Lab: (Command Line) Resolving Merge Conflicts

Estimated time: 10 minutes

> Note: This lab assumes that you are using a command line. If you would prefer to use Sourcetree, there are separate instructions.

In this lab, you will:

1. Create branches that contain a merge conflict.
2. Merge the branches, resolving the merge conflict.

1: Create branches that contain a merge conflict.

1. Create a local repository named `projectd`.

2. Create a commit in your `projectd` repository with a `fileA.txt` file containing a string "feature 1". The commit message should be "add feature 1". This commit should be on the `master` branch.

3. Create and checkout a branch off of the latest master commit named "feature2".

4. In your local repository, **create a commit** on the `feature2` branch with the following:

   - modify `fileA.txt`, adding "feature 2" directly under the line "feature 1"
   - add a commit message of "add feature 2"

5. Checkout the `master` branch.

6. Create a commit on the `master` branch with the following:

   - modify `fileA.txt`, adding "feature 3" directly under the line "feature 1"
   - add a commit message of "add feature 3"

   > Congratulations, you have created branches that contain a merge conflict. The `master` branch and the `feature2` branch have modified the same hunk of `fileA.txt` in different ways.

2: Merge the branches, resolving the merge conflict.

1. Verify that the `master` branch is checked out.

2. Execute `git merge feature2` and attempt to merge in the `feature2` branch. You should

see that there is a merge conflict.

3. Execute `git status` to see that Git has modified `fileA.txt`.

4. View the `fileA.txt` file. Notice the conflict markers in the file. That is the part of the merge that Git couldn't automatically resolve.

5. Rather than fix the conflict right now, abort the merge process by executing `git merge --abort`.

6. Verify that you are back to the state before the merge attempt, with no uncommitted files in the working tree.

7. This time, let's resolve the merge conflict. Repeat the merge attempt. You should again see a merge conflict.

8. **Edit `fileA.txt`** to resolve the merge conflict. Remove the conflict markers and make sure the file contains three lines of text: "feature 1", "feature 2" and "feature 3".

9. **Add** `fileA.txt` to the staging area so that the fixed version of the file is part of the merge commit.

10. **Commit** the merge. Accept the default merge commit message.

11. **Delete** the `feature2` branch label.

12. Verify that you have a commit graph with a merge commit containing all three features.

13. You will not use the `projectd` repository in future labs. You can delete it.

    > Congratulations, you have resolved a merge conflict and completed this lab.