**Week 8 – Git Hands-on Labs**

**Hands-on 1: Git Basics & Notepad++ Integration**

**Objective**

Learn and practice the core Git commands — git init, git status, git add, git commit, git push, git pull — along with setting up Git configuration and making Notepad++ the default editor.

**My Steps**

1. **Git Setup**
   * Checked Git installation:

git --version

* + Configured my name and email:

git config --global user.name "Name"

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

* + Verified config:

git config --list

1. **Notepad++ Integration**
   * My Notepad++ location: D:\Notepad++\notepad++.exe
   * Added to PATH in Environment Variables so Git Bash could find it.
   * Set as default editor:

git config --global core.editor "'D:/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

1. **Creating a Repo & Adding Files**
   * Initialized repo:

mkdir GitDemo

cd GitDemo

git init

* + Created file:

echo "Welcome to Git" > welcome.txt

* + Added & committed:

git add welcome.txt

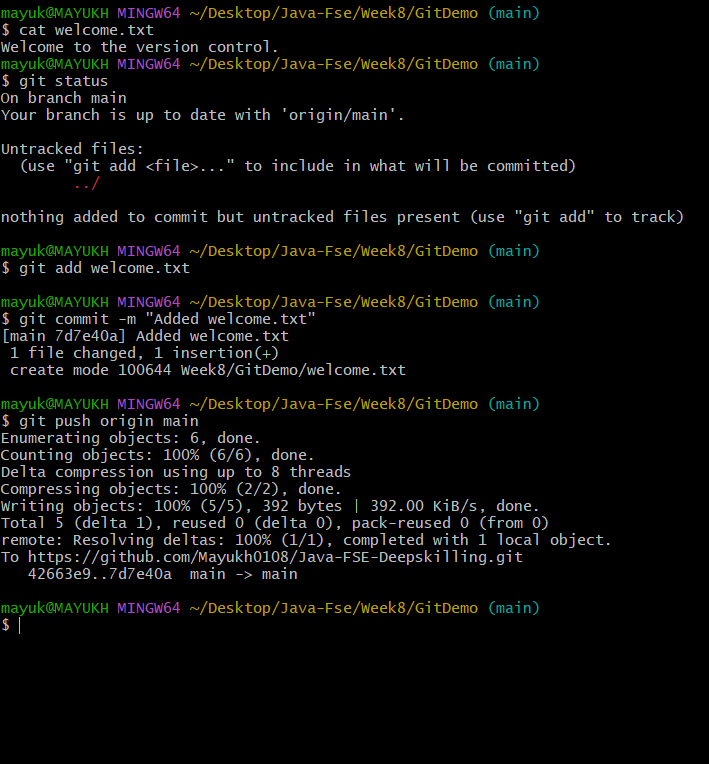
git commit -m "Added welcome.txt"

1. **Connecting to Remote (GitHub)**
   * Created GitDemo repo on GitHub.
   * Linked & pushed:

git remote add origin <repo\_url>

git push -u origin master

**OUTPUT**:



**Hands-on 2: Git Ignore**

**Objective**

Ignore unwanted files like .log files and log/ folders.

**My Steps**

1. Created error.log file & log folder.
2. Made .gitignore file:

\*.log

log/

1. Verified with:

git status

These files were no longer showing as untracked.

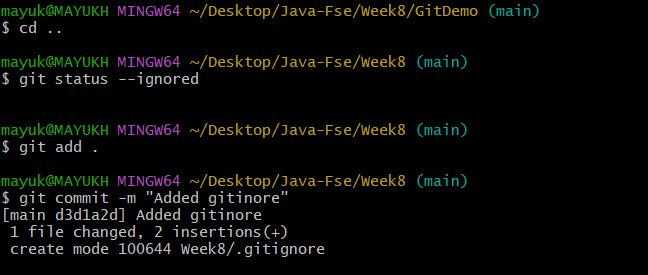
1. Committed .gitignore:

git add .gitignore

git commit -m "Added .gitignore to ignore logs"

git push origin master

**OUTPUT**:





**Hands-on 3: Branching & Merging**

**Objective**

Create a new branch, make changes, and merge into master.

