**Cognizant Deep Skilling - Digital Nurture 4.0**

**GIT**

**1. Git-HOL**

**Git Hands-on Lab: Basic Git Operations with Notepad++ Integration**

**Objectives :**

Use basic Git commands: git init, git status, git add, git commit, git push, git pull

Set up Git configuration and Notepad as default editor

Work with a source code repository

**Step 1: Setup Git Configuration**

Open Git Bash and check Git version:

git --version

Verify configuration:

git config --global --list

**Step 2: Integrate Notepad as Git's Default Editor**

Check if Notepad runs from Git Bash:

notepad

**Step 3: Create Local Git Repository and Add File**

Open Git Bash in your working directory:

Navigate to folder or run:

Initialize Git repository:

git init

Create a file:

echo "Welcome to Git" > welcome.txt

Check if file exists:

ls

View file content:

cat welcome.txt

Check Git status:

git status

Add file to staging area:

git add welcome.txt

Commit changes:

git commit

In Notepad, write message like:  
Initial commit with welcome.txt  
Then save and close the editor.

**Step 4: Push to GitHub**

Go to <https://github.com> and create a new repository named GitDemo

Do not initialize with README.

Add GitHub repo as remote:

Push to GitHub :

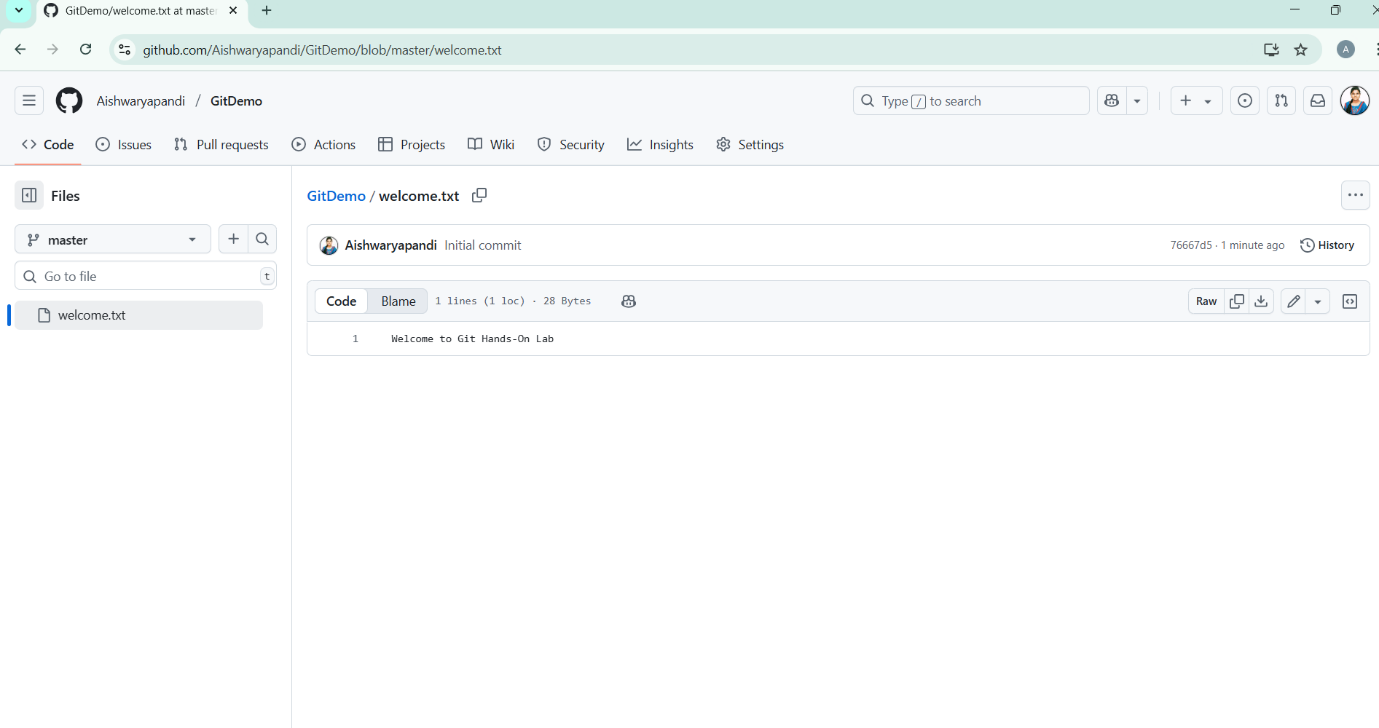
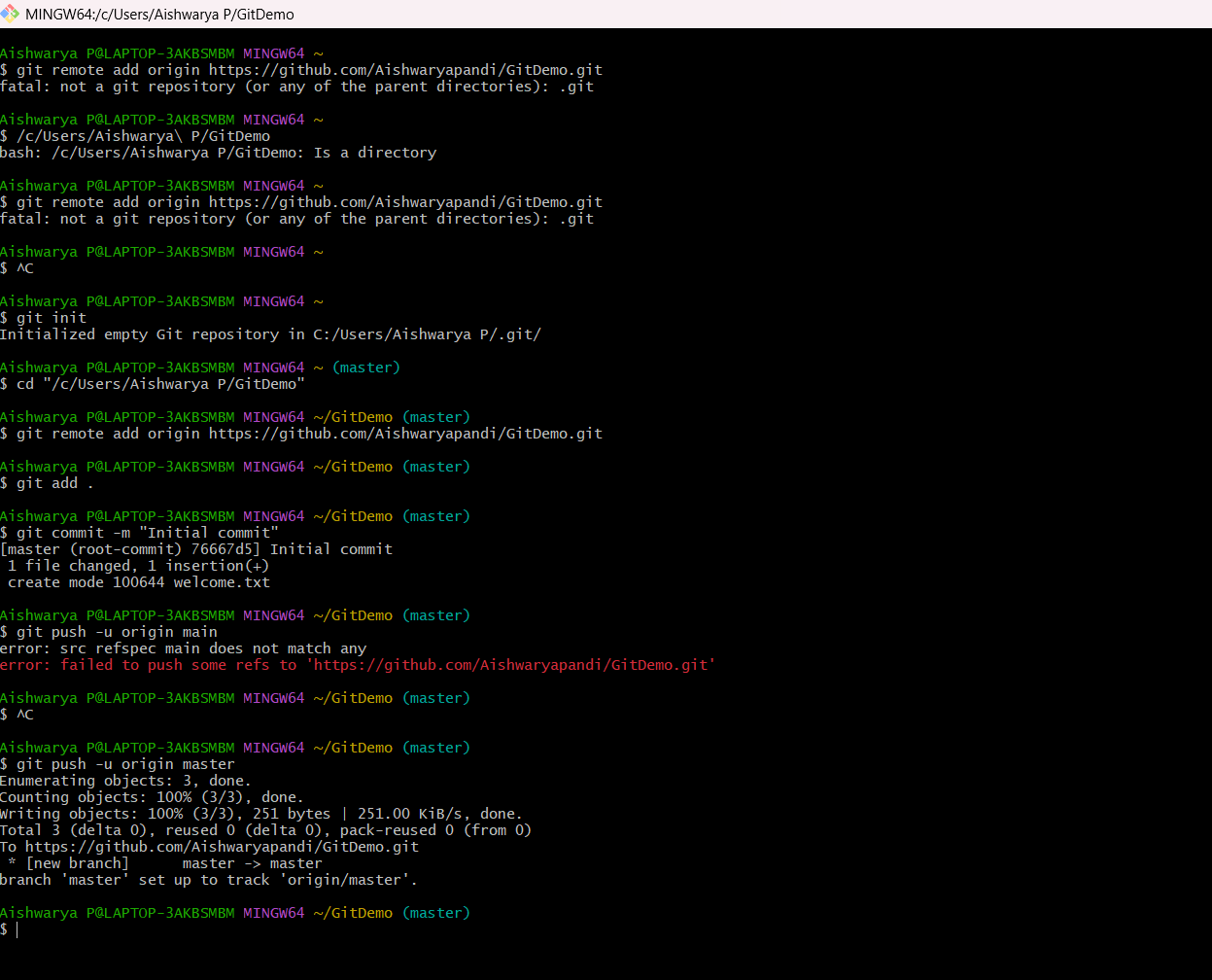
git push -u origin master

