Git Merge and Conflict Resolution Guide

Step 1: Initialize a Git Repository

First, create a new directory for your project and initialize a Git repository.

```
mkdir merge-conflict-demo
cd merge-conflict-demo
git init
```

Step 2: Create a File and Commit It

Create a new file and make an initial commit.

```
echo "This is the initial content of the file." > example.txt
git add example.txt
git commit -m "Initial commit"
```

Step 3: Create and Switch to a New Branch

Create a new branch and switch to it.

```
git checkout -b feature-branch
```

Step 4: Make Changes in the New Branch and Commit

Edit the `example.txt` file and commit the changes.

```
echo "This is a change in the feature branch." >> example.txt
git add example.txt
git commit -m "Change in feature branch"
```

Step 5: Switch Back to the Main Branch

Switch back to the `main` branch (or `master` if your default branch is named that).

```
git checkout main
```

Step 6: Make Conflicting Changes in the Main Branch and Commit

Edit the `example.txt` file again in a way that will conflict with the changes made in the feature branch, then commit the changes.

```
echo "This is a conflicting change in the main branch." >> example.txt
git add example.txt
```

```
git commit -m "Conflicting change in main branch"
```

Step 7: Attempt to Merge the Feature Branch into the Main Branch

Try to merge the `feature-branch` into the `main` branch.

```
git merge feature-branch
```

Step 8: Resolve the Merge Conflict

Git will detect the conflict and mark the conflicting section in the `example.txt` file. Open the file and you will see something like this:

```
This is the initial content of the file.
<<<<<<< HEAD
This is a conflicting change in the main branch.
=======
This is a change in the feature branch.
>>>>>>> feature-branch
```

Edit the file to resolve the conflict. You might decide to keep both changes, one of the changes, or rewrite the conflicting part entirely. Here is an example of a possible resolution:

```
This is the initial content of the file.
This is a change in the feature branch.
This is a conflicting change in the main branch.
```

Step 9: Add the Resolved File and Commit the Merge

After resolving the conflict, add the file and commit the merge.

```
git add example.txt
git commit -m "Resolved merge conflict between main and feature-branch"
```

Step 10: Verify the Merge

You can now verify the merge was successful by checking the commit history.

```
git log --oneline --graph
```

This will show a linearized history of your commits, including the merge commit.