**WEEK-8**

(Implemented in GITBASH)

**GIT**

**1. GIT-HOL-** Git Basics & Adding a File

Familiar with Git commands like git init, git status, git add, git commit, git push, and git pull.

In this hands-on lab, you will learn how to

* Setup your machine with Git Configuration
* Integrate notepad++.exe to Git and make it a default editor
* Add a file to source code repository

**Steps:**

**1. Install & Verify Git**

bash

CopyEdit

git --version

**2. Add Notepad++ to PATH for Git Bash**

bash

CopyEdit

export PATH=$PATH:"/c/Program Files/Notepad++"

**3. Configure Git**

bash

CopyEdit

git config --global user.name "Faiza-19"

git config --global user.email "faizahameed04@gmail.com"

git config --list

**4. Set Notepad++ as Git Default Editor**

bash

CopyEdit

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

git config --global -e

**5. Create Local Repository**

bash

CopyEdit

mkdir GitDemo

cd GitDemo

git init

**6. Add and Commit File**

bash

CopyEdit

echo "Welcome to Git" > welcome.txt

cat welcome.txt

git add welcome.txt

git commit -m "Add welcome.txt"

**7. Link to Remote & Push**

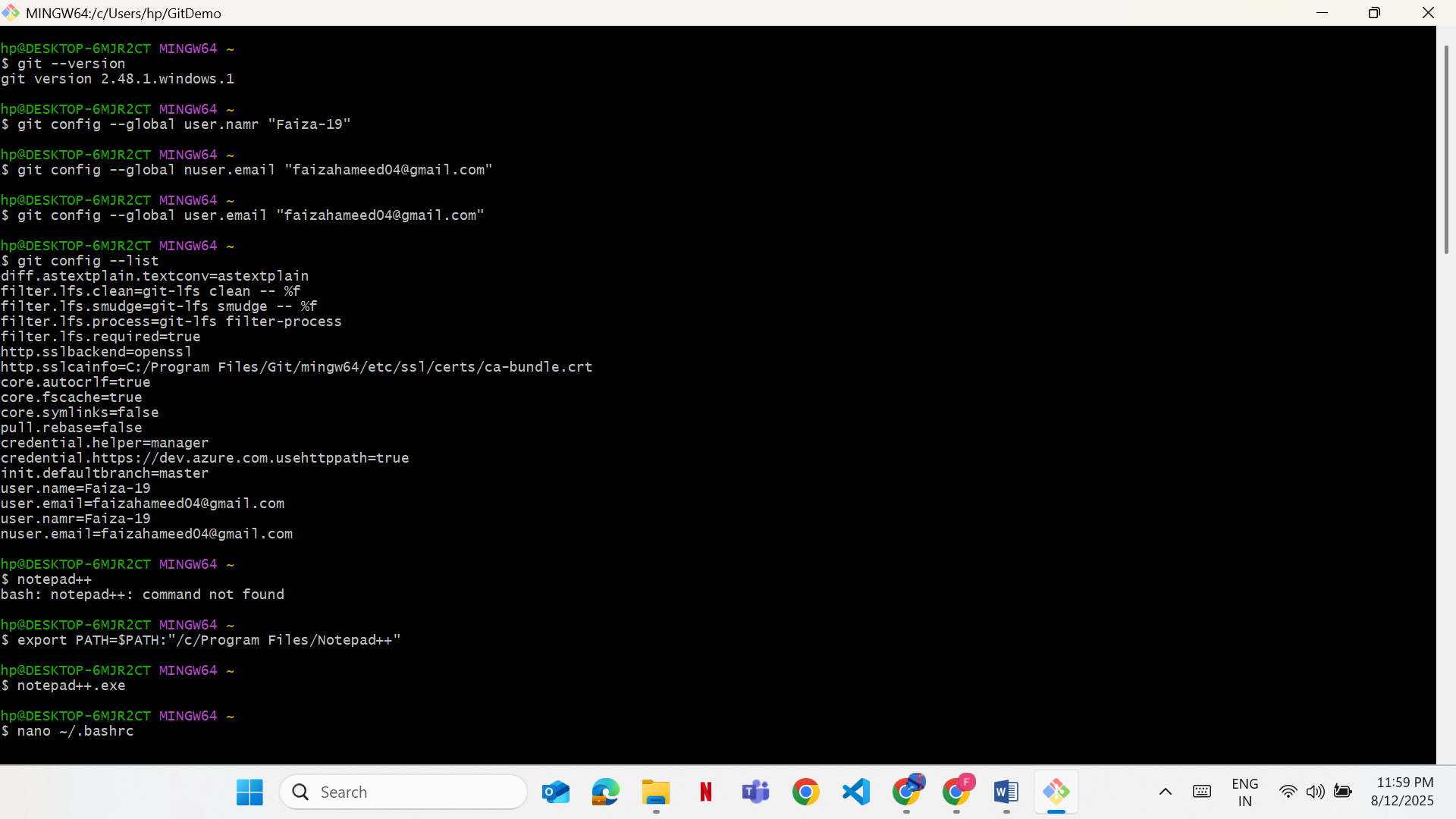
bash

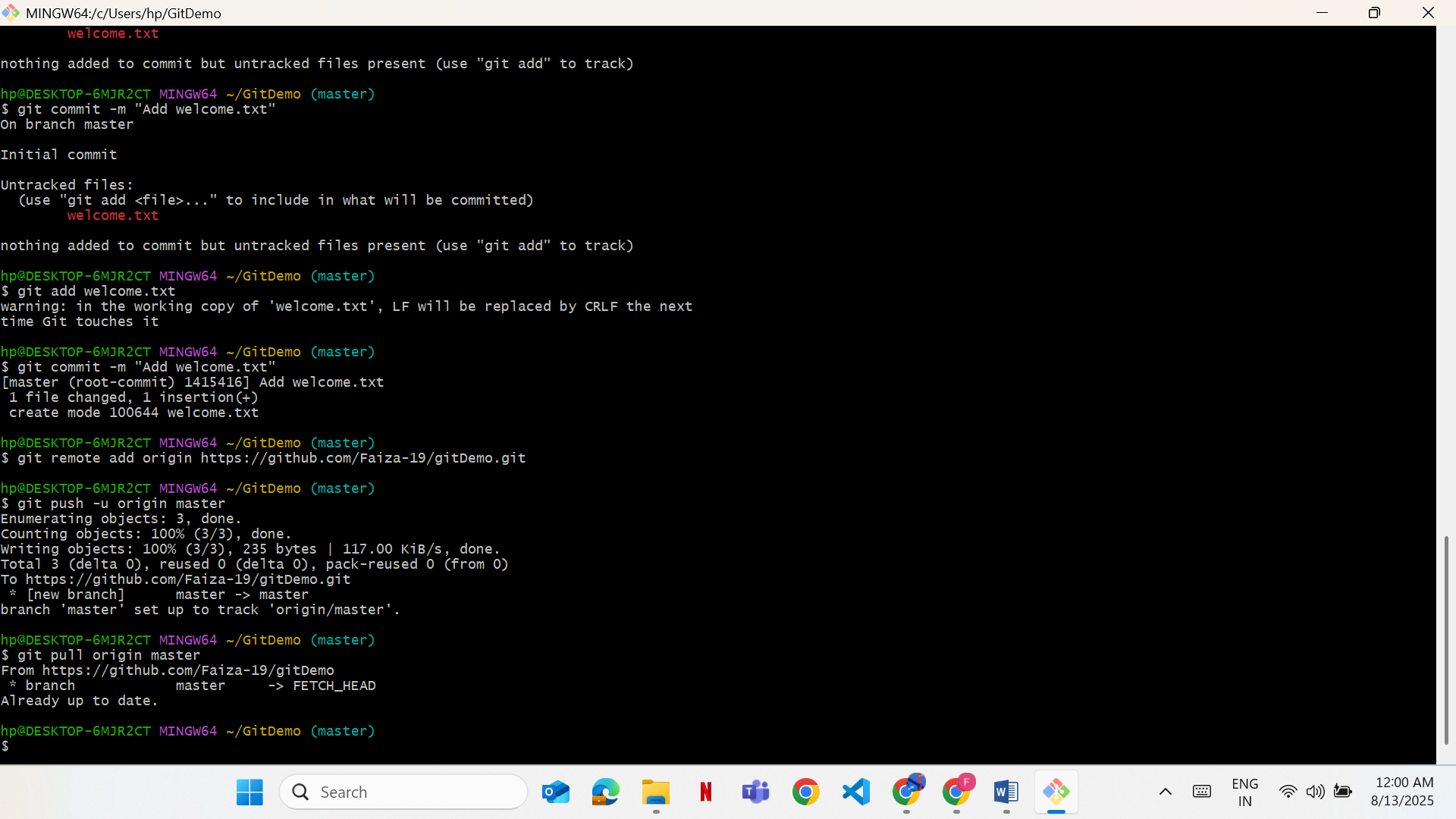
CopyEdit

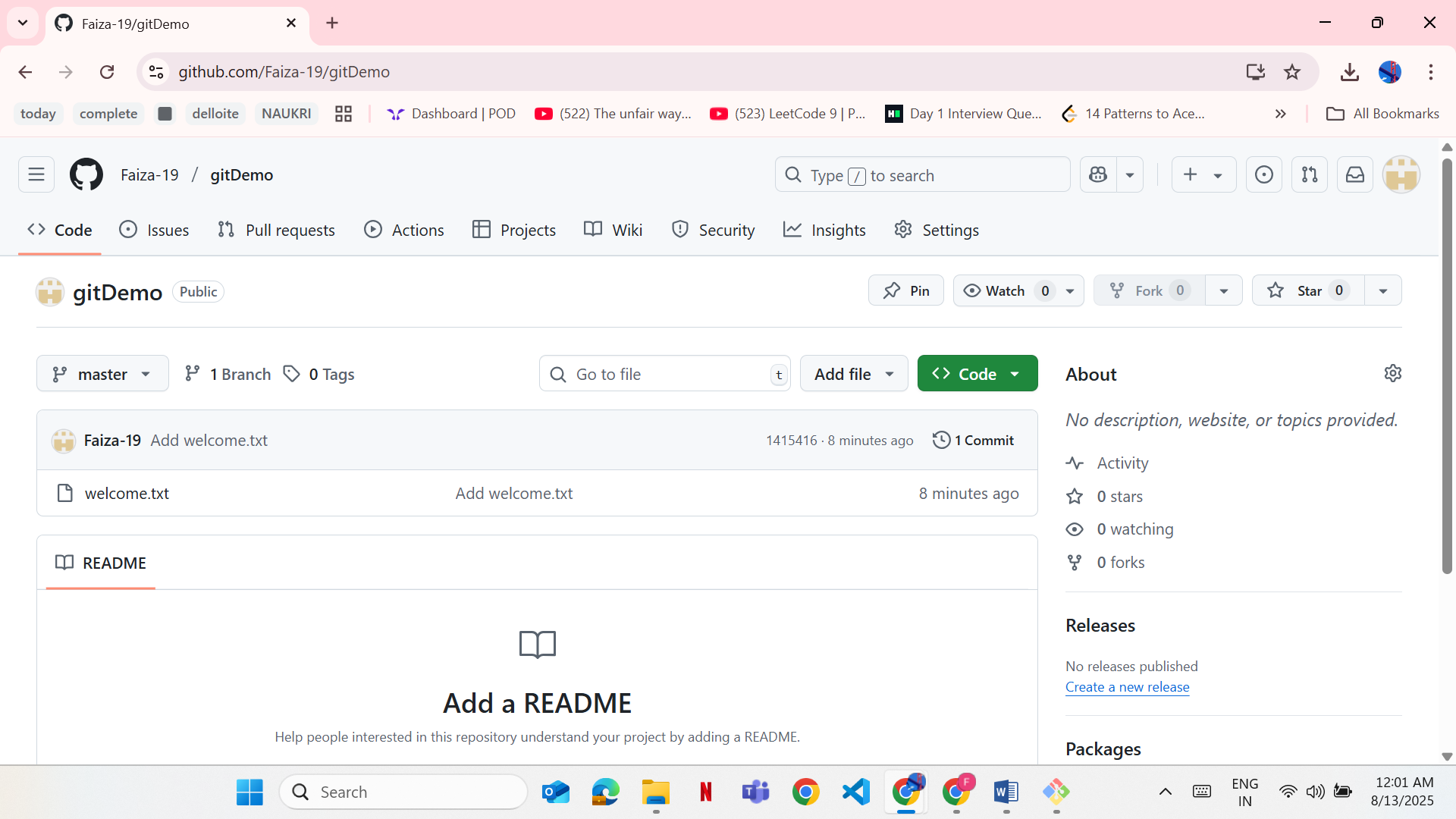
git remote add origin <REMOTE\_URL>

git push -u origin master

**Output:**

****

****

****

**2. GIT-HOL- Ignoring Files with .gitignore**

Create a **“.log”** file and a **log folder** in the working directory of Git. Update the **.gitignore** file in such a way that on committing, these files (.log extensions and log folders) are ignored.

Verify if the git status reflects the same about working directory, local repository and git repository.