**My Steps**

1. Created new branch:

git checkout -b GitNewBranch

1. Added a file:

echo "This is a branch file" > branchfile.txt

git add branchfile.txt

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

1. Switched to master:

git checkout master

1. Merged branch into master:

git merge GitNewBranch

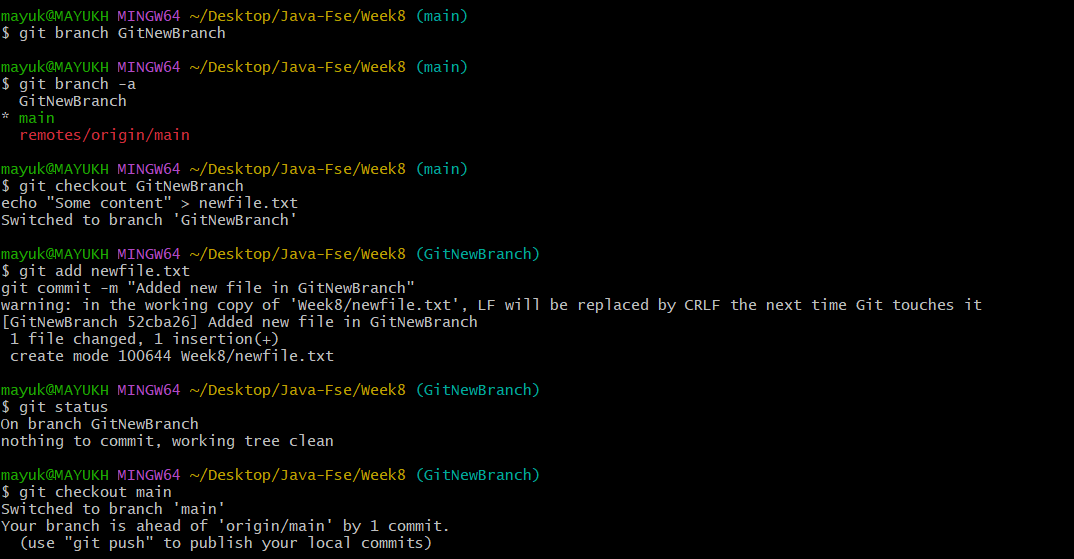
1. Pushed changes:

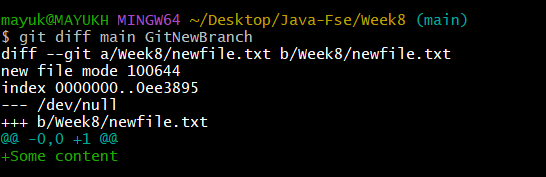
git push origin master

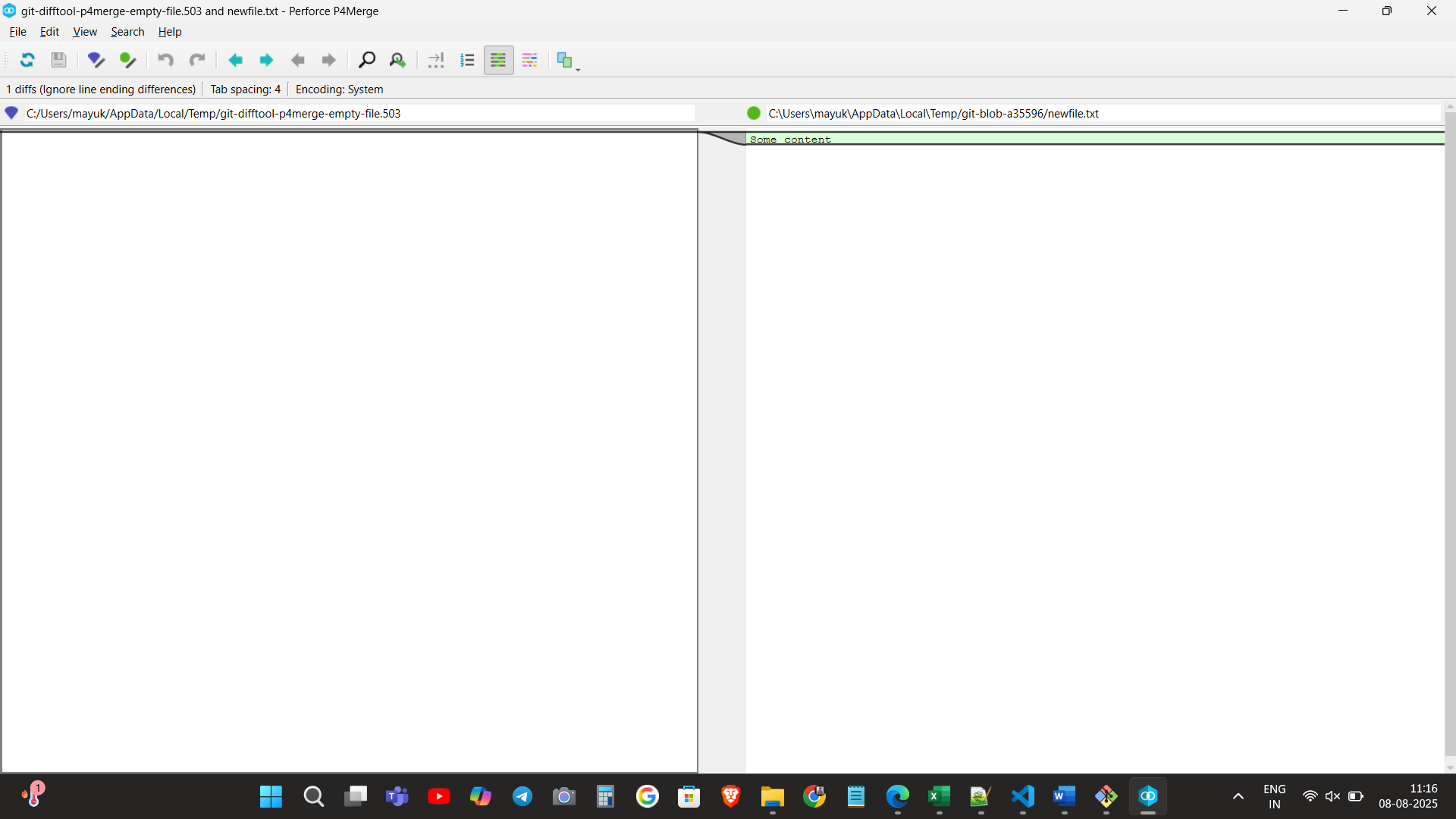
1. Deleted branch after merge:

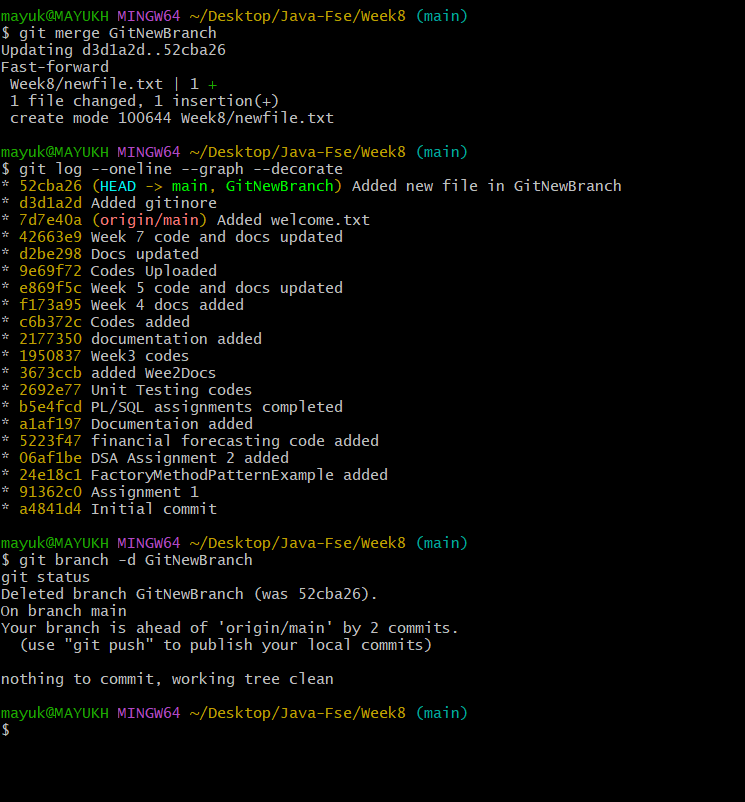
git branch -d GitNewBranch

**OUTPUT**:









**Hands-on 4: Merge Conflict Resolution**

**Objective**

Handle conflicts when the same file is changed in two branches.

**My Steps**

1. Made branch:git checkout -b GitWork

echo "Hello from branch" > hello.xml

