**Cognizant Deep Skilling - Digital Nurture 4.0 - Week 8**

**Git Hands-On Lab 1 – Basic Git Operations and Repository Management  
  
Step 1: Setup your machine with Git Configuration**

git --version

git config --global user.name "Ajithajothiraj"

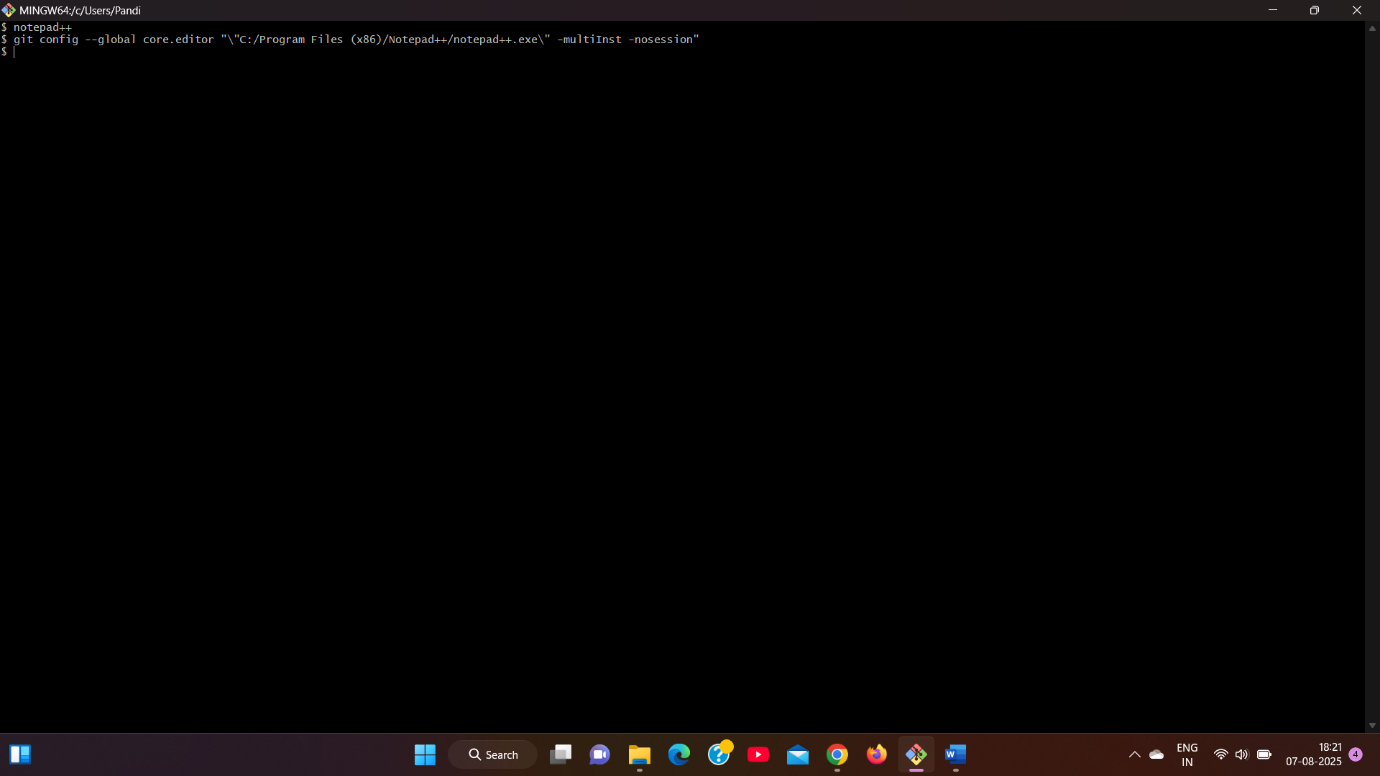
git config --global user.email "ajithajothiraj@gmail.com"

git config –list

**Step 2: Integrate notepad++.exe to Git and make it a default editor**

notepad++  
  
git config --global core.editor "\"C:/Program Files (x86)/Notepad++/notepad++.exe\" -multiInst -nosession"

