**Merge conflicts & Rebase**

**Step 1: Initial Setup**

1. **Create a new repository**:

git init merge-conflict-demo

cd merge-conflict-demo

1. **Create a file**:

echo "Line 1: This is the first line." > example.txt

git add example.txt

git commit -m "Initial commit with example.txt"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 2: Create a Feature Branch**

1. **Create and switch to a new branch**:

git checkout -b feature

1. **Modify the file**:

echo "Line 2: This line is added by the feature branch." >> example.txt

git add example.txt

git commit -m "Add a line in the feature branch"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 3: Modify the File in main**

1. **Switch back to the main branch**:

git checkout main

1. **Modify the same file in a conflicting way**:

echo "Line 2: This line is added by the main branch." >> example.txt

git add example.txt

git commit -m "Add a conflicting line in main branch"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 4: Merge the Feature Branch into main**

1. **Start the merge**:

git merge feature

1. **Observe the conflict**: Git will display a message like:

Auto-merging example.txt

CONFLICT (content): Merge conflict in example.txt

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

**Step 5: Resolve the Conflict**

1. **Open the conflicting file (example.txt)**:

<<<<<<< HEAD

Line 2: This line is added by the main branch.

=======

Line 2: This line is added by the feature branch.

>>>>>>> feature

The HEAD section contains the changes from main, and the feature section contains the changes from the feature branch.

1. **Manually resolve the conflict**: Choose one version, combine them, or create a new one:

Line 2: This line is added by both branches.

1. **Mark the conflict as resolved**:

git add example.txt

1. **Commit the resolution**:

git commit -m "Resolve merge conflict in example.txt"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 6: Verify the Merge**

1. **Check the commit history**:

git log --oneline

1. **Check the file contents**:

cat example.txt

The resolved version should be present.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Key Takeaways**

* A **merge conflict** occurs when Git cannot automatically reconcile changes.
* You must manually edit the conflicting file(s), add them, and create a new merge commit.
* Use tools like git status to identify unresolved conflicts and graphical tools like GitKraken, VS Code, or git merge tool to simplify conflict resolution.

**Merge Conflict vs. Rebase**

Both **merge** and **rebase** are Git operations used to integrate changes from one branch into another. However, they handle conflicts differently and serve distinct purposes.

**Key Differences**

| **Aspect** | **Merge** | **Rebase** |
| --- | --- | --- |
| **Definition** | **Combines the histories of two branches, creating a merge commit.** | **Moves (reapplies) commits from one branch onto another.** |
| **Commit History** | **Preserves the full history of both branches, including the merge commit.** | **Produces a linear history, rewriting commit hashes.** |
| **Conflict Handling** | **Conflicts are resolved during the merge process.** | **Conflicts are resolved during the rebase process (one commit at a time).** |
| **Result** | **A branch with a merge commit reflecting the integration.** | **A branch with a rewritten, linear history.** |

**Example: Merge Conflict vs. Rebase**

Create a repository and simulate conflicts.

1. **Initialize the repository**:

git init merge-vs-rebase-demo

cd merge-vs-rebase-demo

1. **Add a file**:

echo "Line 1: Initial content." > example.txt

git add example.txt

git commit -m "Initial commit"

1. **Create a feature branch**:

git checkout -b feature

echo "Line 2: Feature branch changes." >> example.txt

git add example.txt

git commit -m "Feature branch changes"

1. **Switch to main and make conflicting changes**:

git checkout main

echo "Line 2: Main branch changes." >> example.txt

git add example.txt

git commit -m "Main branch changes"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Scenario 1: Merge**

1. **Merge the feature branch into main**:

git merge feature

1. **Result**: A merge conflict arises:

Auto-merging example.txt

CONFLICT (content): Merge conflict in example.txt

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

1. **Resolve the conflict**: Open example.txt:

<<<<<<< HEAD

Line 2: Main branch changes.

=======

Line 2: Feature branch changes.

>>>>>>> feature

Resolve manually (e.g., combine changes):

Line 2: Main and feature branch combined changes.

1. **Finish the merge**:

git add example.txt

git commit -m "Resolve merge conflict"

1. **Check the history**:

git log --oneline

The history includes the merge commit:

abc123 Resolve merge conflict

def456 Main branch changes

ghi789 Feature branch changes

jkl012 Initial commit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Scenario 2: Rebase**

1. **Rebase the feature branch onto main**:

git checkout feature

git rebase main

1. **Result**: A rebase conflict arises:

Applying: Feature branch changes

CONFLICT (content): Merge conflict in example.txt

1. **Resolve the conflict**: Open example.txt:

<<<<<<<

Line 2: Main branch changes.

=======

Line 2: Feature branch changes.

>>>>>>>

Resolve manually:

plaintext

Copy code

Line 2: Main and feature branch combined changes.

1. **Continue the rebase**:
2. git add example.txt

git rebase --continue

1. **Check the history**:

git log --oneline

The history is now linear:

plaintext

Copy code

xyz123 Feature branch changes

def456 Main branch changes

jkl012 Initial commit

8

| **When to Use Merge vs. RebaseUse Case** | **Preferred Operation** |
| --- | --- |
| **Preserve branch history** | **Merge** |
| **Work in a team environment** | **Merge** |
| **Clean, linear history (e.g., personal)** | **Rebase** |
| **Avoid merge commits** | **Rebase** |

**Summary:**

* **Merge** keeps all history intact, including a merge commit.
* **Rebase** rewrites history to create a linear sequence of commits.
* Conflicts occur in both scenarios but are handled at different stages:
  + Merge: During the merge process.
  + Rebase: During commit application.

Bottom of Form