**Steps:**

**1. Move into Repository**

bash

CopyEdit

cd ~/GitDemo

**2. Create Log Files**

bash

CopyEdit

echo "Log content" > error.log

mkdir log

echo "Some log" > log/test.log

**3. Create .gitignore**

bash

CopyEdit

echo "\*.log" > .gitignore

echo "log/" >> .gitignore

**4. Commit .gitignore**

bash

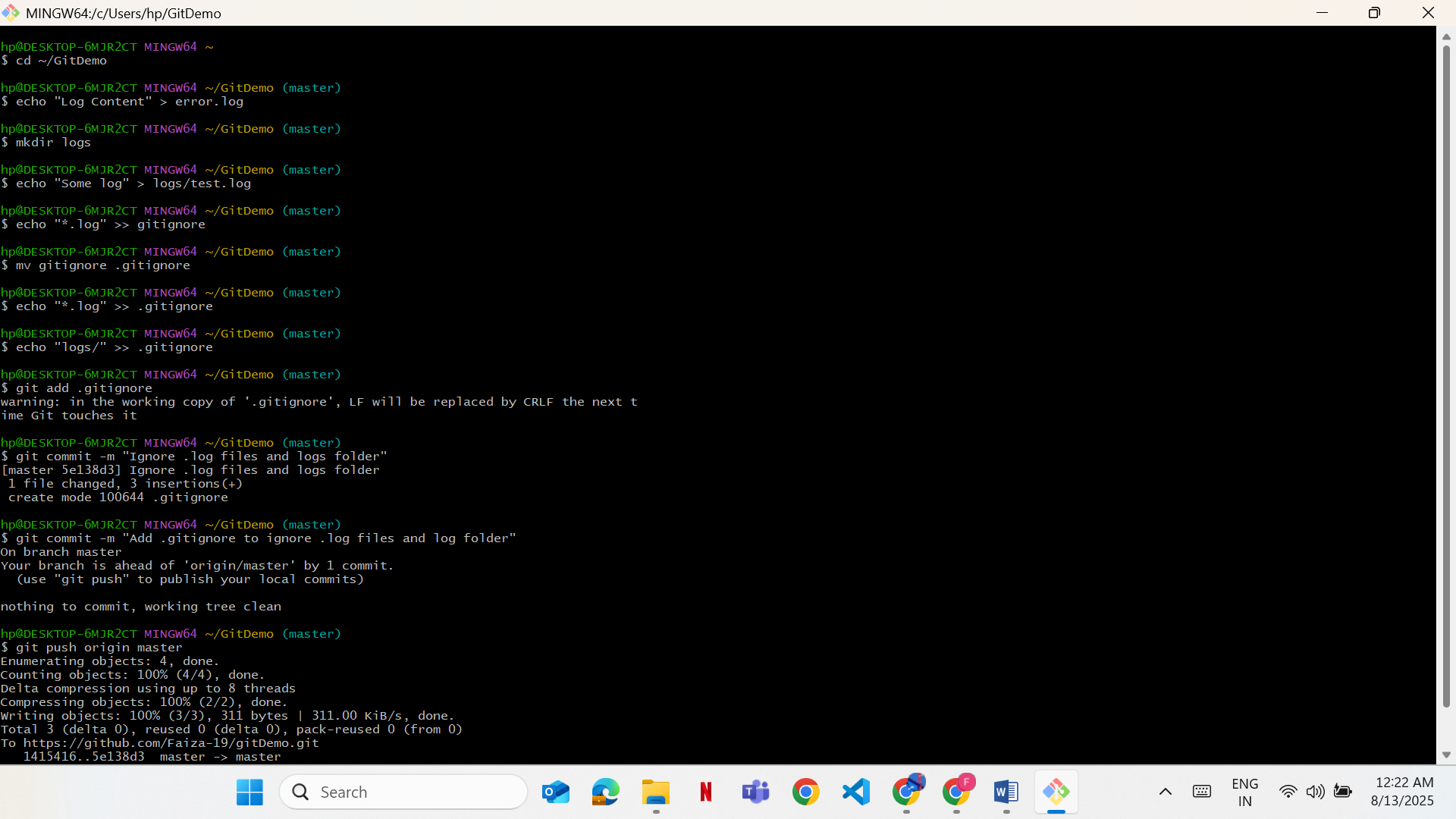
CopyEdit

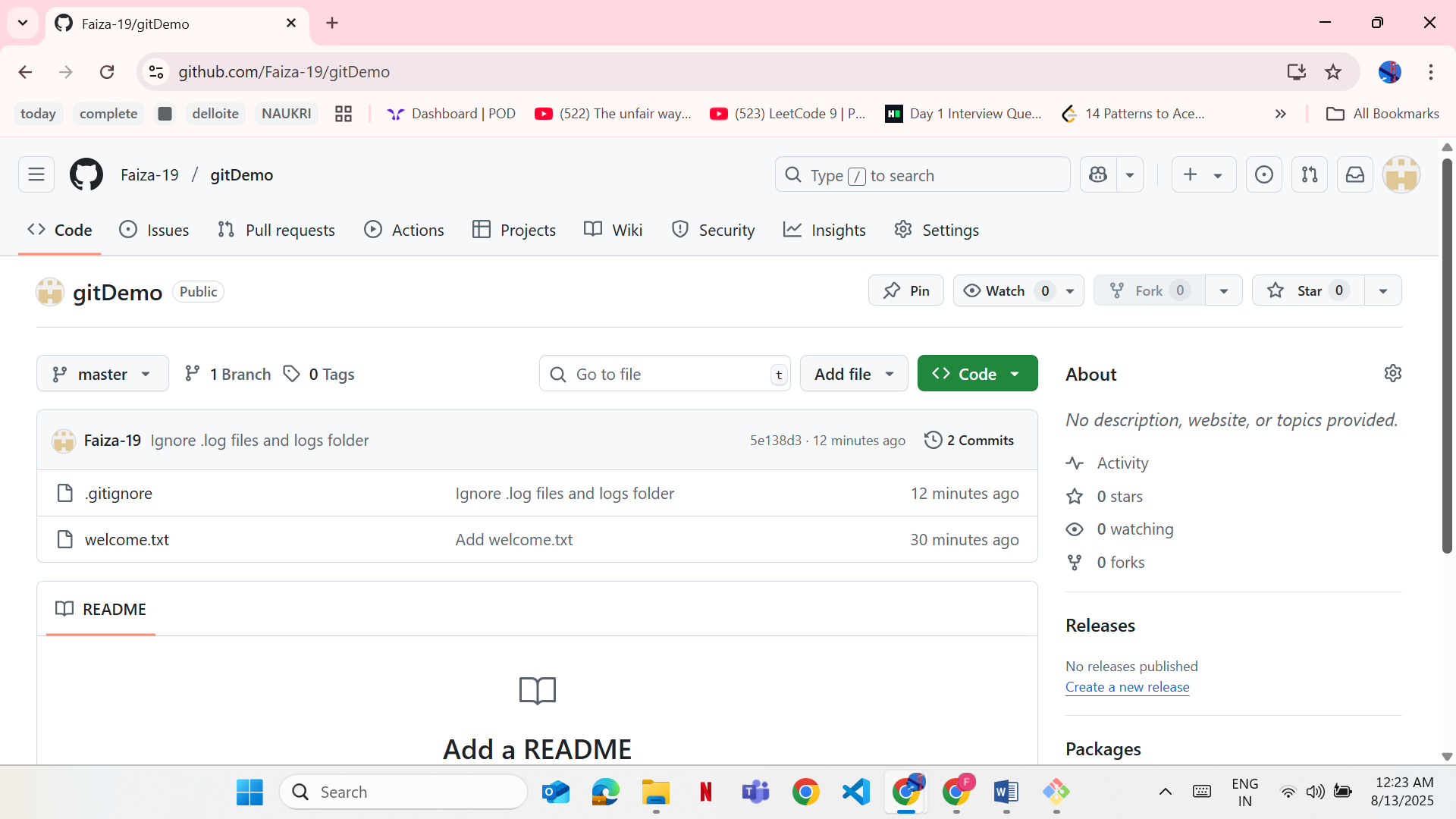
git add .gitignore

git commit -m "Add .gitignore to ignore .log files and log folder"

git push origin master

Output:





### 3. GIT-HOL- ****Branching & Merging****

Construct a branch, do some changes in the branch, and merge it with master (or trunk)

**Steps:**

 Created and switched to a new branch:

bash

CopyEdit

git checkout -b GitNewBranch

 Created a new file in the branch:

bash

CopyEdit

echo "File from new branch" > branch.txt

 Staged the file:

bash

CopyEdit

git add branch.txt

 Committed the file:

bash

CopyEdit

git commit -m "Add branch.txt in GitNewBranch"

 Switched back to master:

bash

CopyEdit

git checkout master

 Compared changes between master and GitNewBranch using command-line diff:

bash

CopyEdit

git diff GitNewBranch

 Compared changes visually using **P4Merge**:

* First, set P4Merge as the merge tool (only needed once):

bash

CopyEdit

git config --global merge.tool p4merge

git config --global mergetool.p4merge.path "/c/Program Files/Perforce/p4merge.exe"

* Then run:

bash

CopyEdit

git mergetool

This opens P4Merge to show file differences.