**Step 5: Pull from Remote Repository**

To pull latest changes from GitHub:

git pull origin master

**Output :**

****

**2. Git-HOL**

**Git Ignore Hands-On Lab**

**Objectives :**

* Explain what .gitignore is.
* Show how to ignore unwanted files and folders using .gitignore.

**Step 1: Create a Git Working Directory**

mkdir GitIgnoreLab

cd GitIgnoreLab

git init

**Step 2: Create Files and Folders to Ignore**

echo "This is a sample log file" > error.log

mkdir log

echo "Log folder content" > log/debug.txt

**Step 3: Create and Edit .gitignore**

notepad .gitignore

Inside .gitignore, add:

gitignore

\*.log # Ignores all .log files

log/ # Ignores log folder and contents

Save and close the file.

**Step 4: Check Git Status**

git status

**Step 5: Stage and Commit .gitignore**

git add .gitignore

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

**Step 6: Add Remote Repository**

git remote add origin https://github.com/yourusername/GITIgnore.git

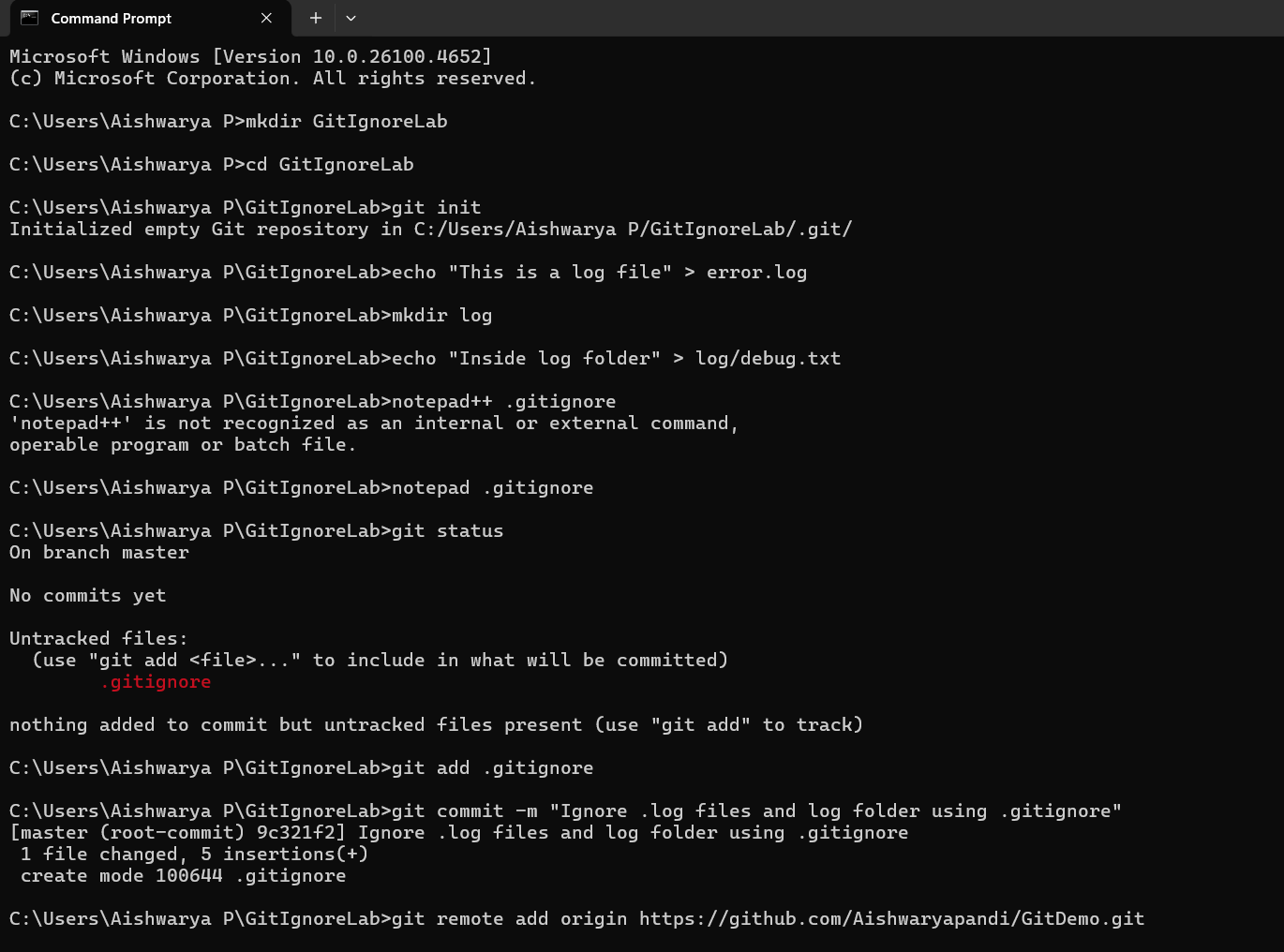
