**Step 1: Setup Prerequisites**

**Description:**

* Git is installed and configured
* P4Merge tool is available for conflict resolution visualization
* Local Git repository is connected to a GitLab remote repository

**Step 2: Verify Clean Master State**

git checkout master

git status

* Output should indicate "nothing to commit, working tree clean"

**Step 3: Create a Branch and Add File**

git branch GitWork

git checkout GitWork

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

git add hello.xml

**Step 4: Commit Branch Changes**

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

**Step 5: Switch to Master and Make Conflicting Change**

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 6: View Logs**

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

**Step 7: Compare Differences**

**In CLI:**

git diff master GitWork

**In P4Merge:**

git mergetool

**Step 8: Merge and Trigger Conflict**

git merge GitWork

* Git will report a **merge conflict** in hello.xml

**Step 9: Resolve Conflict with 3-Way Merge Tool**

git mergetool

* In P4Merge, choose the appropriate changes from **master**, **branch**, or **both**, then save.

**Step 10: Commit Resolved Merge**

git add hello.xml

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

**Step 11: Ignore Backup Files**

echo "\*.orig" >> .gitignore

git add .gitignore

git commit -m "Updated .gitignore to exclude backup files"

**Step 12: Delete Branch After Merge**

git branch -d GitWork

**Step 13: View Final Log**

git log --oneline --graph --decorate