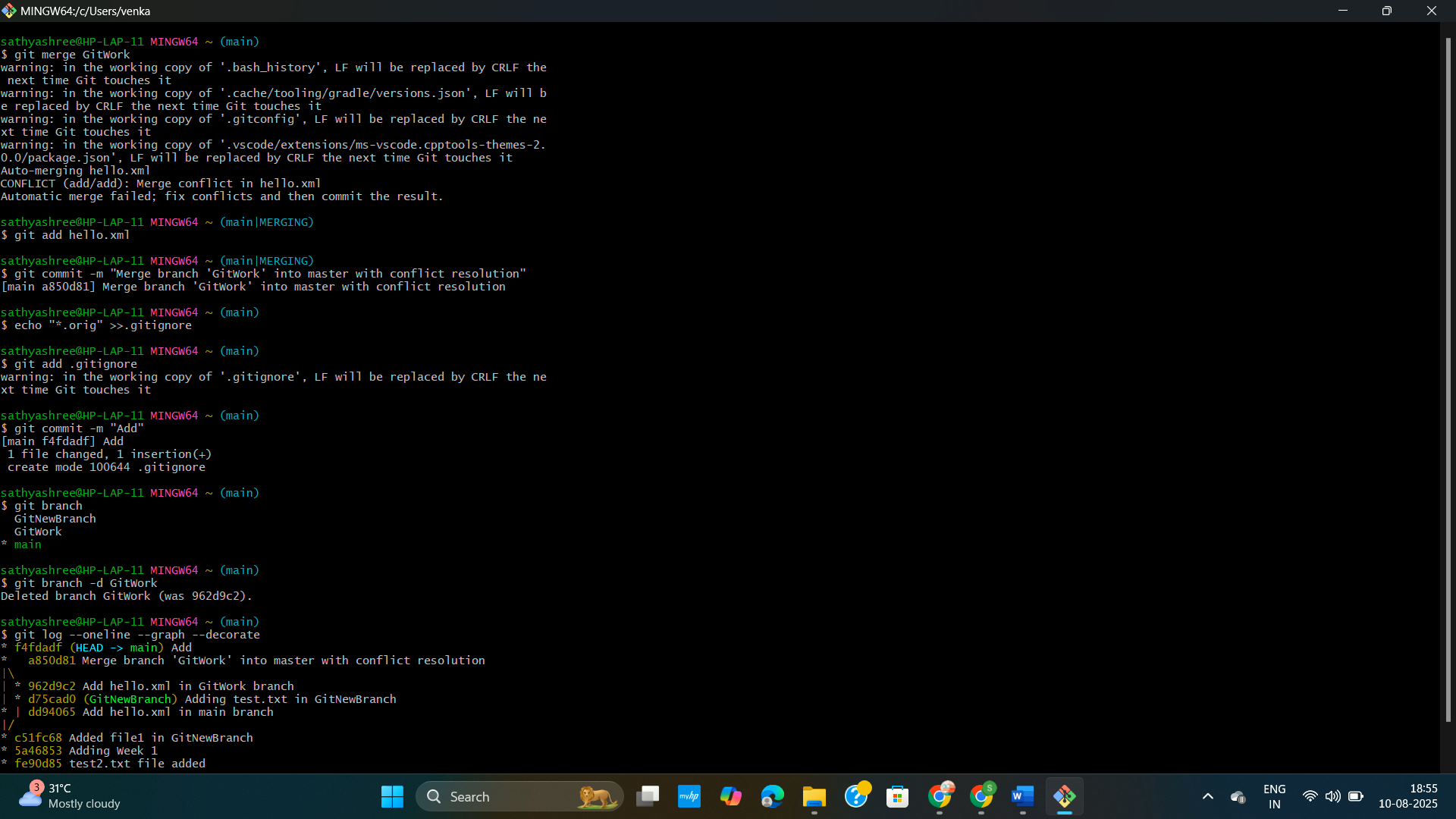
**14. Delete merged branch**

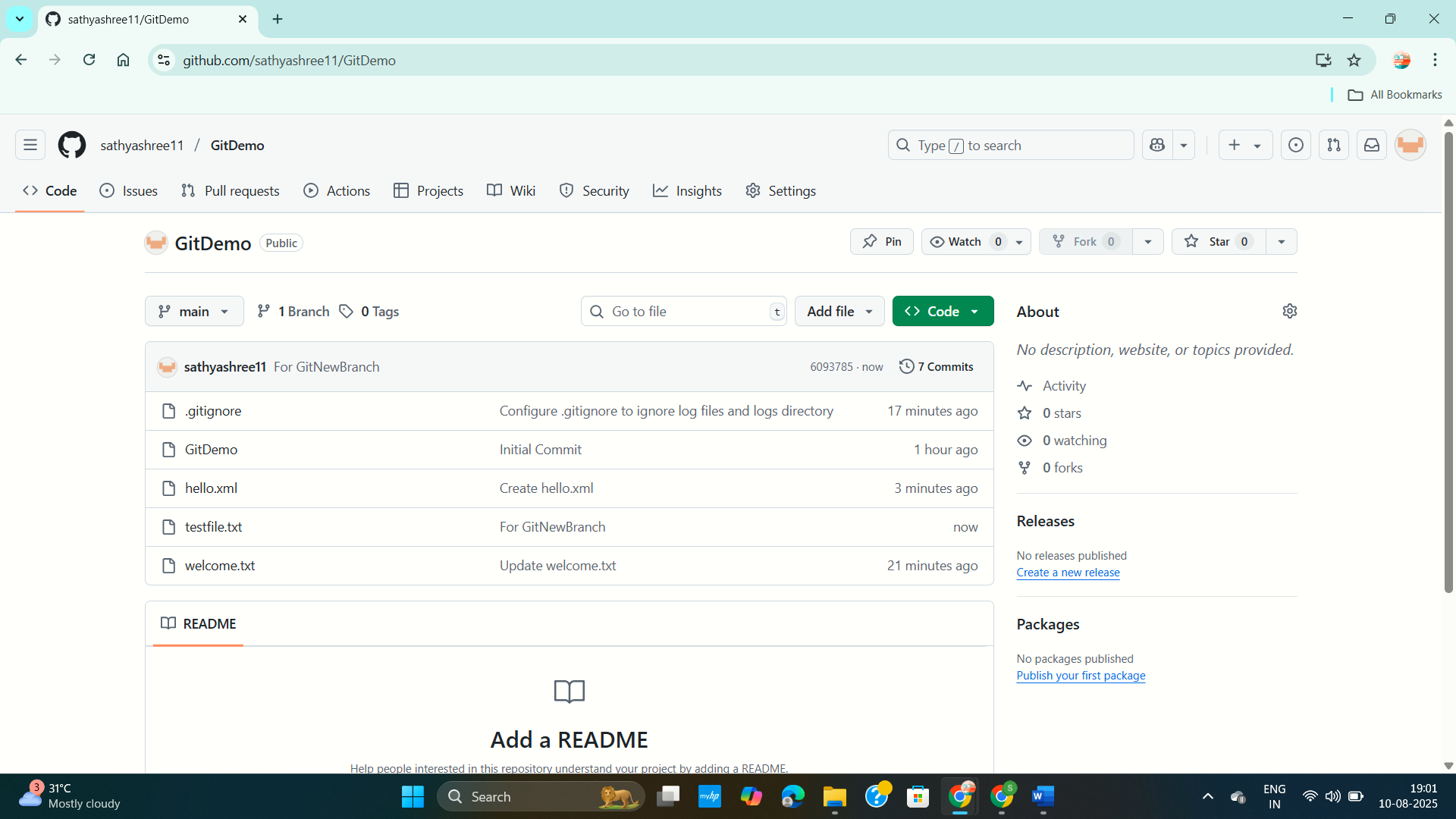
git branch -d GitWork

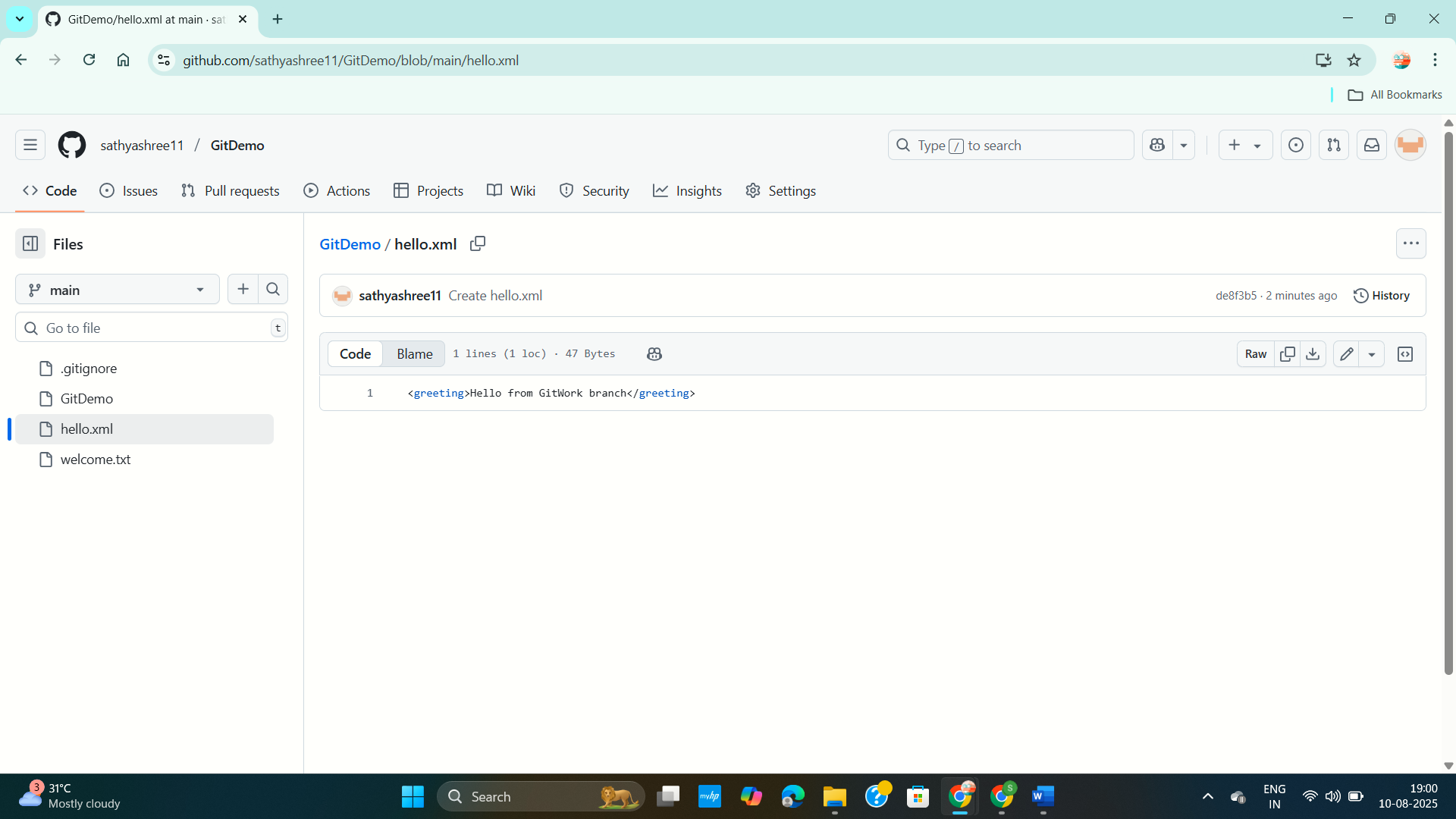
**15. View final commit graph**

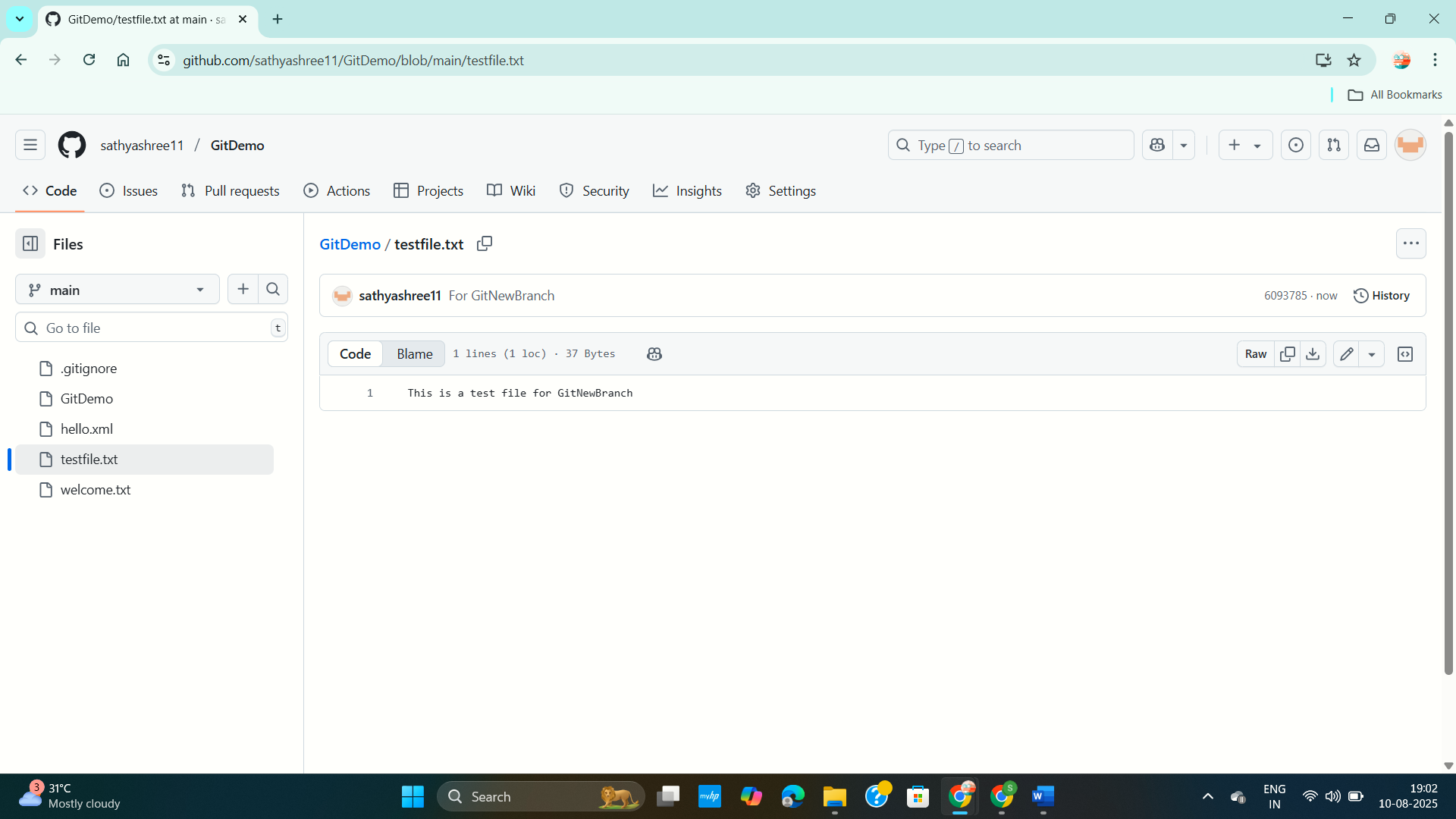
git log --oneline --graph –decorate

**OUTPUT :**









**Exercise 5 : Git Pull and Push Workflow Lab**

**PROBLEM STATEMENT :**

In collaborative development, it’s important to ensure that your local repository is synchronized with the remote repository before pushing changes. This lab demonstrates how to **verify repository state, pull the latest changes, and push local updates** back to the remote Git repository.

**CODE :**

**Cleaning Up and Pushing to Remote Git :**

**1. Verify if master is clean (no pending changes)**

git status

**2. List all available branches**

git branch -a

**3. Pull latest changes from remote master**

git pull origin master

**4. Push pending changes from "Git-T03-HOL\_002" to remote**

git push origin master

**5. Verify changes are reflected in remote repository**

# (Go to GitHub/GitLab web interface and check latest commit)

**OUTPUT :**