**Step 7: Push Code to GitHub**

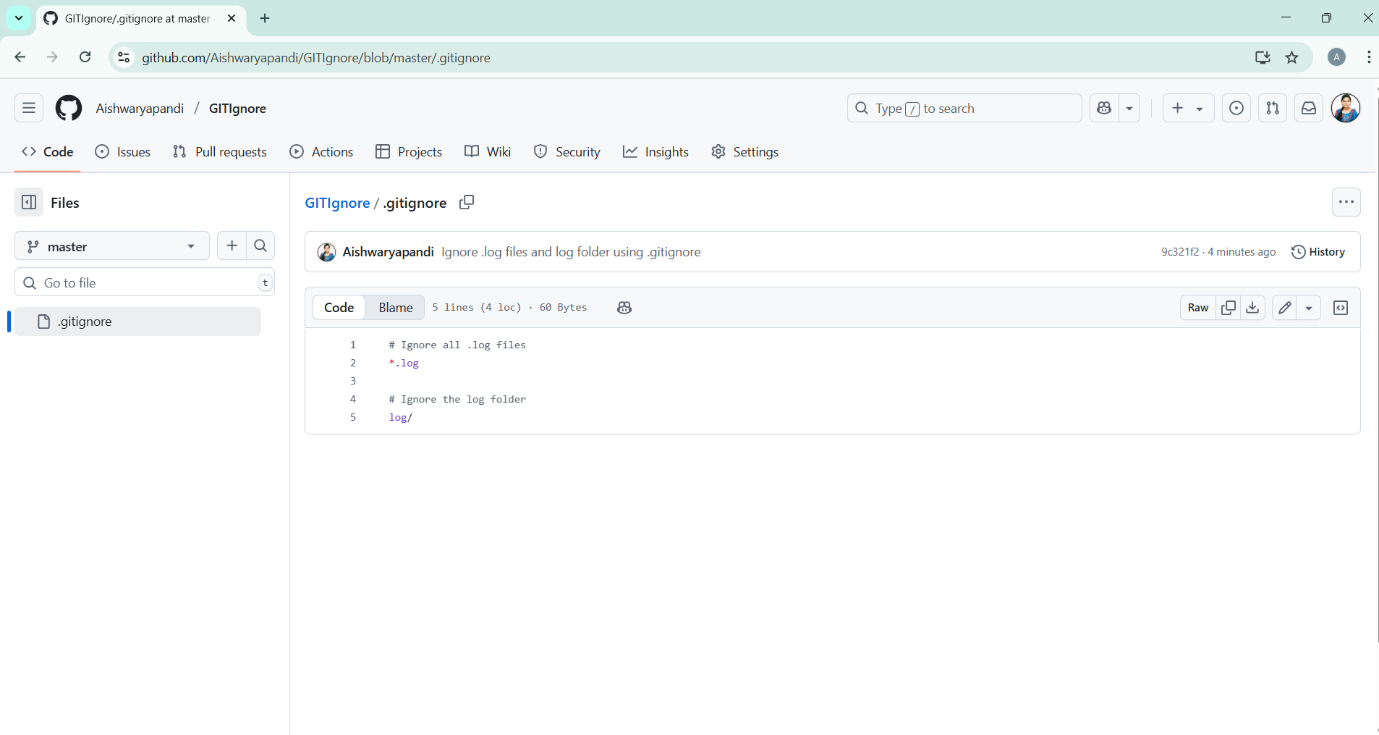
git push -u origin master

**Final Verification**

git status

**Output :**

****

****

**3. Git-HOL**

**Git Branching and Merging Hands-On Lab**

**BRANCHING**

Step 1: Create a new branch “GitNewBranch”

git branch GitNewBranch

Step 2: List all local and remote branches

git branch -a

Step 3: Switch to the newly created branch

git checkout GitNewBranch

Step 4: Add some files with content

echo "This is a file in GitNewBranch" > branchfile.txt

git add branchfile.txt

Step 5: Commit the changes to the branch

git commit -m "Added branchfile.txt in GitNewBranch"

Step 6: Check status

git status

**MERGING**

Step 1: Switch back to master (or main)

git checkout master

Step 2: Show differences between master and GitNewBranch

git diff master..GitNewBranch

Step 4: Merge the source branch (GitNewBranch) into master

git merge GitNewBranch

git add .

git commit -m "Resolved merge conflicts"

Step 5: Observe merge log (graph view)

