What is a Merge Conflict?

A **merge conflict** happens when **two branches modify the same part of the same file**, and Git can't decide which version to keep.

Example:

- main changes line 10 of file.txt
- feature also changes line 10 of the same file
- When merged, Git will throw a conflict

Syntax of Git Merge:

```
git merge <branch-name>
```

Merges <branch-name> into your **currently checked out branch**.

Example: Merge feature/login into main

- git checkout main
- git merge feature/login

Result: All changes from feature/login get merged into main.

Example: Merge with a commit message

```
git merge feature/login -m "Merging login feature"
```

Only works for **non-fast-forward** merges.

Scenario: You are on branch **main**, and you want to merge changes from **feature/login.**

Step 1: Project Setup

- mkdir git-merge-demo
- cd git-merge-demo
- git init

Step 2: Create a File in main

- echo "Welcome to the Homepage!" > index.html
- git add index.html
- git commit -m "Initial homepage in main branch"

Step 3: Create a new branch feature and modify same file

- git checkout -b feature

Step 4: Modify the file

- echo "Welcome to the New Feature Page!" > index.html
- git add index.html
- git commit -m "Update homepage in feature branch"

Step 5: Switch back to main and modify the same file

- git checkout main

Step 6: Modify the file

- echo "Welcome to the Main Site!" > index.html
- git add index.html
- git commit -m "Update homepage in main branch"

Step 7: Merge the feature branch into main

- git merge feature

Step 8: Conflict occurs! Output:

```
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Step 8: Open index.html to resolve the conflict

```
<<<<<<< HEAD
Welcome to the Main Site!
=======
Welcome to the New Feature Page!
>>>>>>> feature
```

Step 9: Edit to resolve:

```
Welcome to the Main Site with New Feature!
```

Step 10: Finalize the merge

- git add index.html
- git commit -m "Resolved conflict between main and feature branch"

Done! The two versions are now merged.

Let’s check the summary of commits>

Step 11: Compare commits in both branches (before merge)

While you're on main, run:

- git log main --oneline
- git log feature –oneline

Step 12: **See what’s different between branches**

- git log main..feature --oneline
- Shows commits in feature **not** in main.

- git log feature..main --oneline
- Shows commits in main **not** in feature.

Step 13: Visual comparison of both branches

- git log --oneline --graph --all -decorate
- Shows a **graphical view** of both branches and how they diverged and merged.

Step 14: After merge: View merge commit

- git log --oneline
- The top commit will be something like:

```
e7d8a3f Merge branch 'feature' into main
```

Step 15: Show which commits came from feature

- git log -merges
- Shows **only merge commits** in your history.