**Output:**



**Step 3: Add a file to source code repository**

cd ~

mkdir GitDemo

cd GitDemo

git init  
  
echo "Welcome to Git Hands-On Lab" > welcome.txt

ls -al

cat welcome.txt  
  
git status

git add welcome.txt

git commit

git remote add origin https://github.com/Ajithajothiraj/GitDemo.git

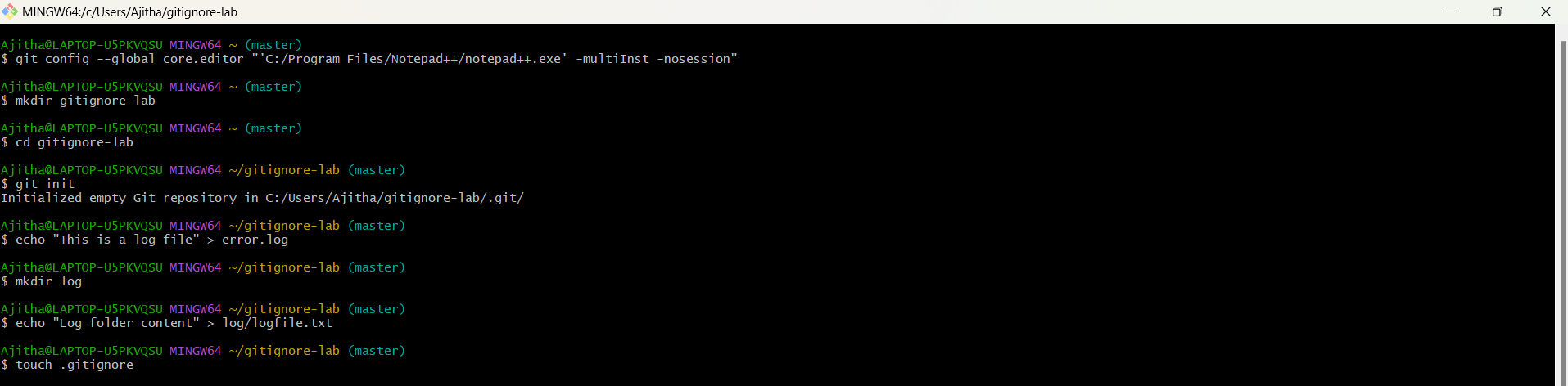
git pull origin master

git push origin master  
  
 **Output:** **Git Hands-On Lab 2 – Git Ignore  
  
Step 1: Create Unwanted Files**echo "This is a log file" > error.log

mkdir log

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