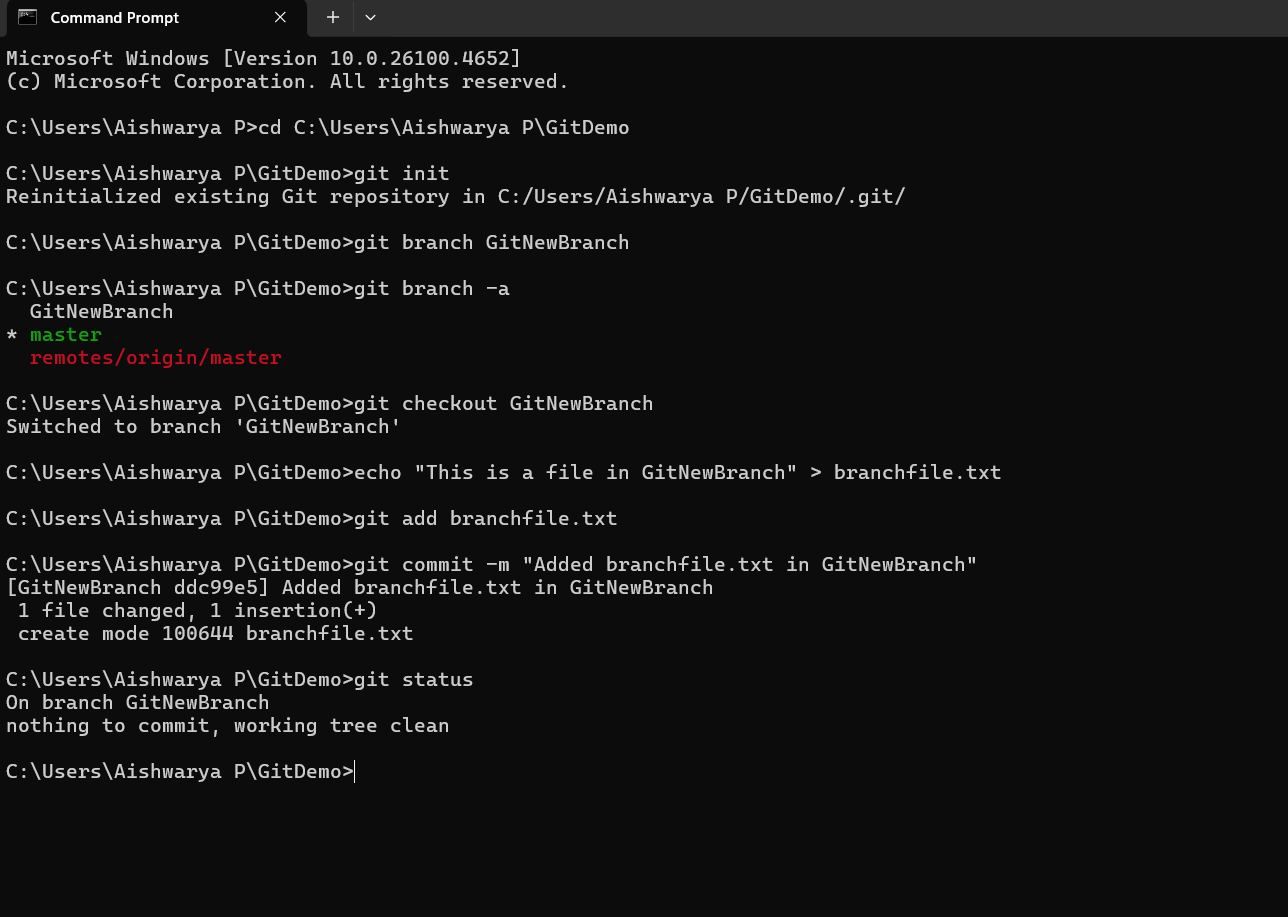
git log --oneline --graph --decorate

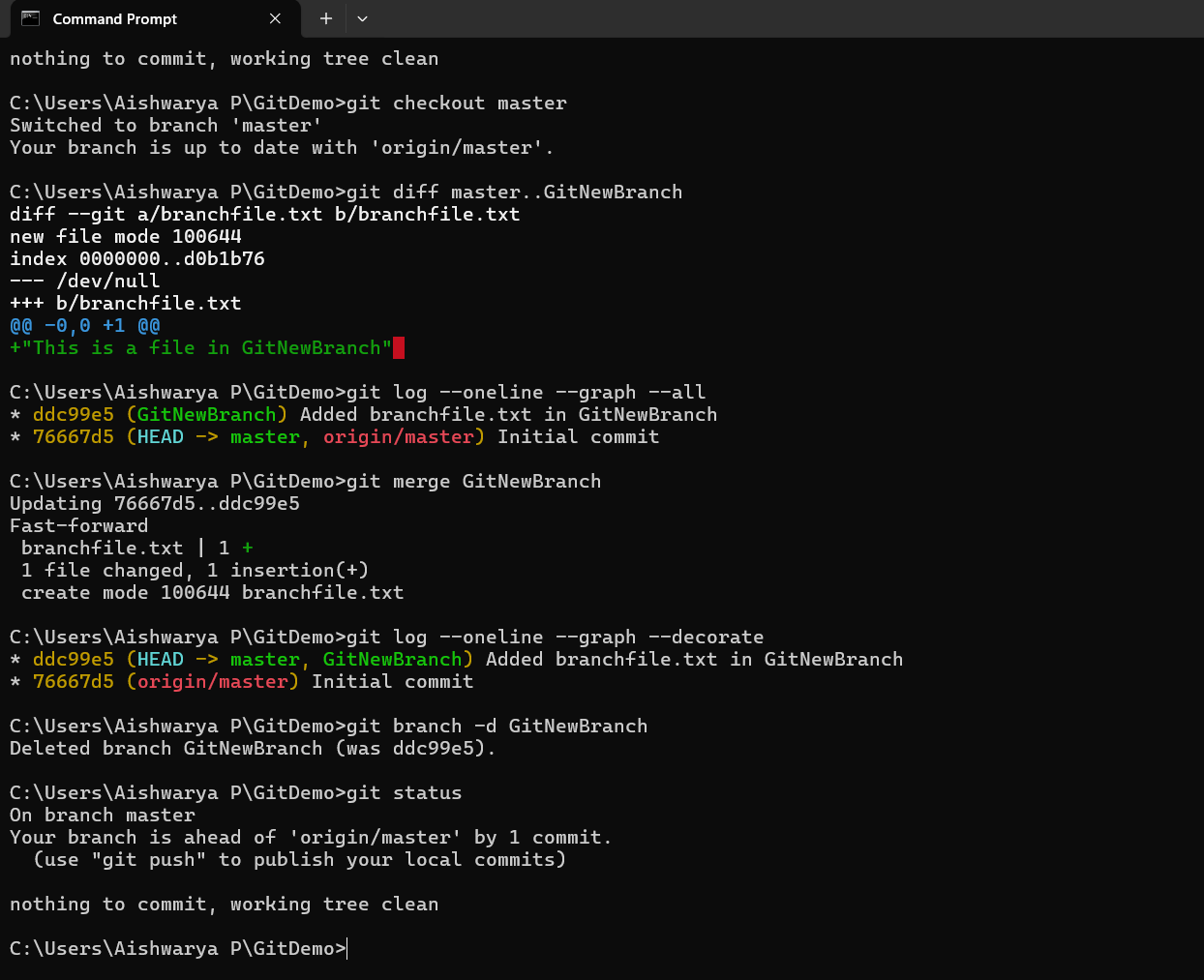
Step 6: Delete the branch and check status

