How to Resolve Merge Conflicts in Git

Resolving merge conflicts in Git involves several steps to ensure that changes from different branches are correctly integrated. Here's a detailed guide:

Step-by-Step Guide to Resolve Merge Conflicts

1. Identify the Conflict:

When you try to merge branches, Git will notify you of conflicts. It marks the files with conflicts, indicating which parts need resolution.

Auto-merging file.txt

CONFLICT (content): Merge conflict in file.txt

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

2. Open the Conflicted File:

Open the file with conflicts. Git uses conflict markers to show where the conflicts are.

<<<<<<< HEAD

This is some content from the current branch.

=======

This is some content from the branch being merged.

>>>>>>> branch-name

3. Resolve the Conflict:

Manually edit the file to combine the changes. Decide what the final content should be and remove

the conflict markers. For example:

This is the combined content after resolving conflicts.

4. Add the Resolved File:

After resolving the conflicts, stage the resolved file using `git add`.

git add file.txt

5. Commit the Merge:

Commit the merge to finalize it. Git suggests a default message for merge commits.

git commit

6. Continue the Merge (If using Rebase):

If you were performing a rebase and encountered conflicts, you need to continue the rebase process after resolving conflicts.

git rebase --continue

Additional Tips

Abort Merge or Rebase: If you decide not to merge or rebase, you can abort the process.

git merge --abort # To abort a merge

git rebase --abort # To abort a rebase

Use a Visual Merge Tool: Git supports visual merge tools that can help resolve conflicts more intuitively.

```
git mergetool
```

Check the Status: After resolving conflicts and before committing, check the status to ensure all conflicts are resolved.

```
git status
```

Summary

1. Identify conflicts during merge or rebase.
2. Open and manually resolve the conflicts in the files.
3. Stage the resolved files.
4. Commit the changes.
5. Continue the merge or rebase if necessary.

Using these steps, you can effectively resolve merge conflicts in Git, ensuring a smooth integration of changes from different branches.