**Output:**

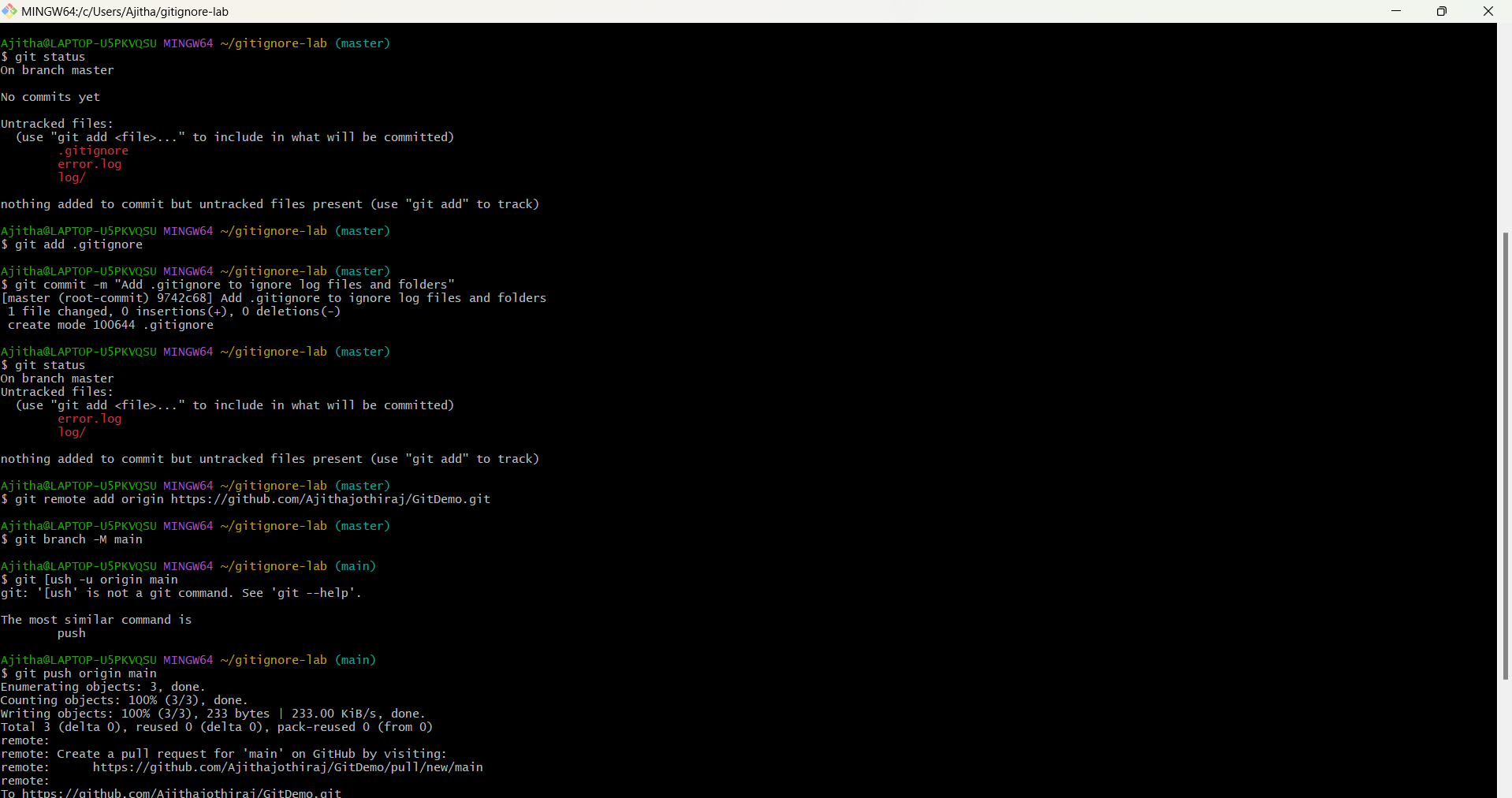


**Step 2: Create .gitignore File**touch .gitignore  
  
Add:  
  
\*.log

log/  
  
**Step 3: Check Git Status**git status  
  
**Step 4: Commit and Push**  
  
git add .gitignore

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

git push origin master **Output:**

 **Git Hands-On Lab 3 – Branching and Merging  
  
1. Created New Branch**git checkout -b GitNewBranch  
 **2. Listed Branches**git branch -a  
 **3. Created a File and Committed**echo "This file is created in GitNewBranch" > feature.txt