git branch -d GitNewBranch

git status

**Output:**





**4. Git-HOL**

**Git Merge Conflict Resolution**

Step 1: Verify master is in clean state

git status

Step 2: Create a branch “GitWork”. Add a file “hello.xml”.

git checkout -b GitWork

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

Step 3: Update the content of “hello.xml” and observe the status

echo "<note>This is branch GitWork</note>" >> hello.xml

git status

Step 4: Commit the changes to reflect in the branch

git add hello.xml

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

Step 5: Switch to master

git checkout master

Step 6: Add a file “hello.xml” to the master with different content

echo "<greeting>Hello from master</greeting>" > hello.xml

echo "<note>This is master branch</note>" >> hello.xml

Step 7: Commit the changes to the master

git add hello.xml

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

Step 8: Observe the log with a visual graph

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

Step 9: Check the differences with Git diff tool

git diff GitWork hello.xml

Step 11: Merge the branch into master

git merge GitWork

Step 12: Observe the Git conflict markers

xml

<<<<<<< HEAD

<greeting>Hello from master</greeting>

<note>This is master branch</note>

=======

<greeting>Hello from GitWork</greeting>

<note>This is branch GitWork</note>

>>>>>>> GitWork

xml

<greeting>Hello from both branches</greeting>

<note>This is master branch</note>

<note>This is branch GitWork</note>

git add hello.xml

Step 14: Commit the merge result

git commit -m "Resolved merge conflict in hello.xml"

Step 15: Check Git status and add backup files to .gitignore

git status

echo "\*~" >> .gitignore

Step 16: Commit the .gitignore changes

git add .gitignore

git commit -m "Added backup file pattern to .gitignore"

