Exercise – 4

**Objective**

* Explain how to resolve conflicts during a Git merge operation

**In this lab, we will:**

* Simulate a merge conflict scenario
* Resolve the conflict using a 3-way merge tool (P4Merge)
* Commit the resolved changes and clean up

**Prerequisites**

* Git environment setup with **P4Merge** tool configured
* Hands-on ID: Git-T03-HOL\_001

**Estimated Time:** 30 minutes

**Step-by-Step Instructions**

**Step 1: Verify Clean Master**

git checkout master

git status

* Ensure that the working directory is clean and has no uncommitted changes.

**Step 2: Create a New Branch and Modify a File**

git checkout -b GitWork

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

git status

* hello.xml is created and added with initial content in GitWork branch.

**Step 3: Commit the File in GitWork**

git add hello.xml

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

**Step 4: Modify the Same File in Master**

git checkout master

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

git add hello.xml

git commit -m "Added hello.xml in master with different content"

**Step 5: Observe the Log**

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

* This shows divergence between GitWork and master due to different hello.xml changes.

**Step 6: View CLI and Visual Differences**

git diff master GitWork

git difftool master GitWork

* difftool opens P4Merge showing line-by-line changes visually.

**Step 7: Merge and Observe Conflict**

git merge GitWork

* Git will throw a **merge conflict** in hello.xml. The file will include conflict markers:

php-template

Copy code

<<<<<<< HEAD

<message>Hello from master</message>

=======

<message>Hello from branch</message>

>>>>>>> GitWork

**Step 8: Resolve Conflict Using P4Merge**

git mergetool

* This opens a **3-way merge tool (P4Merge)**.
* Resolve the conflict manually, save the file, and close the tool.

**Step 9: Commit Resolved File**

git add hello.xml

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

**Step 10: Ignore Backup Files and Commit .gitignore**

1. Add backup file pattern (e.g., .orig) to .gitignore:

echo "\*.orig" >> .gitignore

git add .gitignore

git commit -m "Ignored backup files in .gitignore"

**Step 11: Clean Up and Final Log**

1. **List All Branches**

git branch -a

1. **Delete Merged Branch**

git branch -d GitWork

1. **View Final Log**

git log --oneline --graph --decorate