git add feature.txt

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

**4. Switched Back**git checkout master

git diff master GitNewBranch

**5. Configure & Use P4Merge for visual diff**

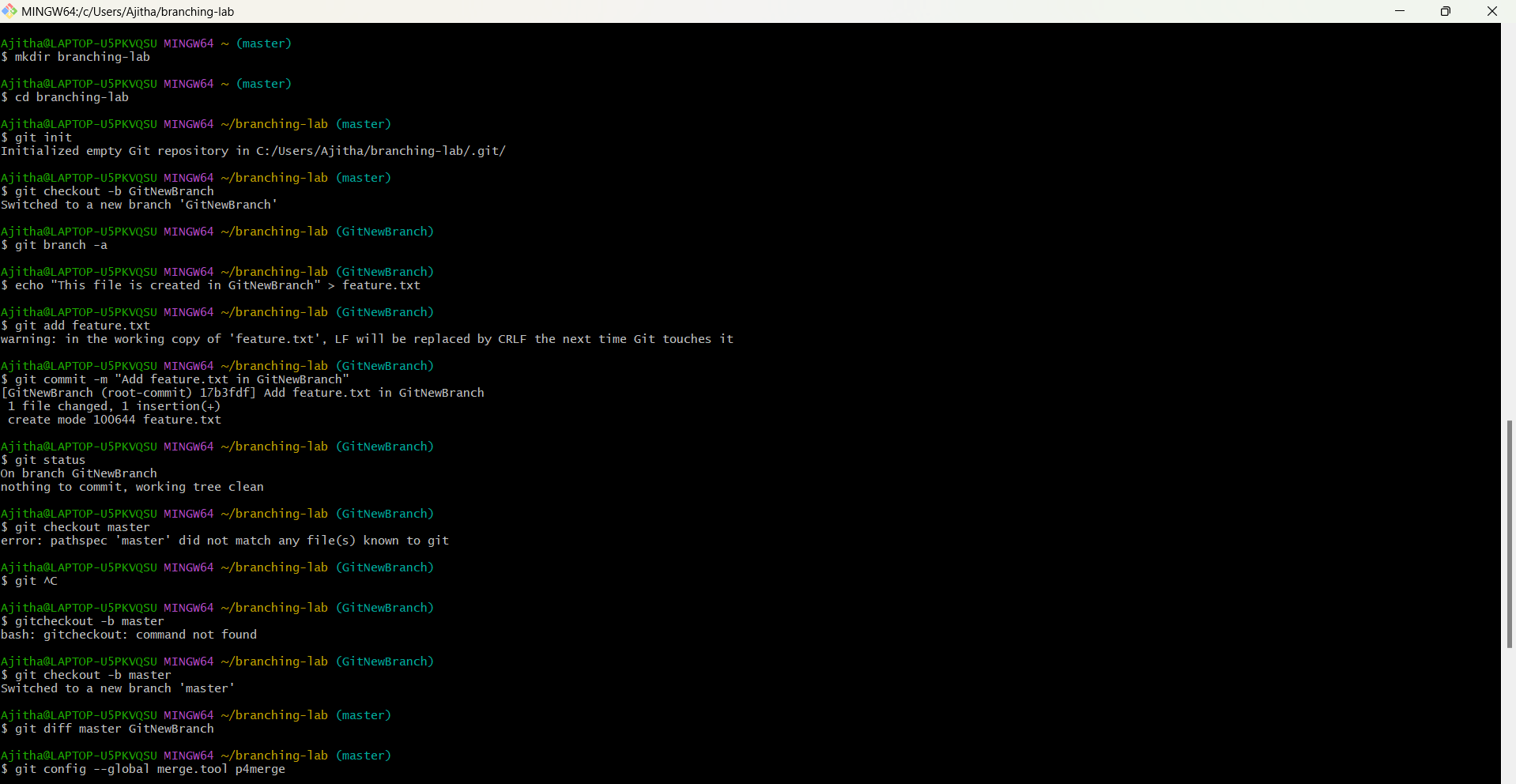
git config --global merge.tool p4merge

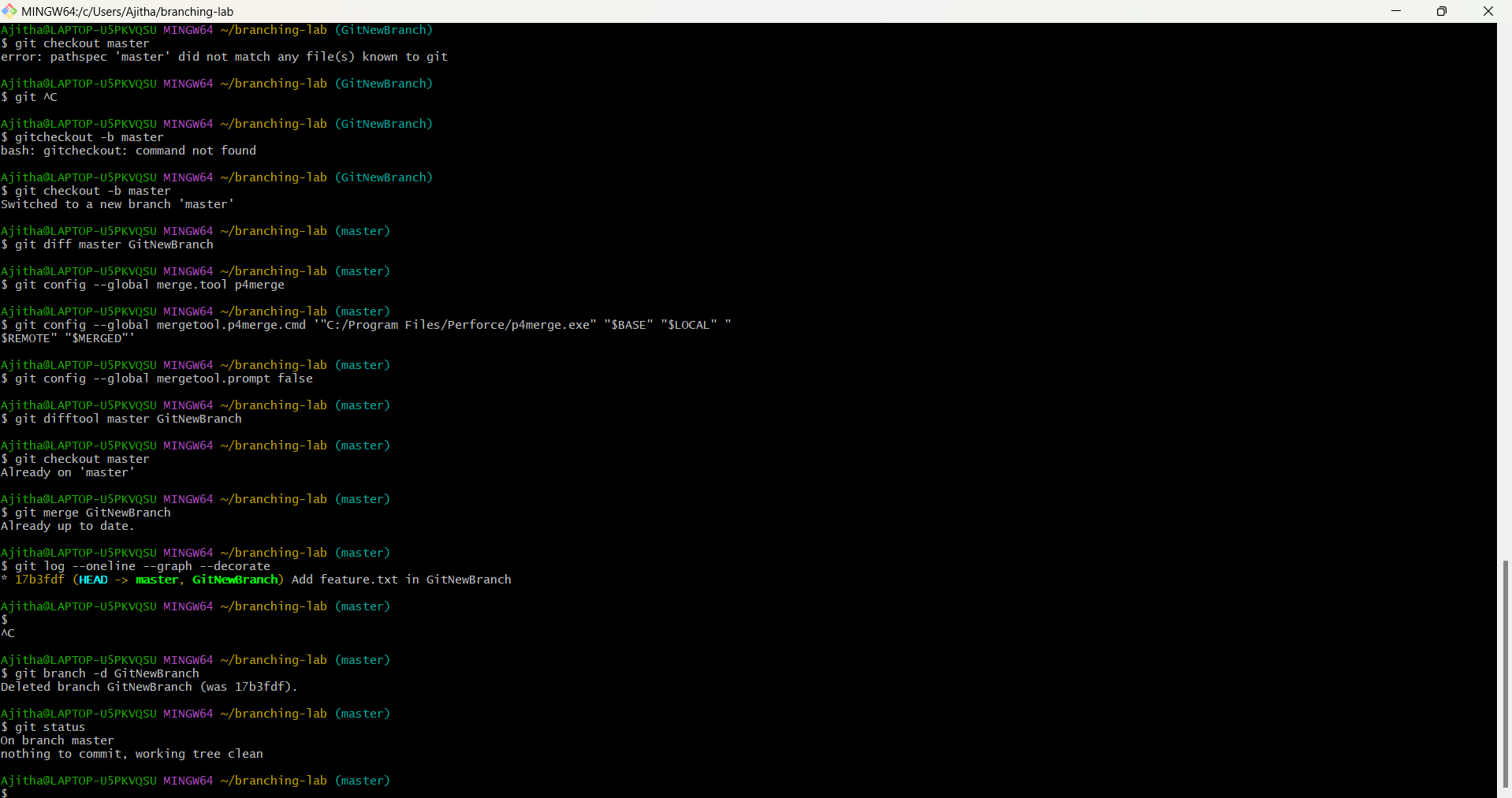
git config --global mergetool.p4merge.cmd '"C:/Program Files/Perforce/p4merge.exe" "$BASE" "$LOCAL" "$REMOTE" "$MERGED"'

git config --global mergetool.prompt false

**6. View commit history as a graph**

git log --oneline --graph --decorate

**Output:**



**Git Hands-On Lab 4 – Git Merge Conflict Resolution  
  
1. Created a branch GitWork**  
git checkout -b GitWork **2. Added hello.xml with unique content in GitWork**echo "<message>Hello from GitWork branch</message>" > hello.xml

git add hello.xml

git commit -m "Added hello.xml in GitWork branch"  
 **3. Switched to master and created conflicting file**git checkout master