 Merged GitNewBranch into master:

bash

CopyEdit

git merge GitNewBranch

 Verified merge log:

bash

CopyEdit

git log --oneline --graph --decorate

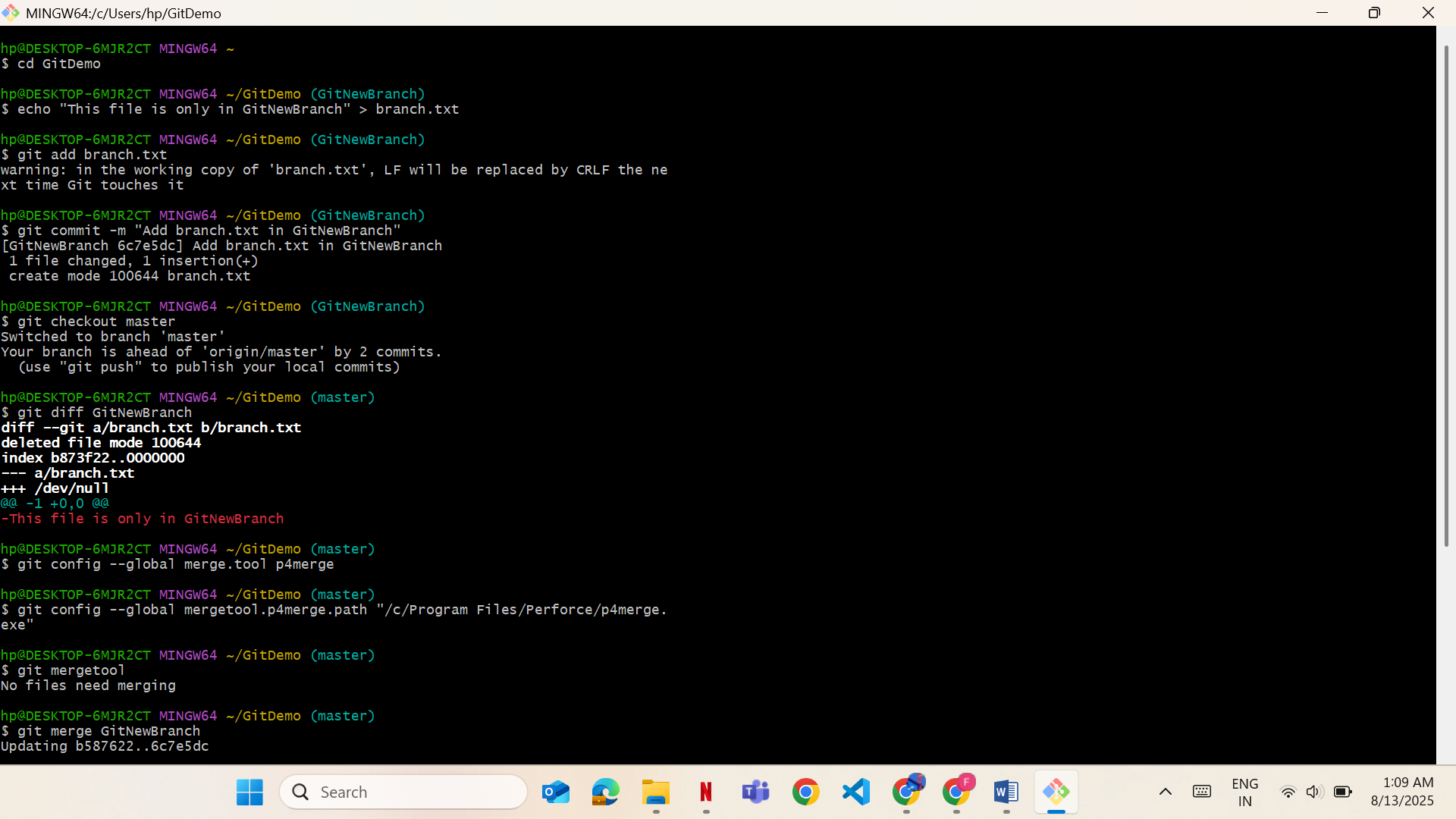
 Deleted the branch after merge:

bash

CopyEdit

git branch -d GitNewBranch

**Output:**



### 

### 4. GIT-HOL ****– Merge Conflict Resolution****

Implement conflict resolution when multiple users are updating the trunk (or master) in such a way that it results into a conflict with the branch’s modification.

**Steps:**

**1. Create Branch and Add File**

git checkout -b GitWork

echo "<note>Hello from branch</note>" > hello.xml

git add hello.xml

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

**2. Switch to Master and Add Conflicting File**

git checkout master

echo "<note>Different content from master branch</note>" > hello.xml

git add hello.xml

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

**3. Merge (Conflict Occurs)**

git merge GitWork

Conflict markers will appear in *hello.xml*.

