how to resolve the conflict during merge.

1. Why Merge Conflicts Happen

A merge conflict occurs when **two branches** have changes in the same file **at the same location**, and Git can't decide which version to keep.

Example:

- You change line 5 in file.txt in **branch A**.
- Someone else changes line 5 in the same file.txt in **branch B**.
- When merging, Git sees both changes and says, "I don't know which one to keep."

2. Identify the Conflict

When you run:

git merge <branch_name>

If there's a conflict, Git will show:

pgsql

Auto-merging file.txt

CONFLICT (content): Merge conflict in file.txt

Automatic merge failed; fix conflicts and then commit the result.

3. Check Conflicted Files

Use:

git status

You'll see something like:

yaml

Unmerged paths:

both modified: file.txt

4. Open the File

Inside file.txt, Git marks the conflicting section like this:

txt

<<<<<<< HEAD

This is your branch's version

=======

This is the other branch's version

>>>>>>> branch_name

Meaning:

- **Between <<<<<<< HEAD and =======** → Your changes (from the branch you are on)
- **Between ======= and >>>>>>> branch_name** → Changes from the branch you're merging in

5. Resolve the Conflict

You have 3 main choices:

1. **Keep your version**
   Delete the other branch's part.
2. **Keep the other branch's version**
   Delete your part.
3. **Manually combine**
   Merge both versions logically.

Example resolution (merged version):

This is the combined version after resolving conflict

Remove **all conflict markers** (<<<<<<<, =======, >>>>>>>) after fixing.

6. Mark as Resolved

Once you've fixed the file:

git add file.txt

7. Complete the Merge

git commit

Git will open the default commit message for the merge; you can keep or edit it.

8. Verify

Run:

git log --graph --oneline --all

You should see the merge completed.