git add hello.xml

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

1. Switched to master & created same file with different content:

git checkout master

echo "Hello from master" > hello.xml

git add hello.xml

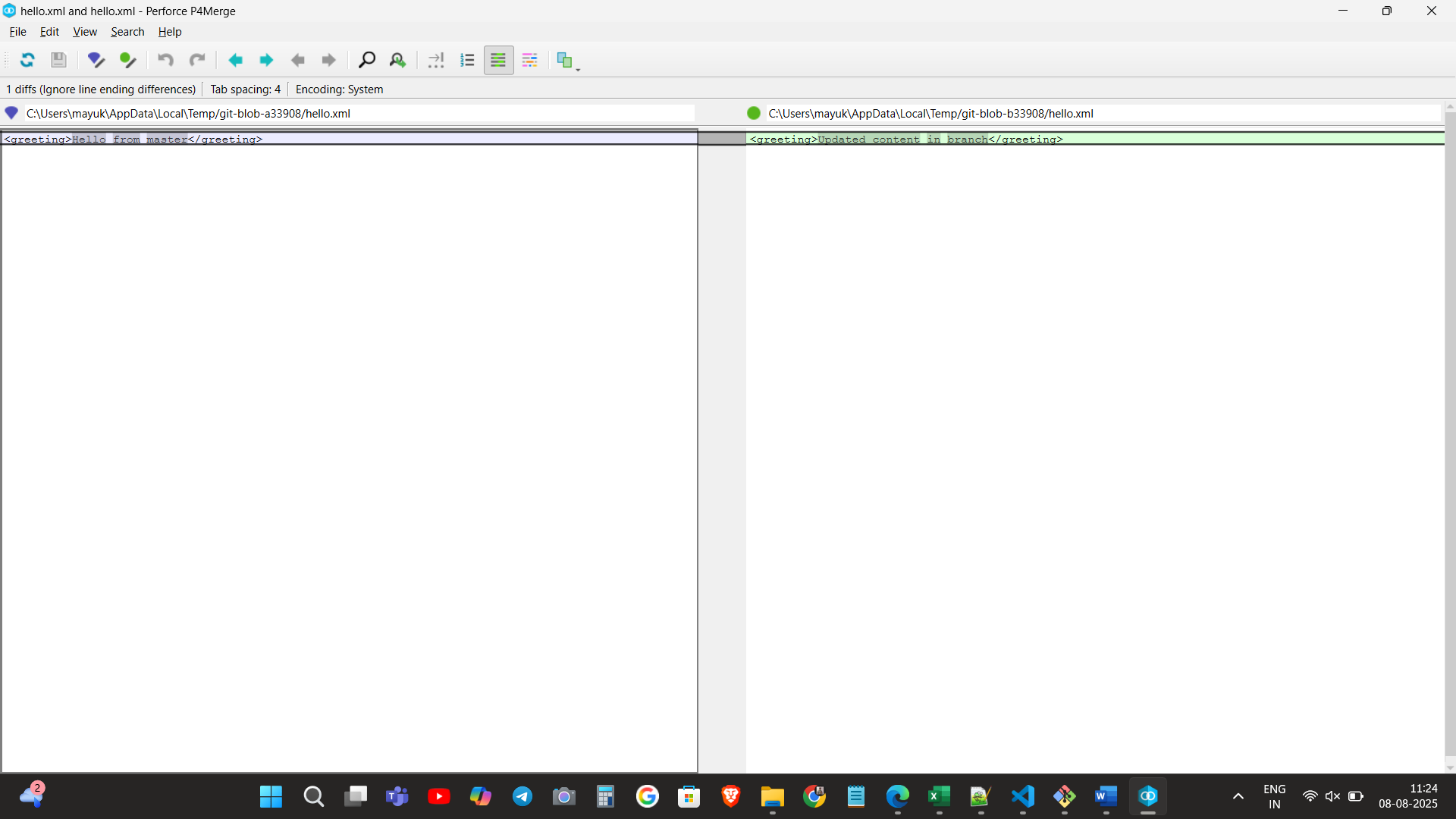
git commit -m "Added hello.xml in master"