**4. Resolve Conflict**  
Edit hello.xml to:

xml

CopyEdit

<note>Different content from master branch</note>

<message>Hello again from GitWork branch</message>

**5. Save and Commit**

bash

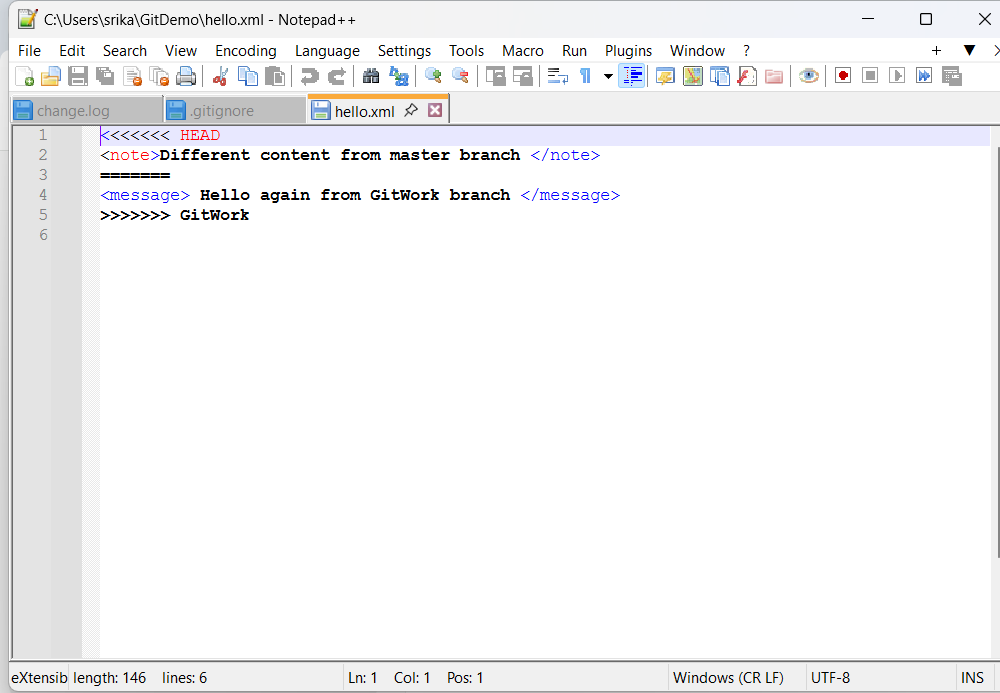
CopyEdit

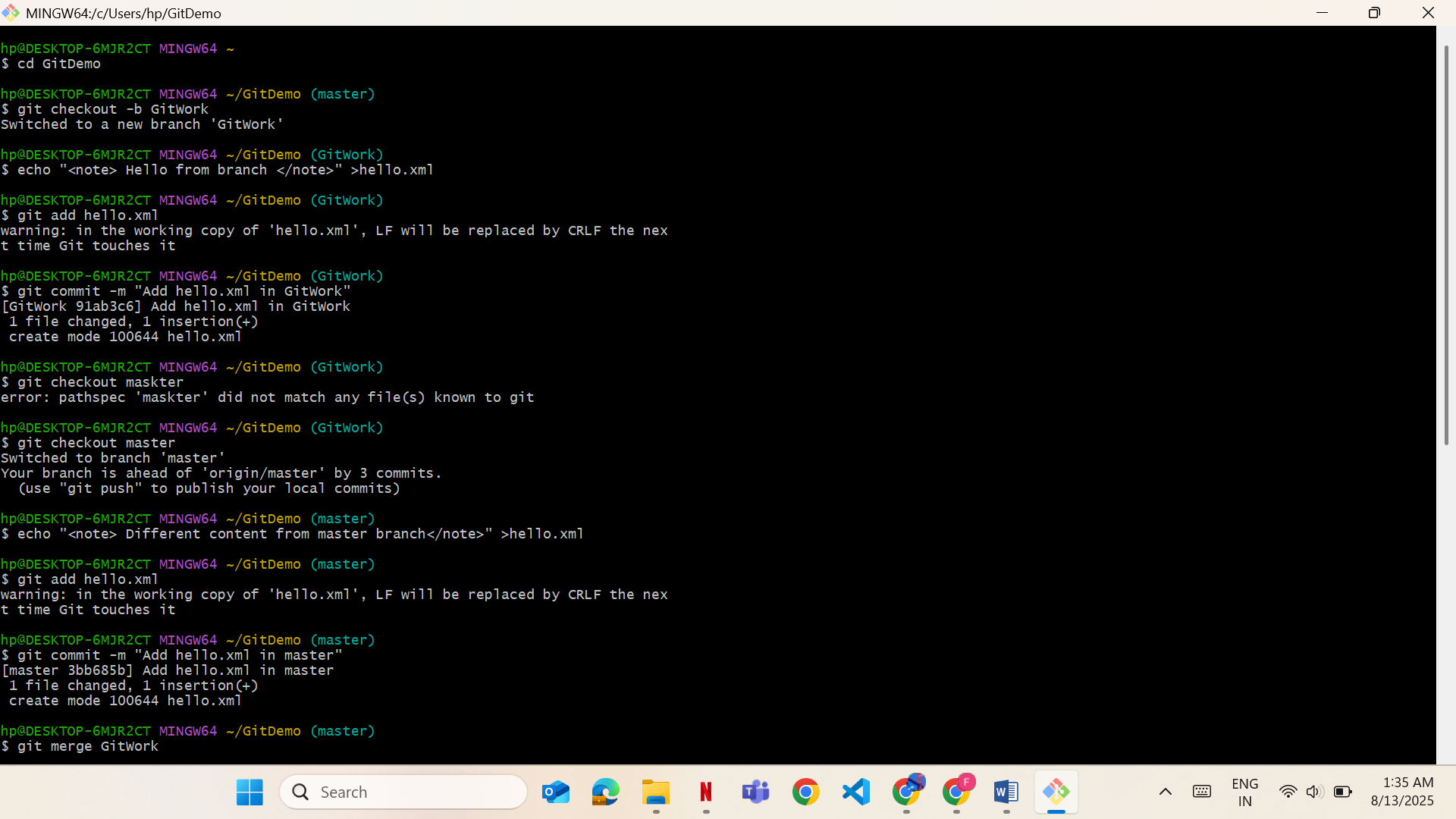
git add hello.xml

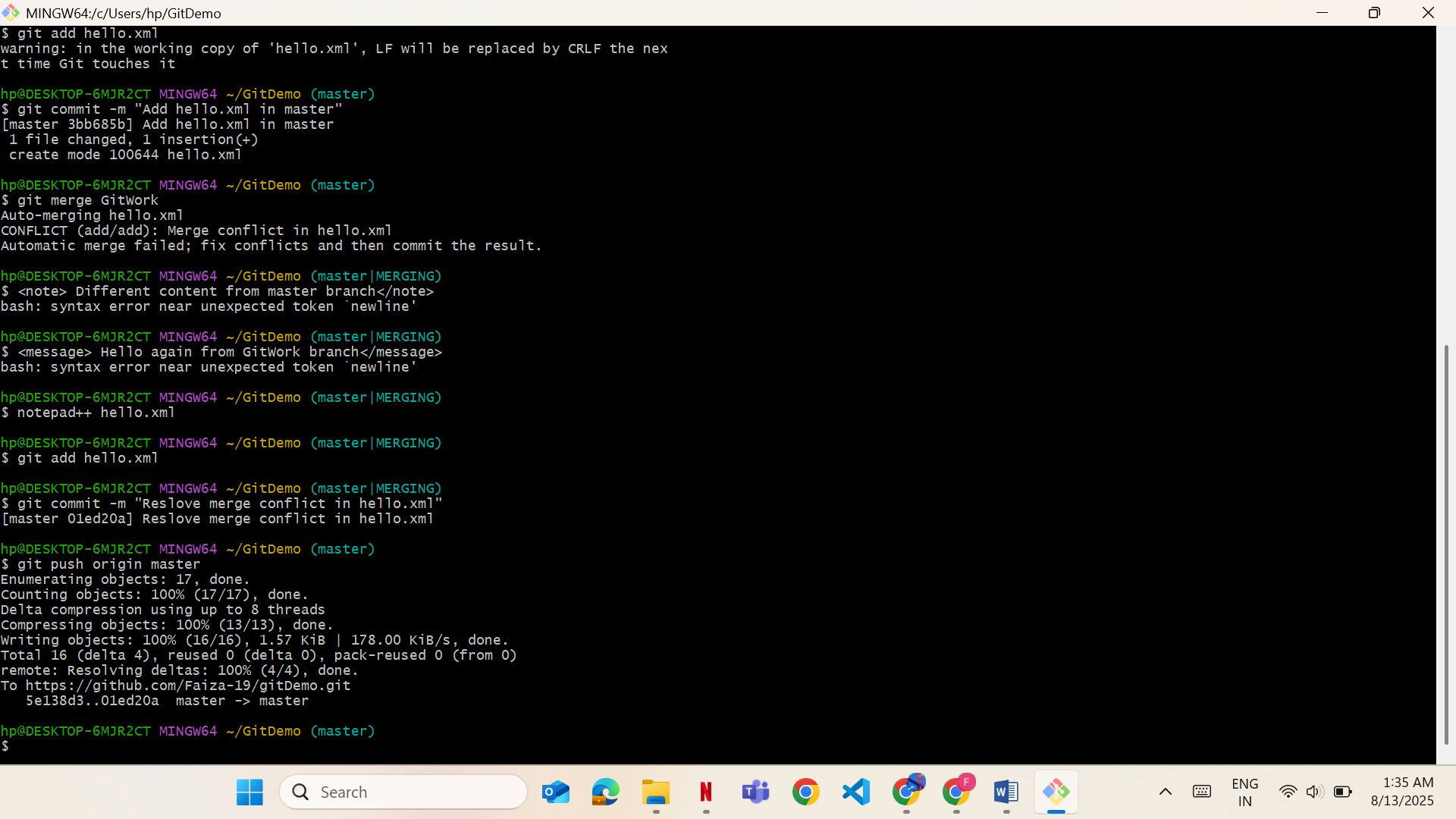
git commit -m "Resolve merge conflict for hello.xml"

