**4. Git-HOL**

## **Objectives**

* **Explain how to resolve the conflict during merge.**

To resolve a merge conflict in Git:

1. Open the conflicted file — Git marks conflicts like:

<<<<<<< HEAD

your changes

=======

incoming branch changes

>>>>>>> branch-name

1. Edit the file to keep the correct content, remove the <<<<<<<, =======, and >>>>>>> markers.
2. Mark the conflict as resolved:

git add <file>

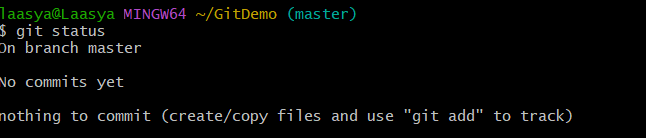
1. Complete the merge:

git commit

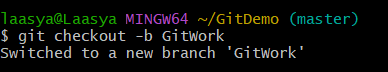
This commits the resolved version and finishes the merge.

**Hands-on**

1. Verify if master is in clean state.

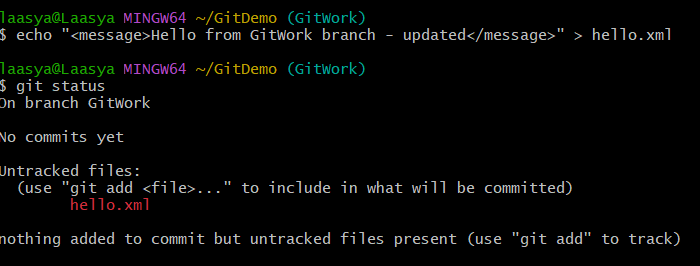


1. Create a branch **“GitWork”.** Add a file “hello.xml”.

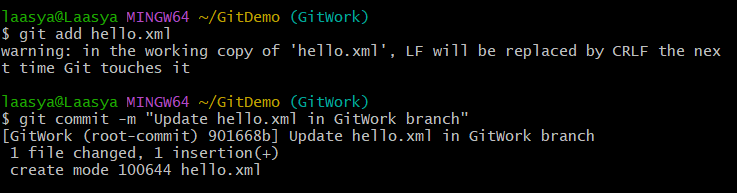




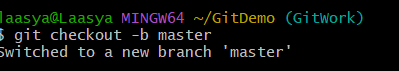
1. Update the content of “hello.xml” and observe the status



1. Commit the changes to reflect in the branch



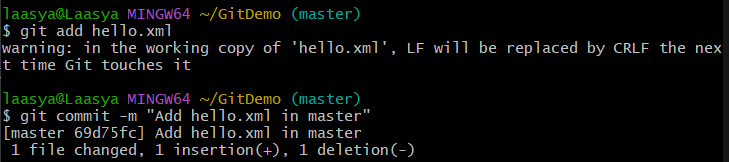
1. Switch to master.



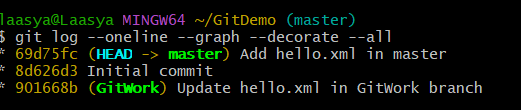
1. Add a file **“hello.xml”** to the master and add some different content than previous.



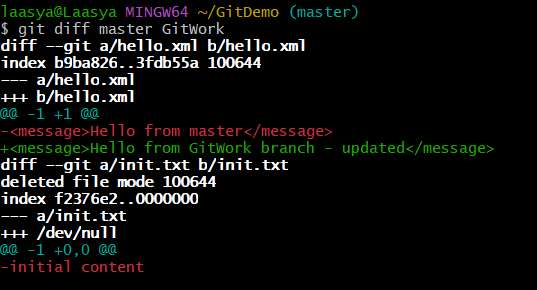
1. Commit the changes to the master



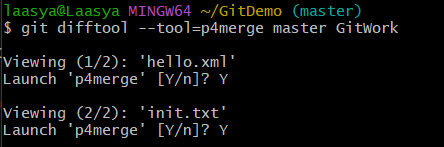
1. Observe the log by executing **“git log –oneline –graph –decorate –all”**

****

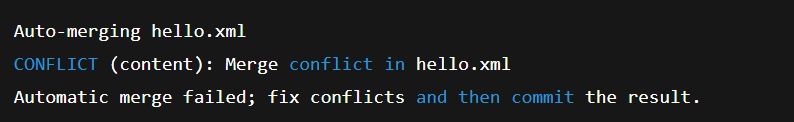
1. Check the differences with Git diff tool

****

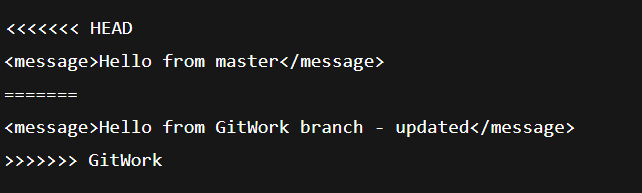
1. For better visualization, use P4Merge tool to list out all the differences between master and branch

