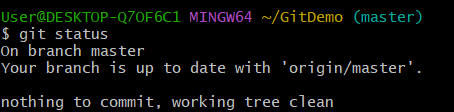
## **WEEK-8**

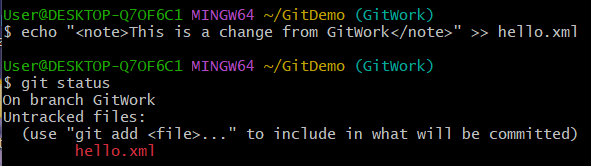
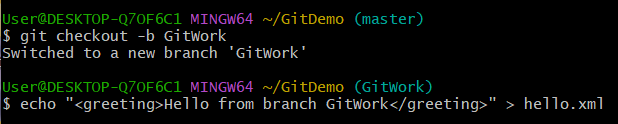
**4. Git – HOL**

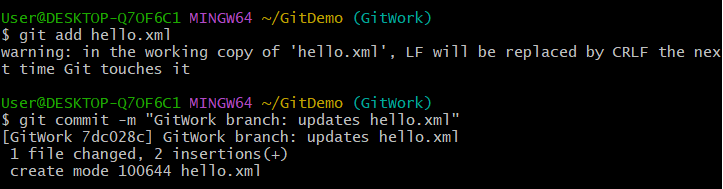
\* Explain how to resolve the conflict during merge.

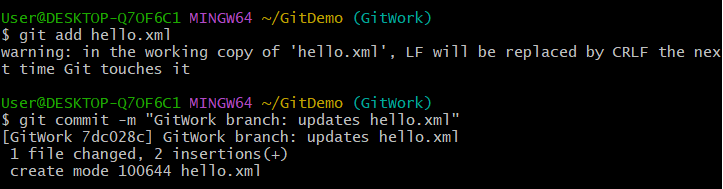
When Git detects a merge conflict, it means that two branches have modified the same part of a file differently, and Git cannot decide which version to keep.  
To resolve the conflict:

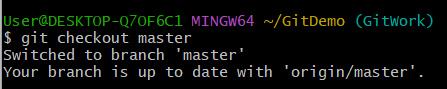
1. Identify the conflicting file – Git will mark the file in conflict after a merge attempt.
2. Open the file – You will see special conflict markers (<<<<<<<, =======, >>>>>>>) showing both versions.
3. Manually edit the file – Keep the correct content you want and remove the conflict markers.
4. Use a merge tool – Tools like P4Merge or VS Code’s merge editor make this easier by showing both versions side-by-side.
5. Mark the conflict as resolved – Run git add <filename> after fixing the file.
6. Commit the merge – Finalize the merge by running git commit, which saves the resolved file to the repository.



