**Git Assignment 4 Documentation**

**Objective**

The purpose of this hands-on lab is to practice resolving merge conflicts when changes in a branch and the master branch overlap.

**Prerequisites**

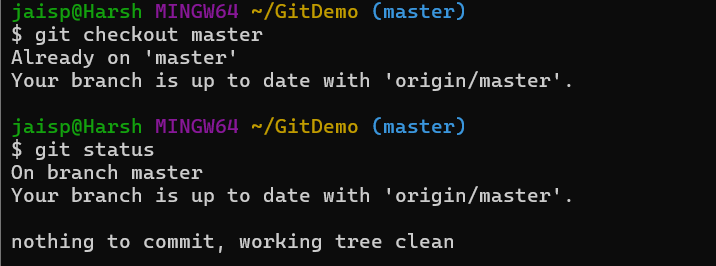
* Git Bash installed on your machine
* GitLab account with a repository already linked
* Local repository (e.g., GitDemo) set up

**Steps**

**Step 1: Ensure Master is Clean**

git checkout master

git status

****

**Step 2: Create Branch and Add File**

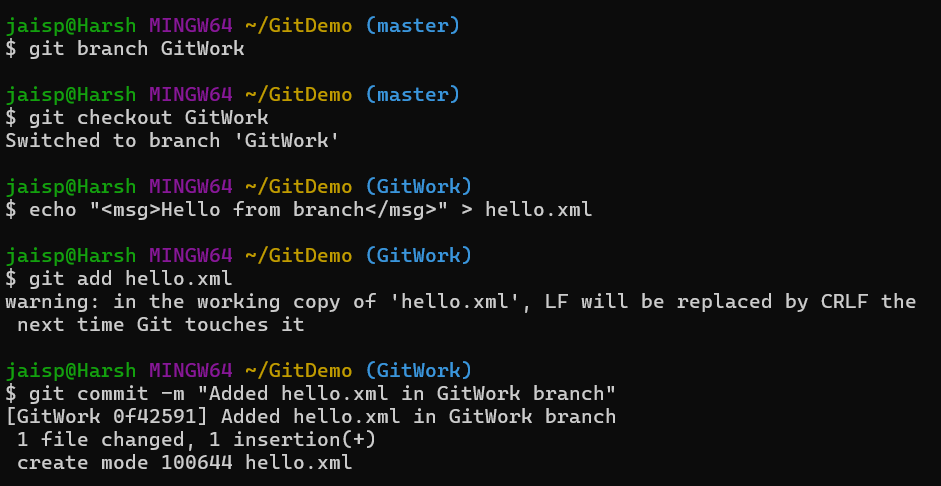
git branch GitWork

git checkout GitWork

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

git add hello.xml

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

****

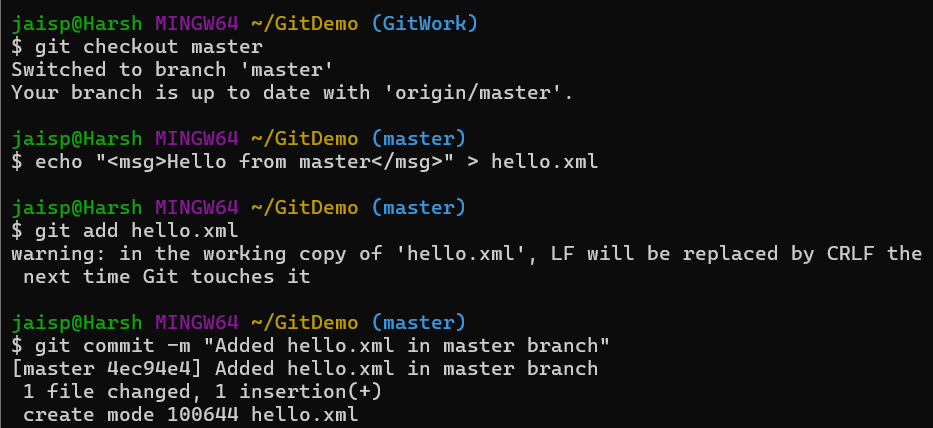
**Step 3: Switch to Master and Make Conflicting Change**

git checkout master

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

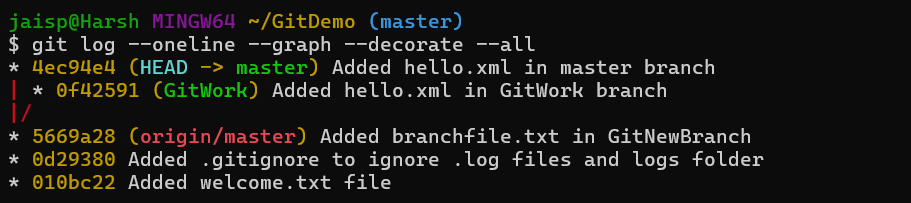
git add hello.xml

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

****

**Step 4: View Logs**

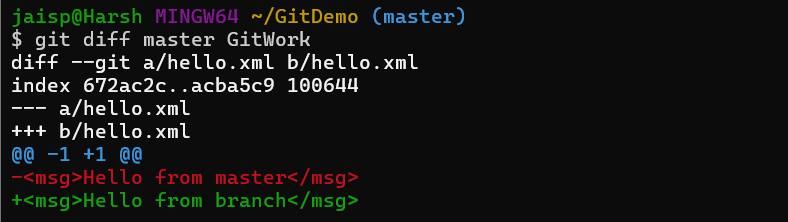
git log --oneline --graph --decorate --all

****

**Step 5: Check Differences**

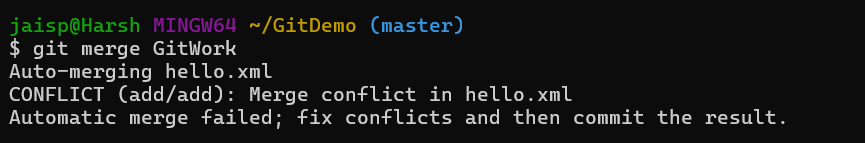
* Command-line diff:

git diff master GitWork

****

**Step 6: Merge and Trigger Conflict**

git merge GitWork

****

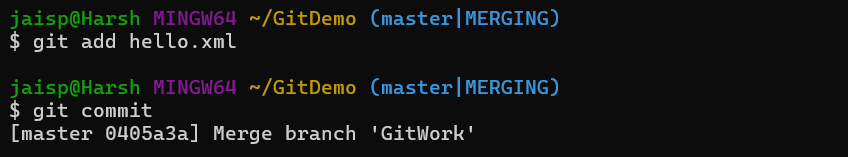
**Step 7: Resolve Conflict**

1. Open hello.xml in your editor
2. Remove conflict markers (<<<<<<<, =======, >>>>>>>).
3. Keep desired final content, for example:

<msg>Hello from both master and branch</msg>

1. Stage the resolved file:

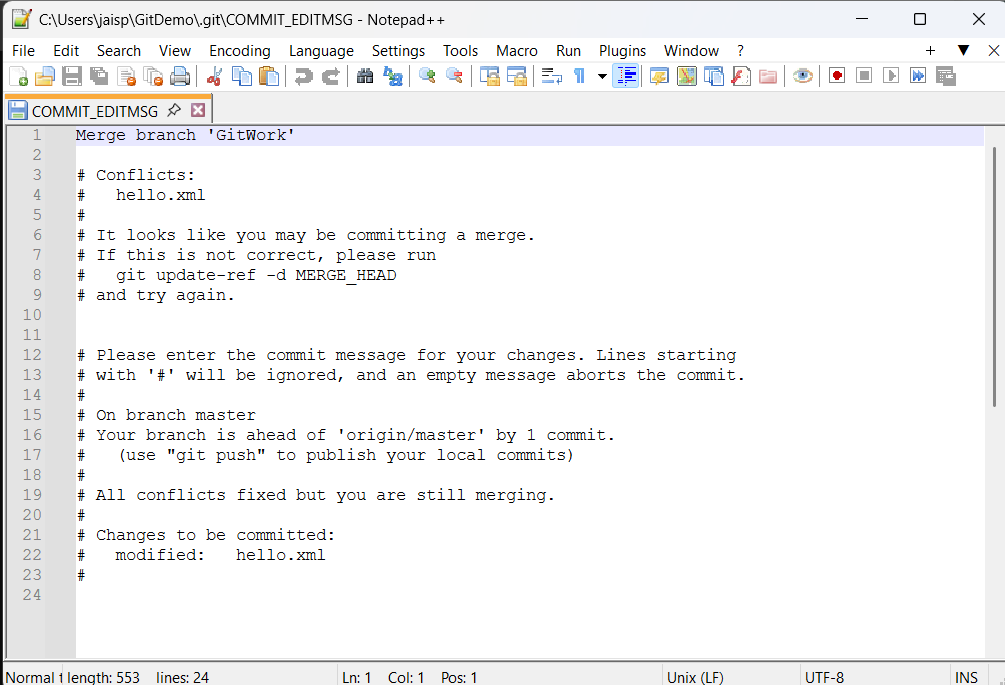
git add hello.xml

****



**Step 8: Commit Merge**

git commit

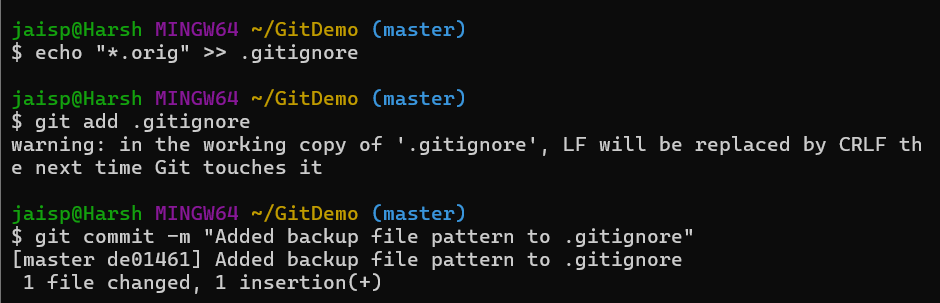
Save and close the merge commit message in Notepad++.  


**Step 9: Ignore Backup Files**

echo "\*.orig" >> .gitignore

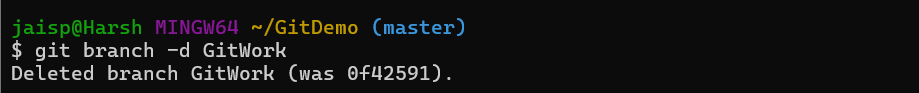
git add .gitignore

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

****

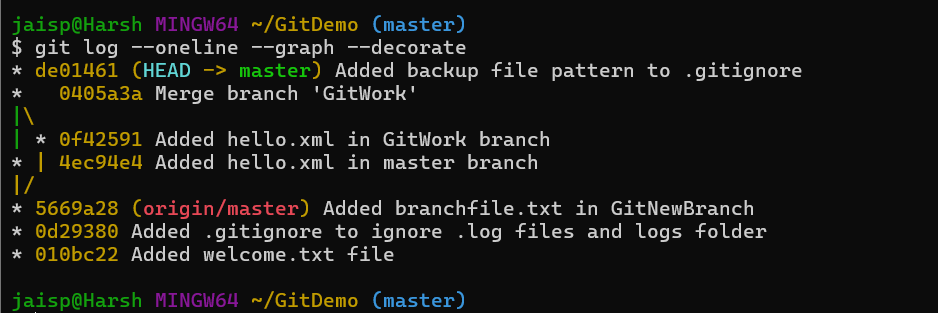
**Step 10: Delete Merged Branch**

git branch -d GitWork

****

**Step 11: View Final Log**

git log --oneline --graph --decorate

****