****

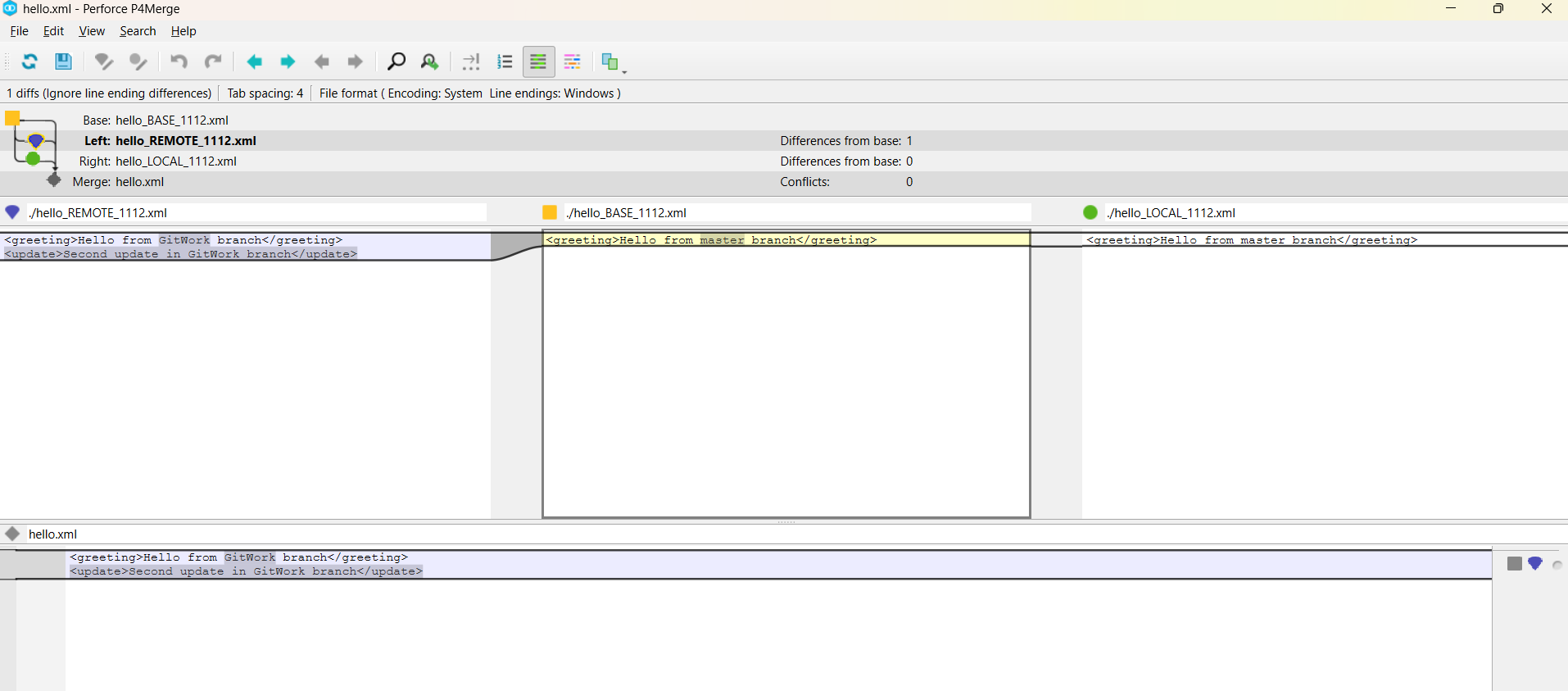
1. Merge the bran to the master

****

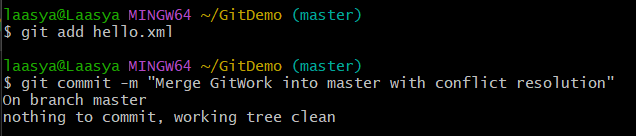
1. Observe the git mark up.

****

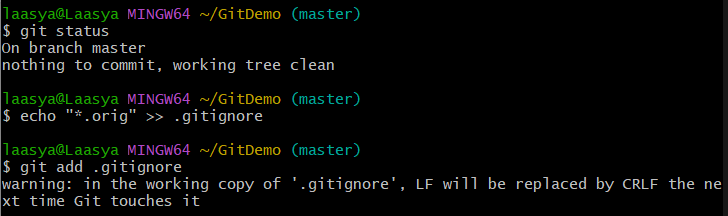
1. Use 3-way merge tool to resolve the conflict

****

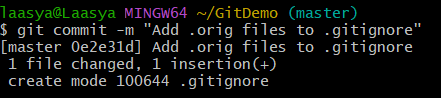
1. Commit the changes to the master, once done with conflict

****

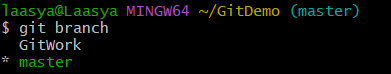
1. Observe the git status and add backup file to the .gitignore file.

****

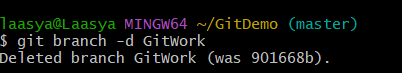
1. Commit the changes to the .gitignore

****

1. List out all the available branches

****

1. Delete the branch, which merge to master.

****

1. Observe the log by executing **“git log –oneline –graph –decorate”**