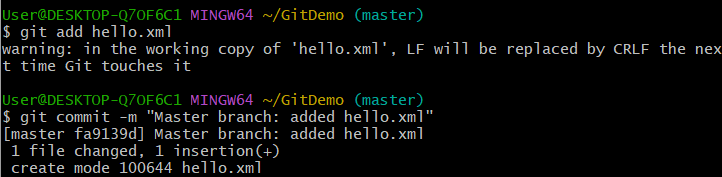


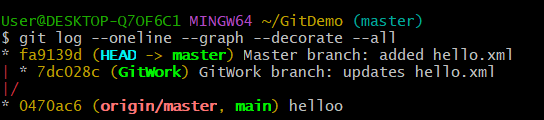


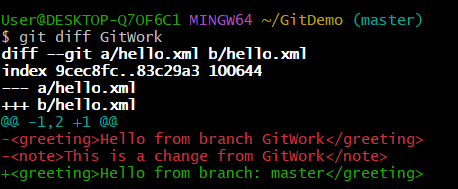




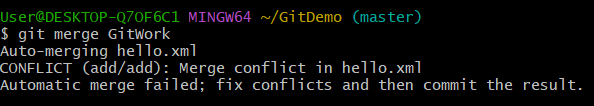




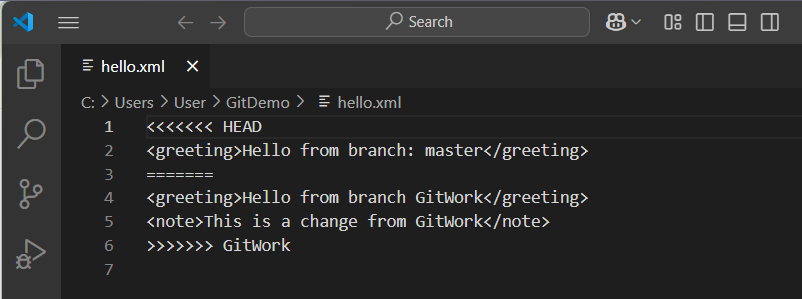




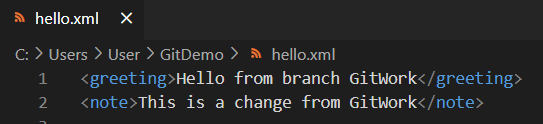


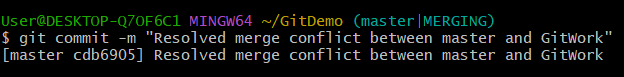


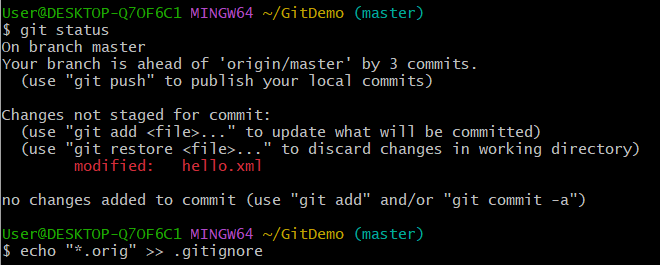


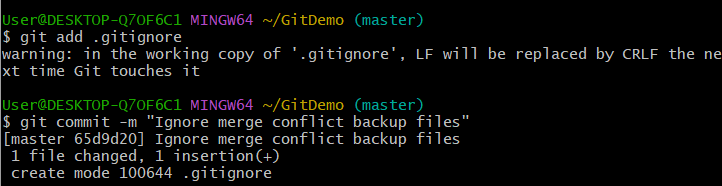
The above command opens hello.xml file in Visual Studio code.

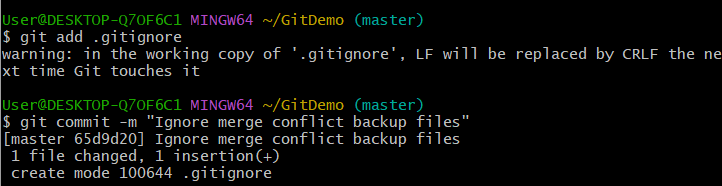


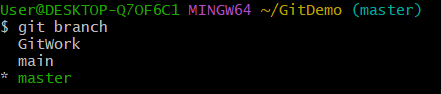




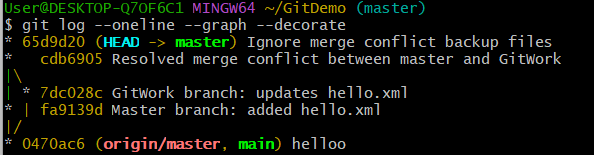












**Explanation:**

* Verified that the master branch was clean before starting.
* Created a new branch GitWork, added hello.xml, and committed changes.
* Switched to master, created another hello.xml with different content, and committed it.
* Compared differences between master and GitWork using both Git diff and P4Merge.
* Attempted to merge GitWork into master, encountered a conflict, and resolved it using the 3-way merge tool.
* Committed the resolved changes, updated .gitignore to exclude backup files, and deleted the merged branch.