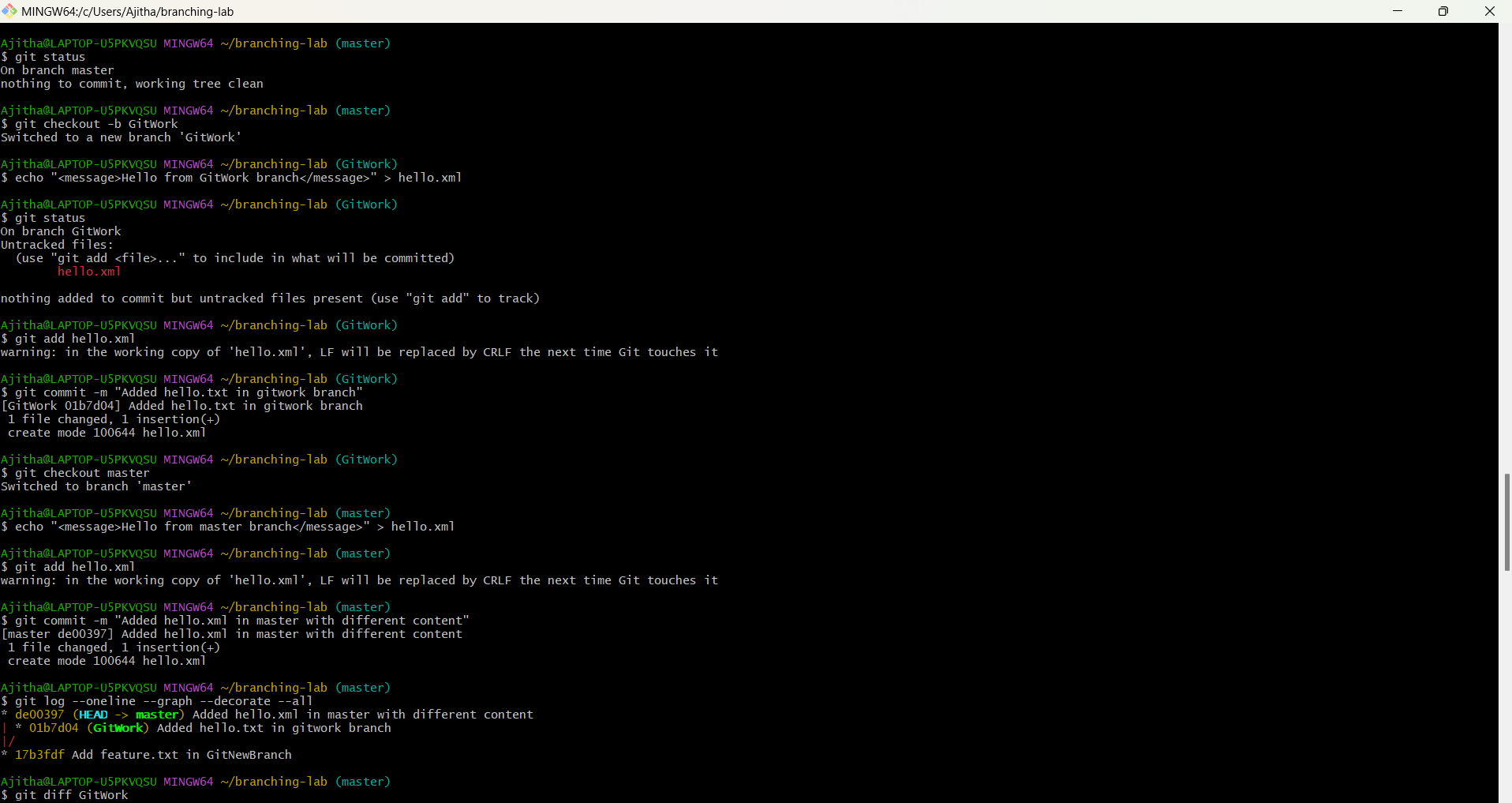
echo "<message>Hello from master branch</message>" > hello.xml

