| 4. Git-HOL |
| --- |

1. Introduction

A merge conflict happens when Git cannot automatically merge differences between two branches. This usually occurs when the same lines in a file are changed differently in two branches, or one branch deletes a file that another branch modifies.

In this lab, we intentionally create a merge conflict and then resolve it manually by editing the file.

Objectives:

- Learn how to intentionally create a merge conflict

- Resolve a merge conflict by manually editing the file

- Commit the resolved changes and clean up the repository

Prerequisites:

- Completion of previous Git lab “Git-T03-HOL\_001”

- Working Git environment

- This example is shown in a Linux terminal

- We will use manual editing instead of P4Merge, which works on any system

-------------------------

PART 1: CREATING A MERGE CONFLICT

-------------------------

Step 1: Create a branch and make a change

First, ensure master is clean.

git status

Create a new branch named GitWork and switch to it.

git branch GitWork

git switch GitWork

In GitWork, create hello.xml with unique content.

echo '<?xml version="1.0" encoding="UTF-8"?>' > hello.xml

echo '<message>Hello from the GitWork Branch!</message>' >> hello.xml

Stage and commit the file.

git add hello.xml

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

Step 2: Create a competing change in master

Switch back to master.

git switch master

Create the same file hello.xml but with different content.

echo "<message>Hello from the Master Branch!</message>" > hello.xml

Stage and commit the file.

git add hello.xml

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

Step 3: Observe diverged history

At this point, master and GitWork each have different versions of hello.xml.

View the commit graph.

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

-------------------------

PART 2: RESOLVING THE CONFLICT

-------------------------

Step 1: Attempt the merge

While on master, try merging GitWork.

git merge GitWork

Git will stop and report a conflict because hello.xml differs in both branches.

Step 2: Inspect the conflict

Check status to see unmerged paths.

git status

Open hello.xml in a text editor. You will see conflict markers like:

<<<<<<< HEAD

<message>Hello from the Master Branch!</message>

=======

<message>Hello from the GitWork Branch!</message>

>>>>>>> GitWork

Explanation:

- Content between <<<<<<< HEAD and ======= is from the current branch (master)

- Content between ======= and >>>>>>> GitWork is from the branch being merged

- The conflict markers themselves are not valid file content

Step 3: Manually resolve the conflict

Edit hello.xml to keep both messages without conflict markers:

<message>Hello from the Master Branch!</message>

<message>And also, hello from the GitWork Branch!</message>

Save the file.

Step 4: Complete the merge

Stage the resolved file.

git add hello.xml

Commit the merge. Git will provide a default merge commit message.

git commit

-------------------------

PART 3: CLEANUP AFTER MERGING

-------------------------

Step 1: Handle backup files

Some merge tools create backups like hello.xml.orig.

Add these to .gitignore so they are not tracked.

echo "\*.orig" >> .gitignore

git add .gitignore

git commit -m "Update .gitignore to exclude merge tool backups"

Step 2: Delete the merged branch

Since GitWork has been merged, delete it locally.

git branch -d GitWork

Step 3: View final history

See the merge commit connecting the two branches.

git log --oneline --graph --decorate