Step 17: List all branches

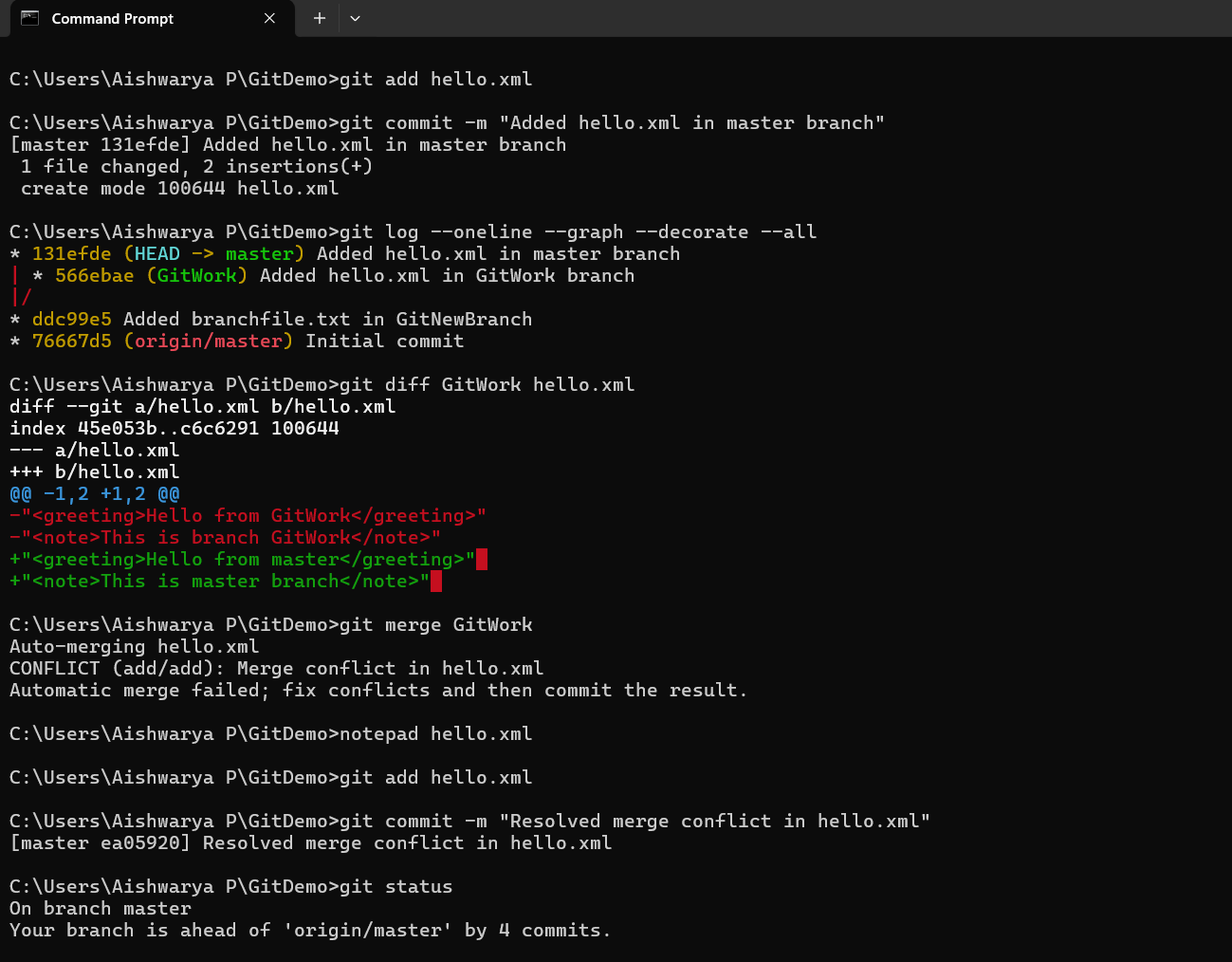
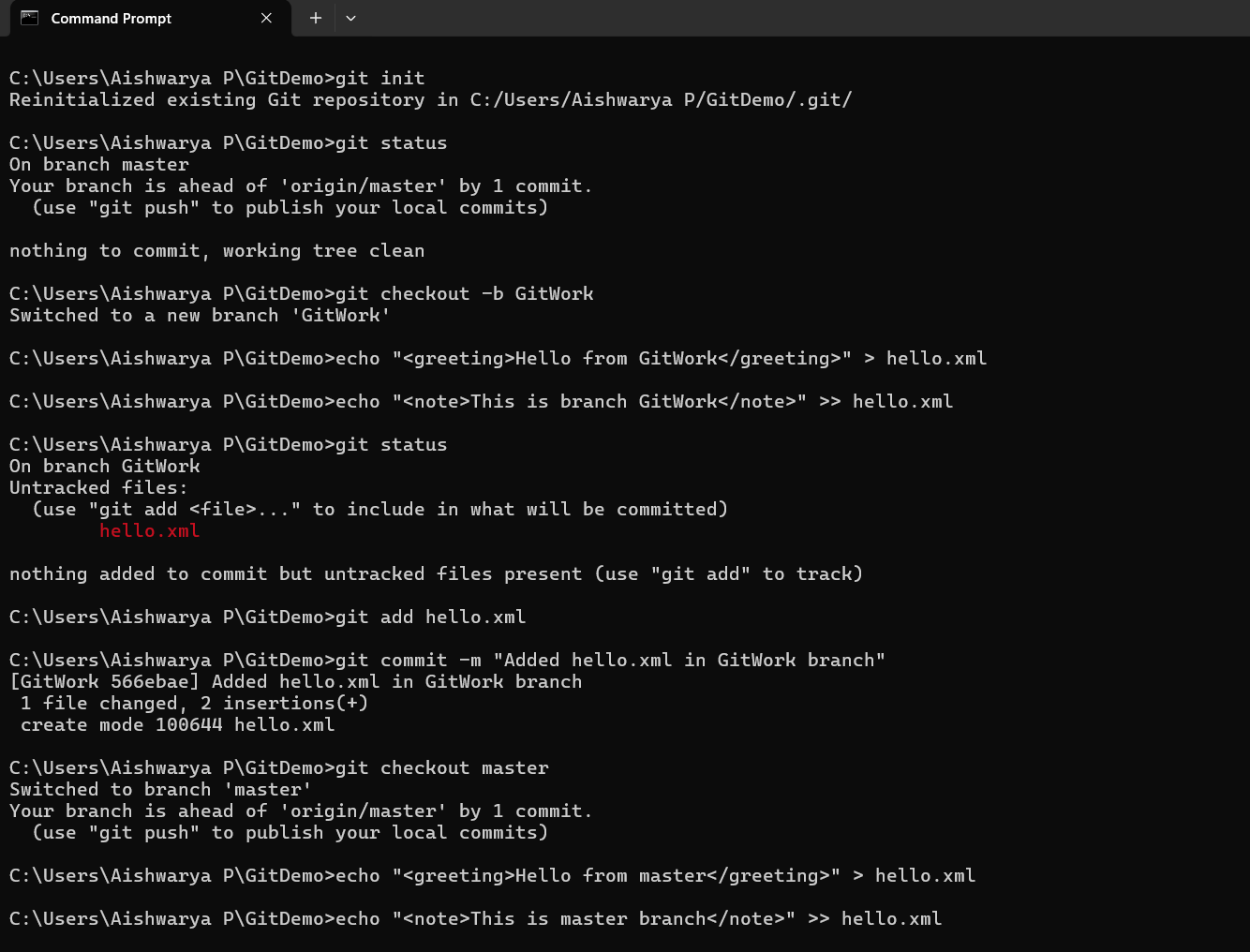
git branch