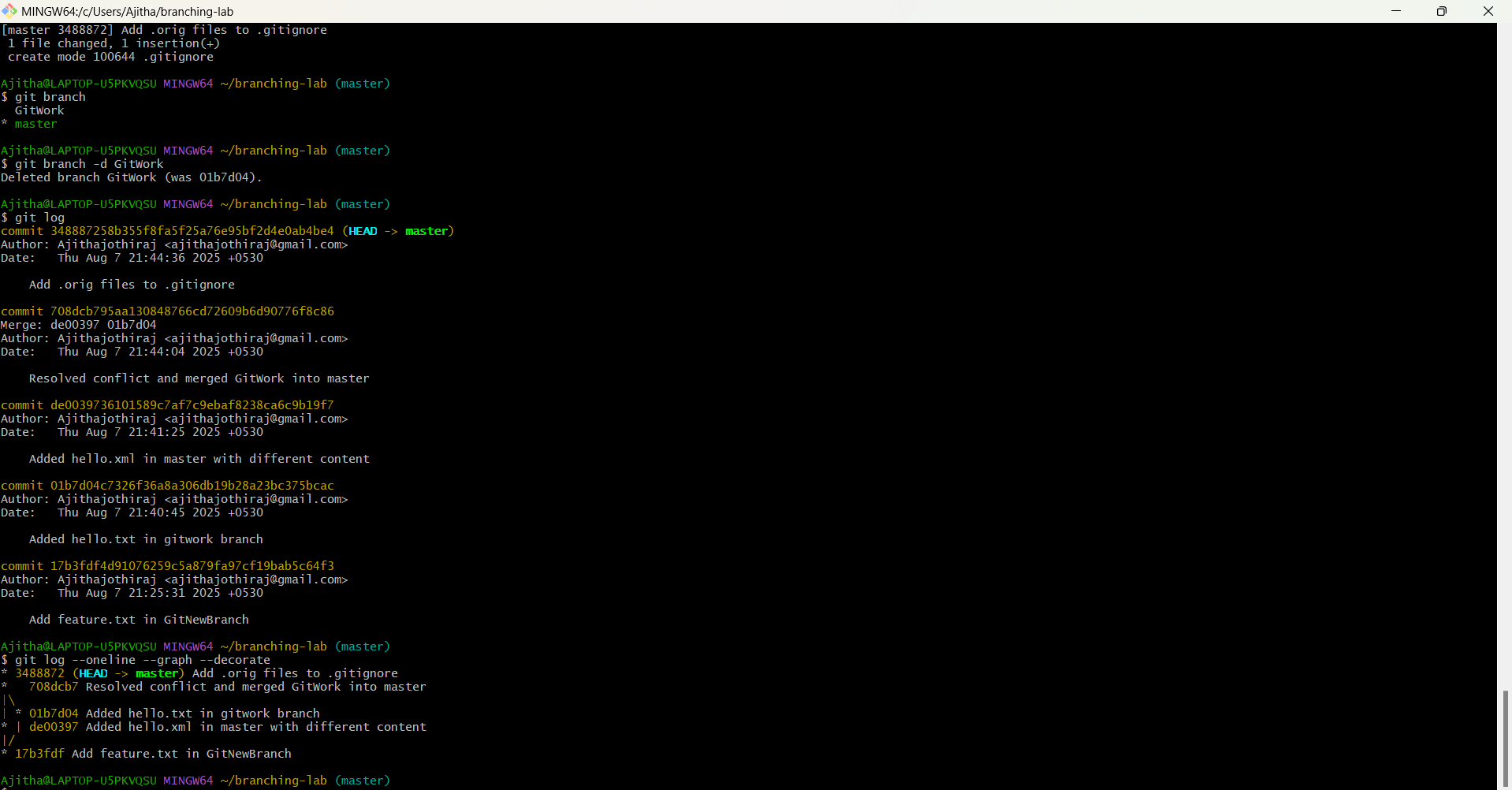
git add hello.xml

git commit -m "Added hello.xml in master with different content"  
  
**4. Merged and got conflict**git merge GitWork **5. Resolved conflict manually and committed**git add hello.xml

git commit -m "Resolved merge conflict in hello.xml"  
 **6. Observe git status and add backup file to .gitignore**git status

echo "\*.orig" >> .gitignore

**7. Deleted merged branch**git branch -d GitWork  
 **8. View the final graph**  
git log --oneline --graph --decorate **Output:**

 **Git Hands-On Lab 5 – Git Cleanup & Push Lab  
  
1. Switched to master branch**git checkout master  
 **2. Checked git status**git status  
 **3. Listed all branches**git branch -a **4. Pulled latest changes**git pull origin master **5. Pushed pending changes**git push origin master **6. Verified updates in GitLab repo  
  
Visited:**https://github.com/Ajithajothiraj/GitDemo.git

**Output:**