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

**What Causes Merge Conflicts?**

Merge conflicts happen when Git can't automatically reconcile differences between two branches—usually because:

* The same lines in a file were changed in both branches.
* A file was deleted in one branch but modified in another.

**Steps to Resolve Merge Conflicts**

**1. Start the Merge**

*git merge feature-branch*

If there's a conflict, Git will pause the merge and mark the conflicted files.

**2. Identify Conflicted Files**

*git status*

Look for files marked as “unmerged” or “both modified.”

**3. Open the Conflicted File**

You'll see conflict markers like:

*<<<<<<< HEAD*

*Your changes from the current branch*

*=======*

*Incoming changes from the feature branch*

*>>>>>>> feature-branch*

**4. Manually Resolve the Conflict**

Edit the file to keep the desired changes. You can:

* Keep one side
* Combine both changes
* Rewrite the section entirely

Then remove the conflict markers (<<<<<<<, =======, >>>>>>>).

**5. Mark as Resolved**

*git add <filename>*

**6. Complete the Merge**

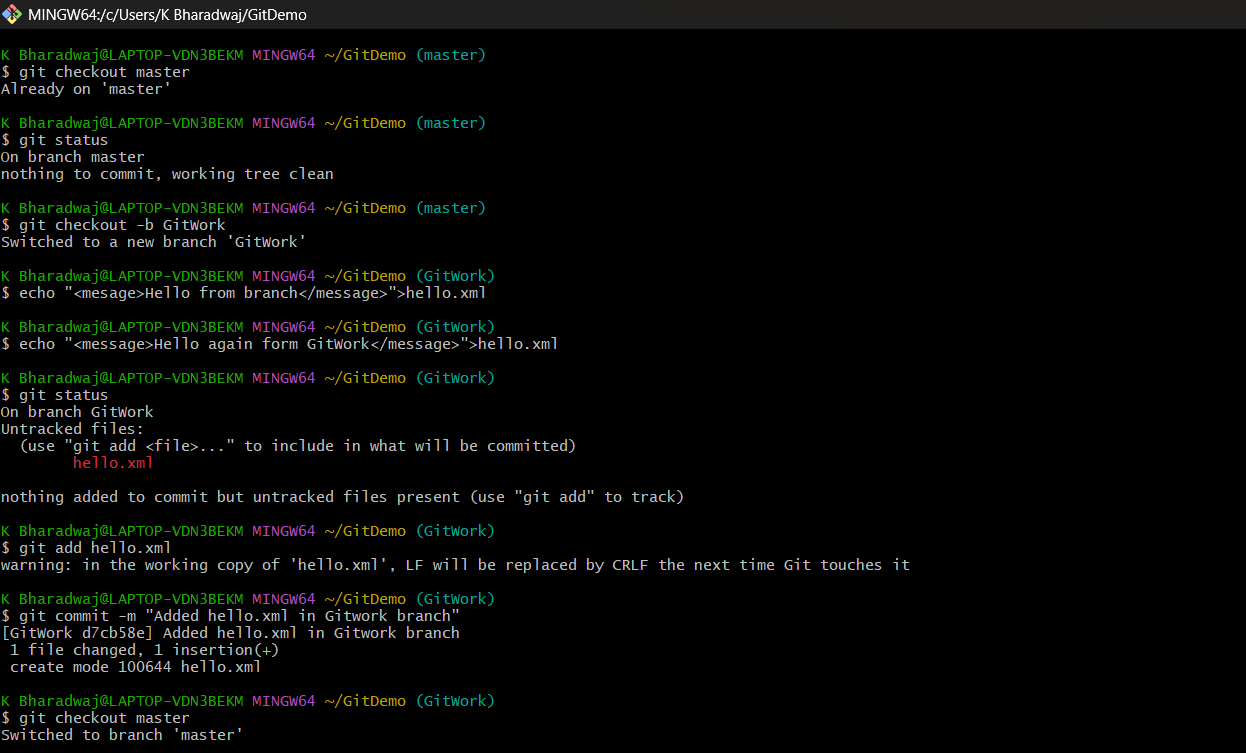
*git commit*

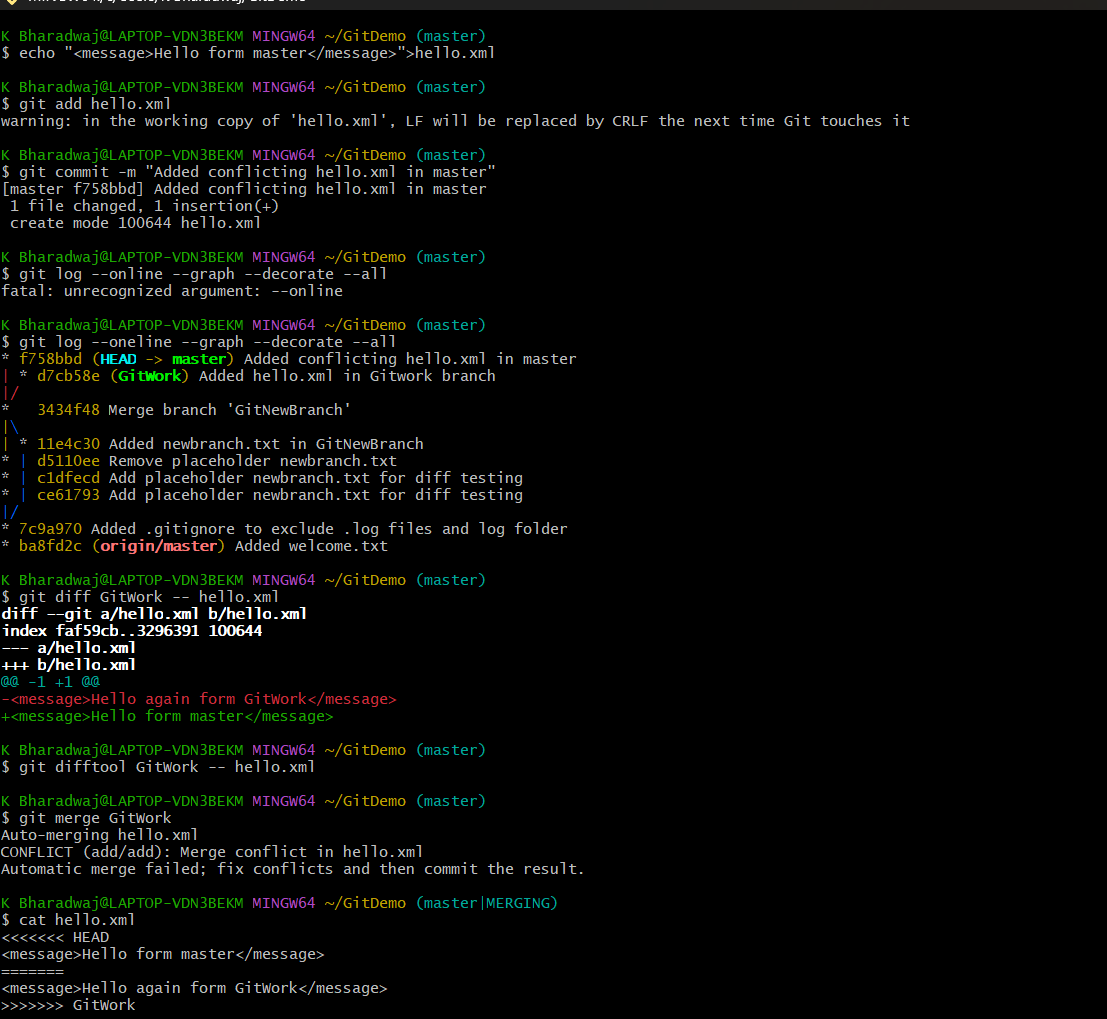
Git will use a default merge commit message unless you provide one.

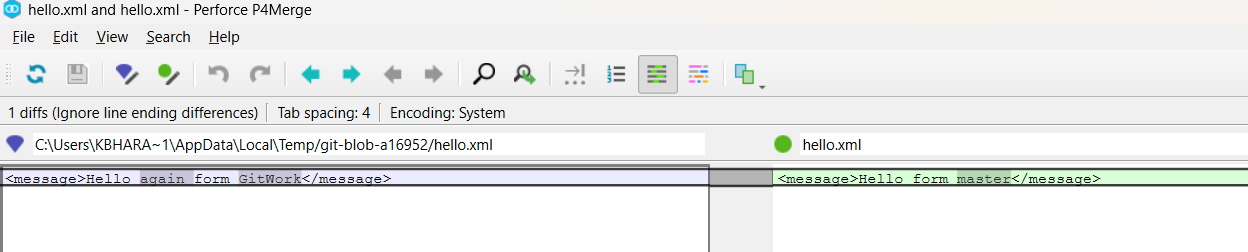
**Tips for Cleaner Conflict Resolution**

* Use a **difftool** like meld, kdiff3, or VS Code’s built-in merge editor.
* Run git log --merge to see commits from both sides.
* If overwhelmed, stash your changes first:
* git stash
* git merge feature-branch
* git stash pop

**OUTPUTS:**





**Difference**

**Resolving merge conflict by selecting both messages.**