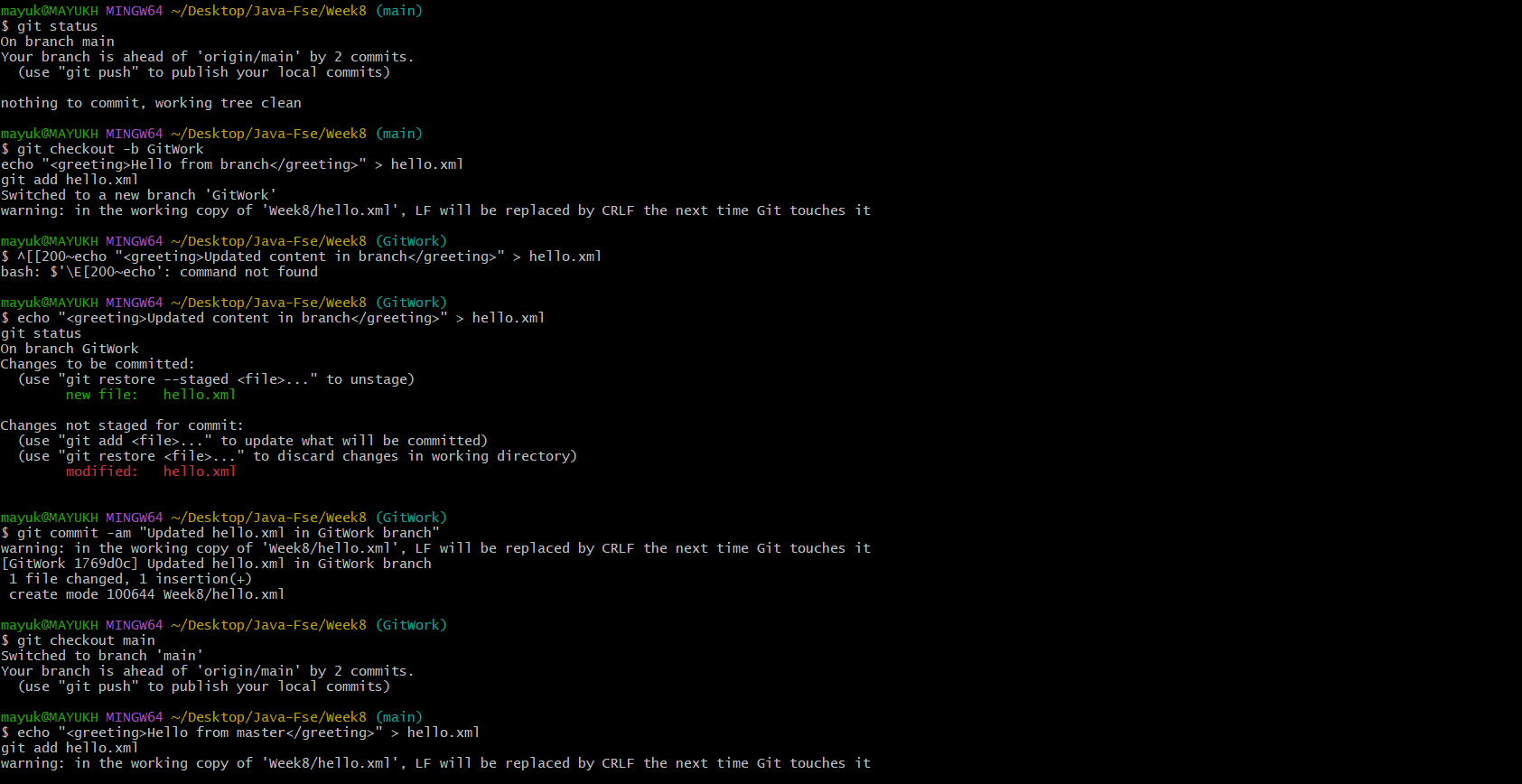
1. Merged branch into master — conflict appeared.
2. Opened file in Notepad++, chose correct content (or merged both).
3. Marked conflict as resolved:

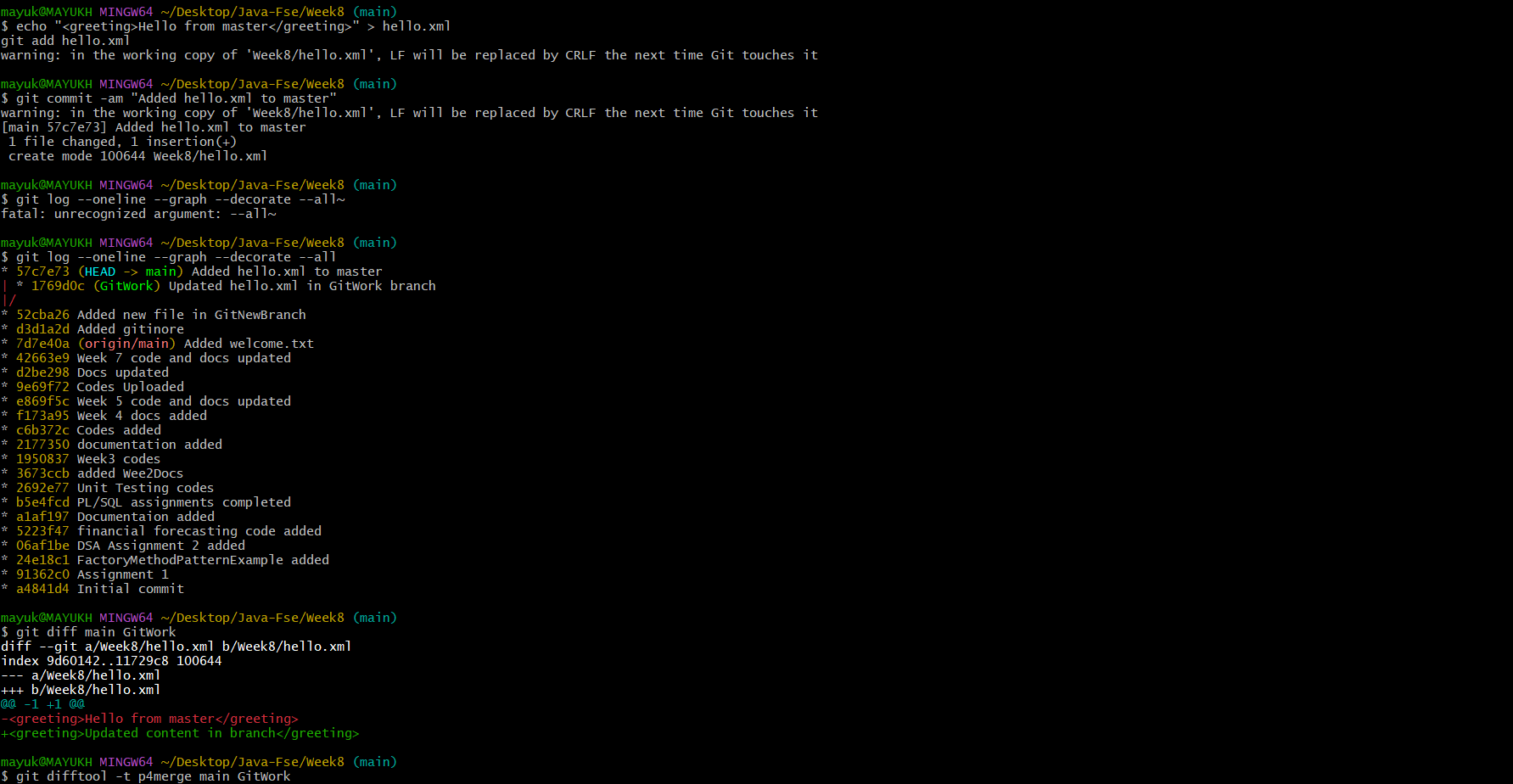
git add hello.xml

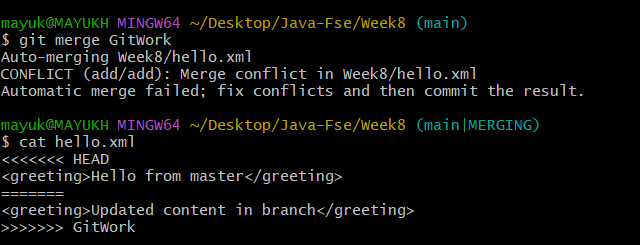
git commit -m "Resolved merge conflict in hello.xml"

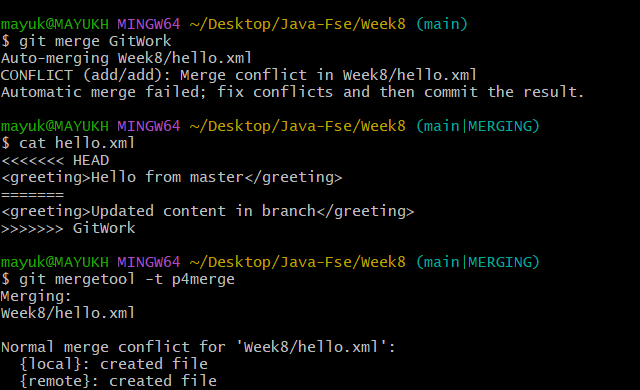
1. Added backup file to .gitignore and pushed.