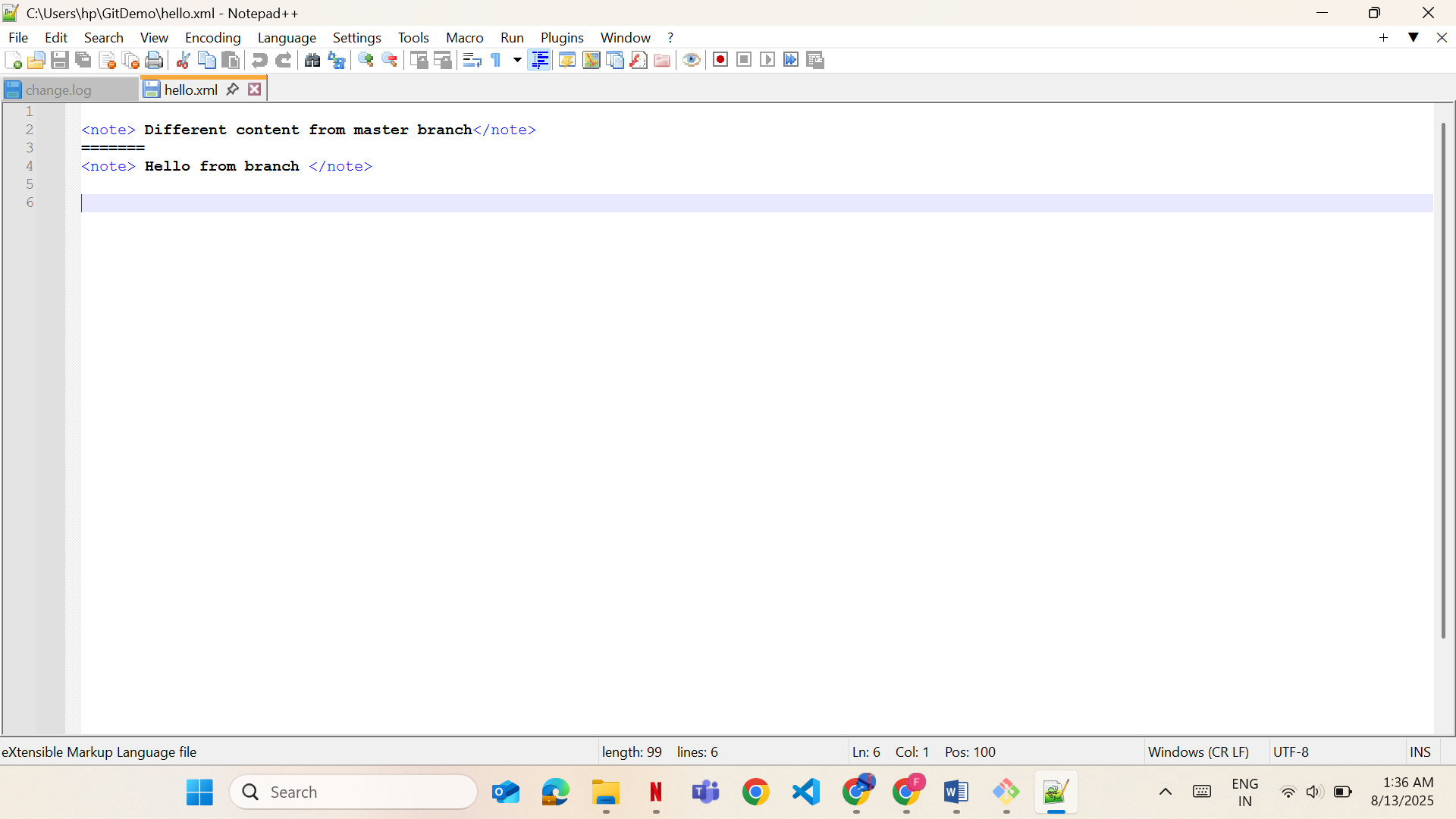
git push origin master

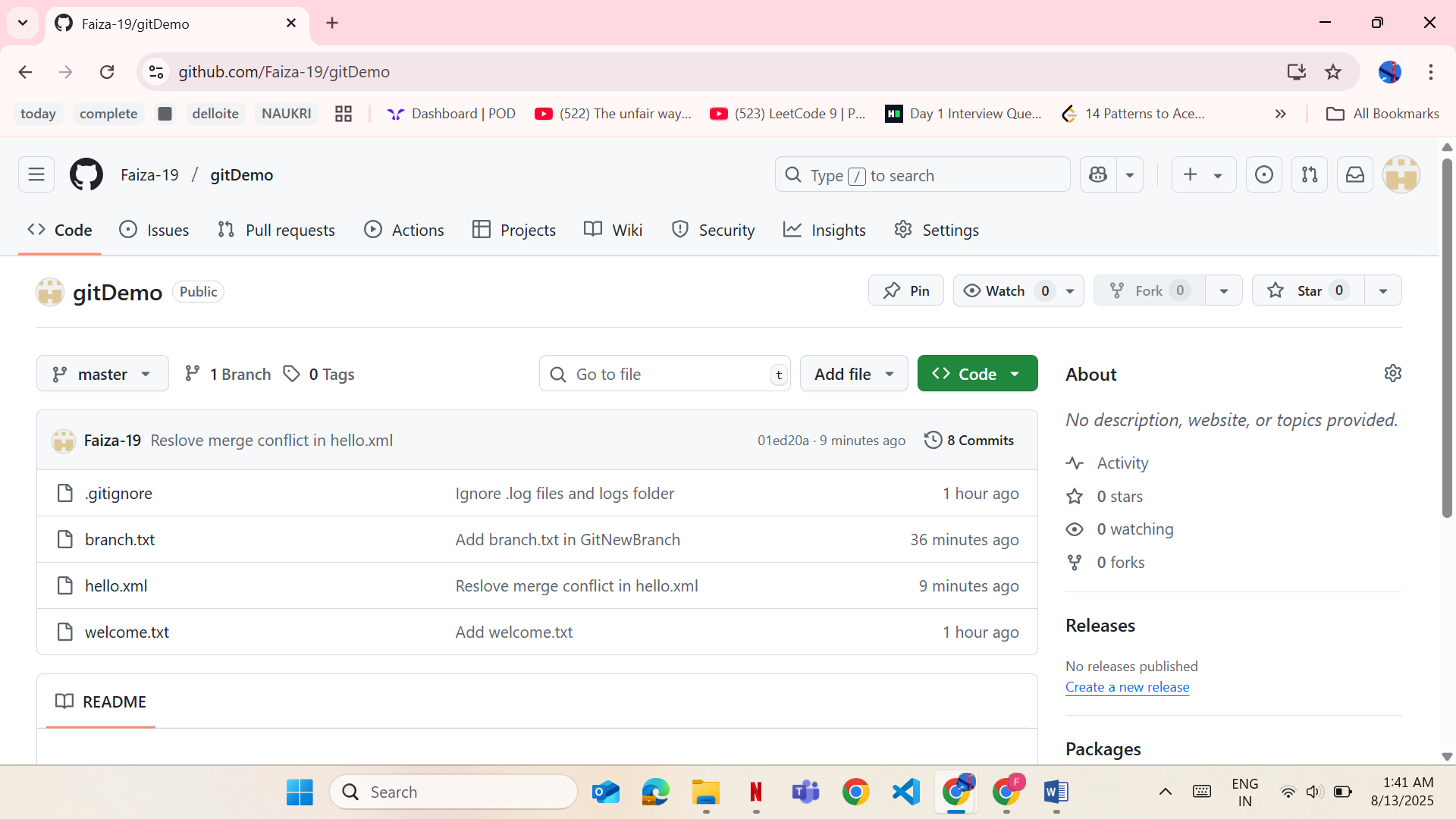
**Output:**

****

****

****

****

****

**5. GIT-HOL-**

Execute steps involving clean up and push back to remote Git.

**Steps:**

 Switched to master:

git checkout master

 Checked status:

git status

 Listed all branches:

git branch -a

 Pulled latest changes from remote:

git pull origin master

 Pushed pending changes:

git push origin master

**Output:**

