**4. Git-HOL**

**Hands-on in this document**

**Merge Conflict Resolution – Hands-on Lab**

**Objectives**

* Learn to resolve conflicts during a merge.
* Practice conflict resolution using a 3-way merge tool (P4Merge).

**Estimated Time:** 30 minutes

**Prerequisites**

* You have completed **Git-T03-HOL\_001** (Branching & Merging lab).
* Git is installed and P4Merge is configured as the merge/diff tool.
* Local repository is already set up.
* **Do not use Cognizant credentials** for GitHub/GitLab.

**Step 1 – Ensure Master is Clean**

git checkout master

git status

The output should say: *nothing to commit, working tree clean*.

**Step 2 – Create a New Branch & Add File**

git branch GitWork

git checkout GitWork

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

git status

**Step 3 – Commit the Branch Changes**

git add hello.xml

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

**Step 4 – Switch to Master and Make a Different Change**

git checkout master

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

git add hello.xml

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

**Step 5 – Compare Before Merge**

1. **View commit history:**
2. git log --oneline --graph --decorate --all
3. **Check CLI differences:**
4. git diff master GitWork
5. **Check visual differences with P4Merge:**
6. git difftool master GitWork

**Step 6 – Merge and Trigger Conflict**

git merge GitWork

Git will show a conflict in hello.xml and mark it with:

<<<<<<< HEAD

<message>Hello from master branch</message>

=======

<message>Hello from GitWork branch</message>

>>>>>>> GitWork

**Step 7 – Resolve the Conflict**

1. **Open the 3-way merge tool (P4Merge):**
2. git mergetool
3. In P4Merge:
   * **Left** = master branch version
   * **Right** = GitWork branch version
   * **Middle** = Merge result (edit to final version)
4. Save and close.
5. **Mark conflict as resolved:**
6. git add hello.xml

**Step 8 – Commit the Merge**

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

**Step 9 – Update .gitignore for Backup Files**

echo "\*~" >> .gitignore

git add .gitignore

git commit -m "Ignore backup files"

**Step 10 – Clean Up**

1. **List branches:**
2. git branch
3. **Delete merged branch:**
4. git branch -d GitWork
5. **View final log:**
6. git log --oneline --graph --decorate