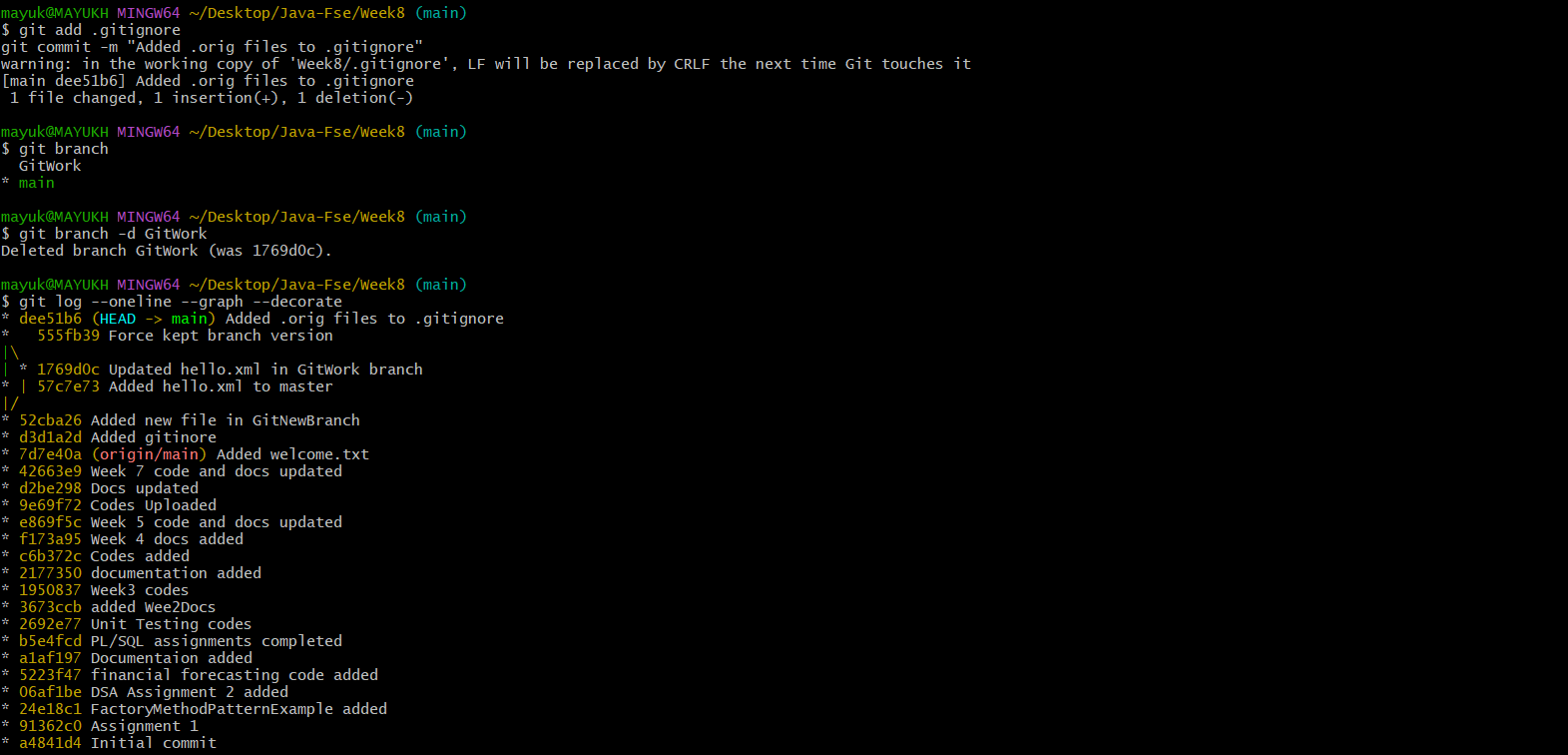
OUTPUT:  
  












**Hands-on 5: Clean up & Push to Remote**

**Objective**

Sync everything with remote repository.

**My Steps**

1. Verified clean state:

git status

1. Pulled latest changes:

git pull origin master

1. Pushed pending work (from branch Git-T03-HOL\_002 — created earlier in my case):

git checkout Git-T03-HOL\_002

git push origin Git-T03-HOL\_002

1. Merged into master & pushed:

git checkout master

git merge Git-T03-HOL\_002

git push origin master

**My Takeaways**

* Git is super powerful, but it’s also very strict — forget one command and it can get messy.
* .gitignore is your best friend for keeping repos clean.
* Merge conflicts aren’t scary once you understand what’s happening.
* Branching is a game-changer — safer than editing directly in master.
* Always pull before you push to avoid unnecessary conflicts.

**OUTPUT**:

