**4. Git-HOL**

**Objectives**

* Understand how to resolve merge conflicts in Git.
* Learn how to merge branches that have changes to the same file.
* Practice using Git commands and optional 3-way merge tools (e.g., P4Merge).

**Prerequisites**

* Git installed and configured
* GitHub account (optional for remote push)
* Editor or merge tool (e.g., Notepad, VS Code, P4Merge)

**1. Initialize Repository**

mkdir GitConflictLab

cd GitConflictLab

git init

echo "# Git Conflict Lab" > README.md

git add README.md

git commit -m "Initial commit on master"

**2. Create Branch GitWork and Add File**

git checkout -b GitWork

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

echo "<info>This is some new content from GitWork</info>" >> hello.xml

git add hello.xml

git commit -m "Add hello.xml with GitWork content"

**3. Switch to Master and Create Conflict**

git checkout master

echo "<message>Conflicting content from master</message>" > hello.xml

git add hello.xml

git commit -m "Add hello.xml with master content"

**4. View Commit Log**

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

**5. Merge GitWork into master**

git merge GitWork

This caused a **merge conflict** in hello.xml.

**6. Conflict Markers (in hello.xml)**

<<<<<<< HEAD

<message>Conflicting content from master</message>

=======

<message>Hello from GitWork</message>

<info>This is some new content from GitWork</info>

>>>>>>> GitWork

**7. Resolve Conflict**

git mergetool

Then:

bash

CopyEdit

git add hello.xml

git commit -m "Resolved conflict between master and GitWork"

**8. Ignore Backup/Merge Files**

echo "hello\_\*.xml" >> .gitignore

git add .gitignore

git commit -m "Ignore conflict backup files"

**9. Clean Up**

git branch -d GitWork

git status

**10. Final Git Log**

git log --oneline --graph –decorate







