4. Git-HOL: How to Resolve Conflict During Merge

# What is a Merge Conflict?

A merge conflict in Git occurs when changes from two different branches affect the same part of a file and Git cannot automatically determine which change should be kept. This usually happens when both the current branch and the branch being merged have modified the same lines in a file.

# Steps to Resolve Merge Conflicts in Git

1. Attempt to Merge the Branches

Use the `git merge` command to combine changes from one branch into another.  
Example:  
git merge feature-branch

2. Identify the Conflict

If Git cannot merge the branches automatically, it will mark the conflicting files and pause the merge process.  
You will see a message like:  
'CONFLICT (content): Merge conflict in filename.txt'

3. Open the Conflicted File

Git adds special markers in the file to highlight the conflicting sections:  
<<<<<<< HEAD  
Changes from the current branch  
=======  
Changes from the branch being merged  
>>>>>>> feature-branch

4. Resolve the Conflict

Manually edit the file to combine or choose between the conflicting changes. Remove the conflict markers after resolution.

5. Use a Merge Tool (Optional)

Use tools like P4Merge to visually resolve conflicts.  
Command:  
git mergetool

6. Mark the Conflict as Resolved

Once you’ve resolved all conflicts, mark the file as resolved:  
git add filename.txt

7. Commit the Merge

Finally, commit the merge to complete the process:  
git commit -m "Resolved merge conflict in filename.txt"

# Summary

Merge conflicts are a normal part of collaborative development in Git. With a clear understanding of how to detect and resolve them — either manually or using tools like P4Merge — you can maintain a clean, stable project history.