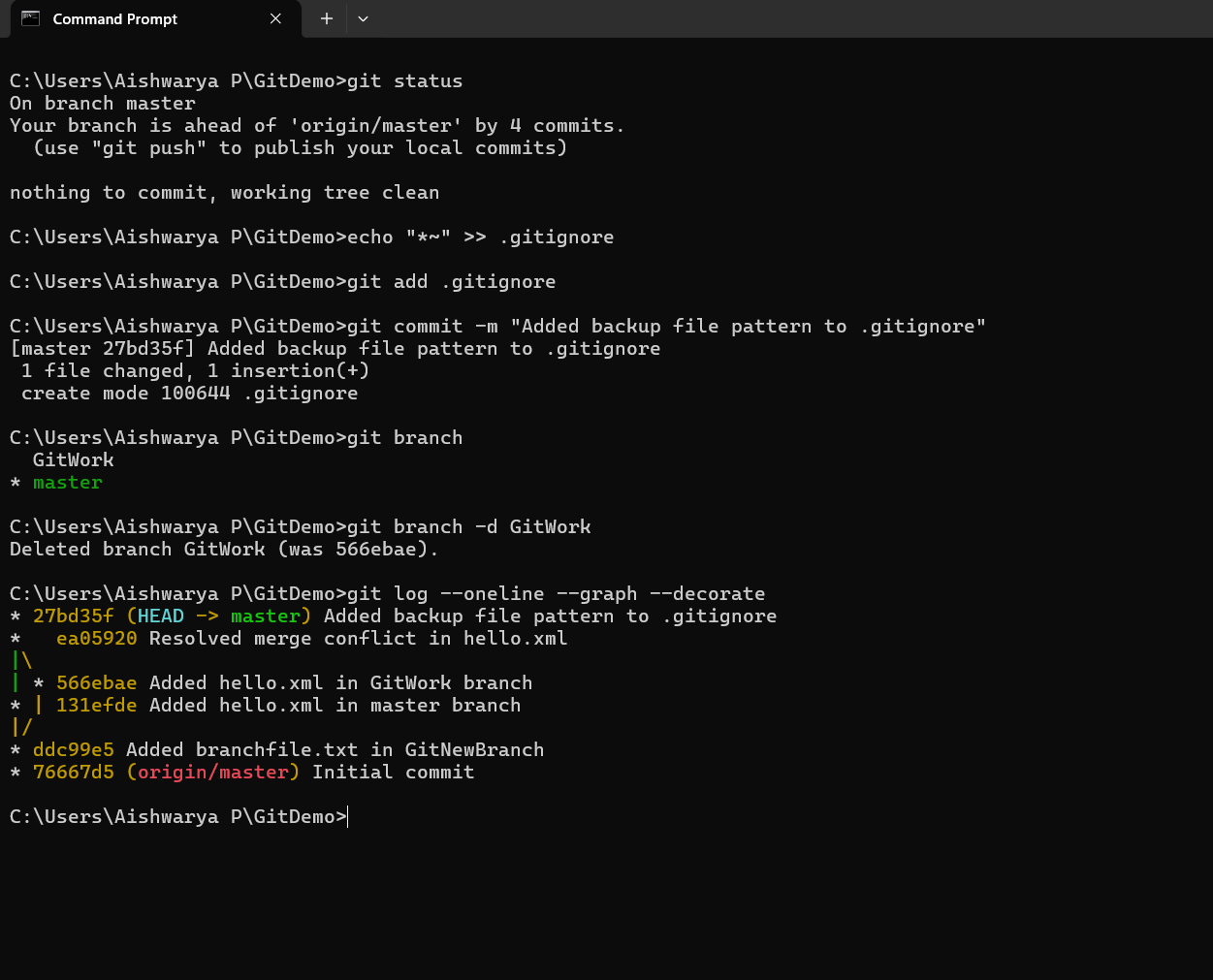
Step 18: Delete the branch which was merged (GitWork)

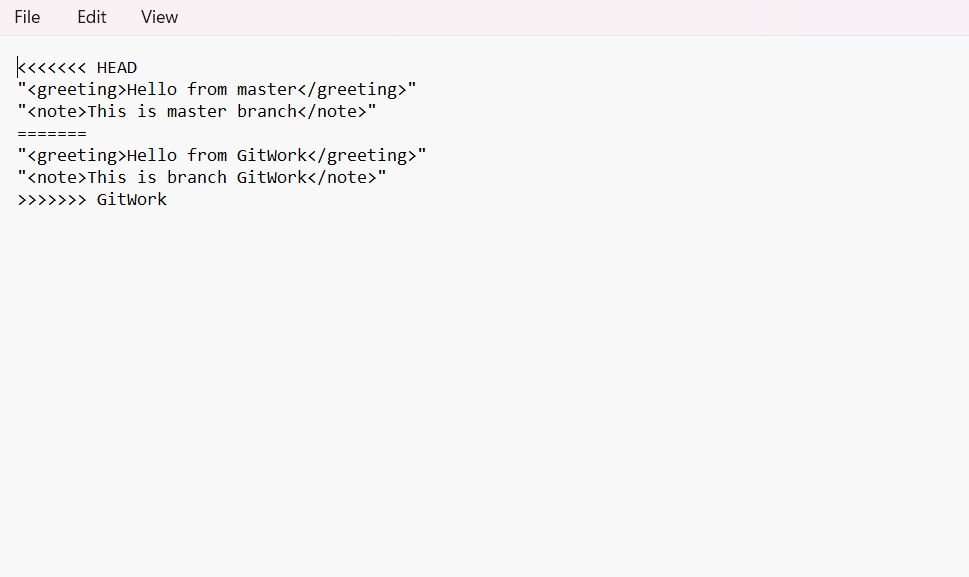
git branch -d GitWork

Step 19: Final log observation

git log --oneline --graph –decorate

**Output:**

****

****

**5. Git-HOL**

**Git Hands-On Lab - Clean up and Push to Remote Repository**

**Objectives:**

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

1. Open Command Prompt and go to your repository folder:

cd C:\Users\Aishwarya P\Git-T03-HOL\_002

2. Verify if the working tree is clean:

git status

3. List all the available branches:

git branch -a

5. Pull the latest changes from remote:

git pull origin main

6. Add files to track:

echo This is a test file for Git lab > readme.txt

git add readme.txt

7. Commit your changes:

git commit -m "Initial commit for Git-T03-HOL\_002"

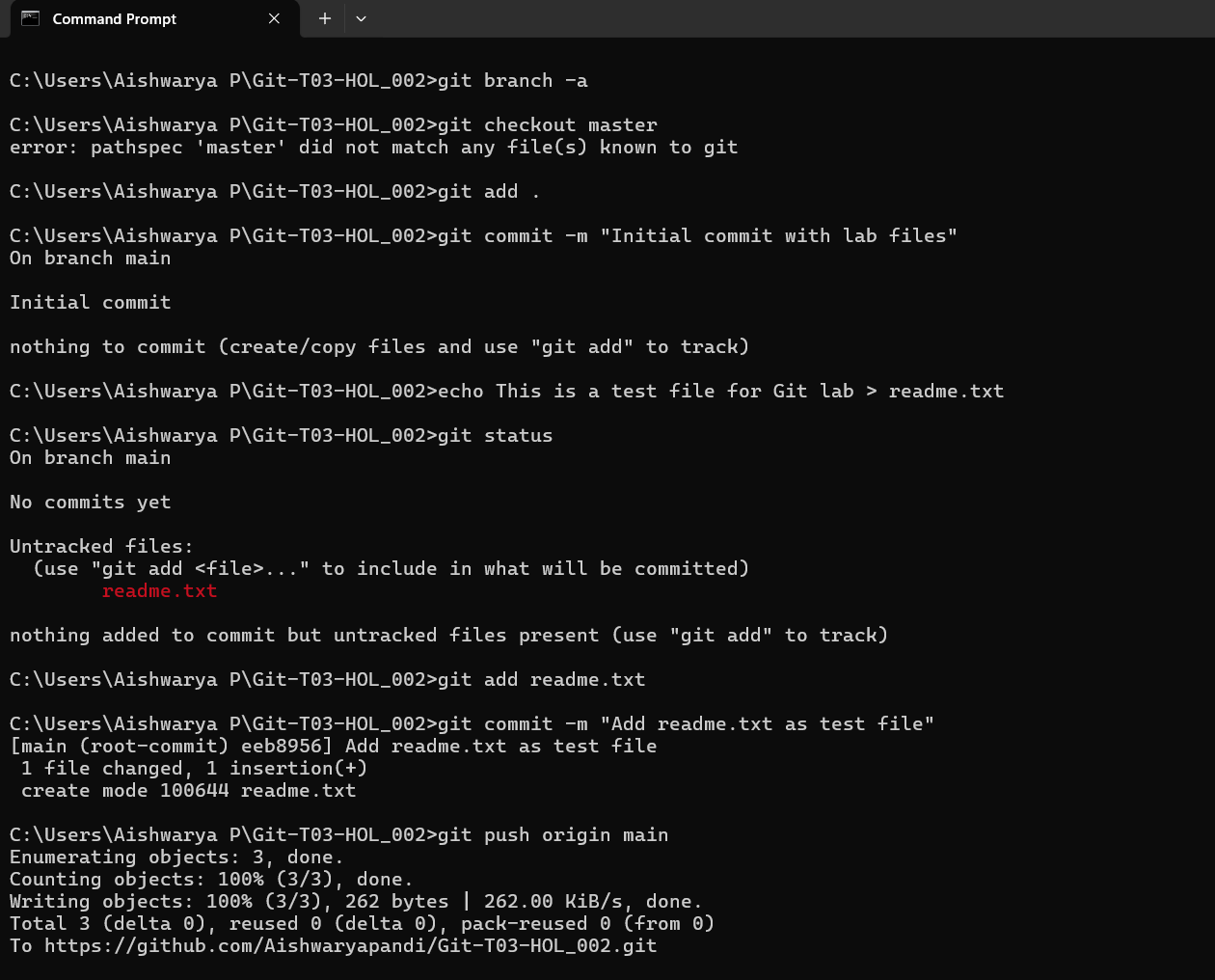
8. Push your changes to GitHub:

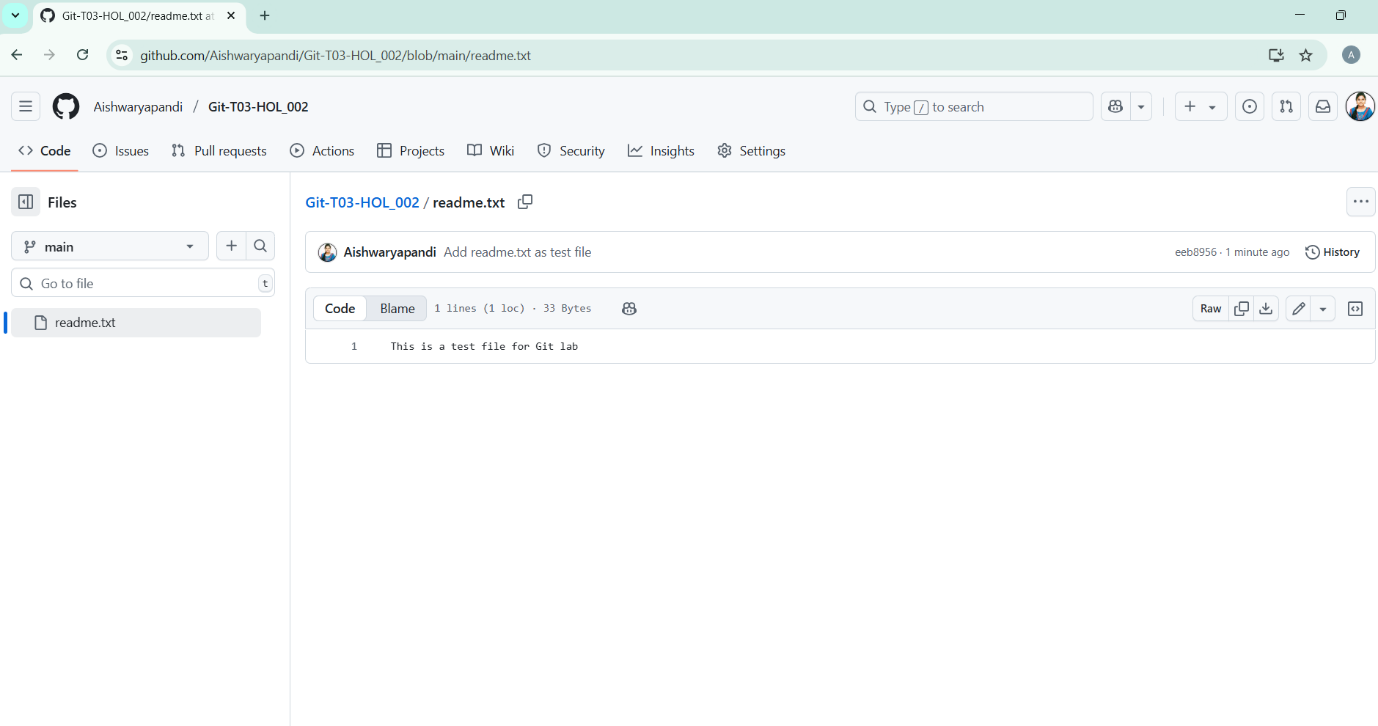
git push origin main

9. Go to GitHub and verify your files were uploaded:

<https://github.com/Aishwaryapandi/Git-T03-HOL_002>

**Output:**

****

****