Lab: (Sourcetree) Resolving Merge Conflicts

Estimated time: 10 minutes

> Note: This lab assumes that you are using Sourcetree. If you would prefer to use a command line interface, 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. Use Sourcetree to 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. Click **Merge** and attempt to merge in the `feature2` branch. You should be warned that

there is a merge conflict. Click to dismiss this message.

3. View the uncommitted `fileA.txt` file that Git has placed in your working tree. Notice the conflict markers in the file. That is the part of the merge that Git couldn't automatically resolve.

4. **(Mac)** Rather than fix the conflict right now, abort the merge process by selecting **Reset...** under the **Repository** menu of Sourcetree. Click both **Reset all** buttons to abort the merge.

   (Windows) Rather than fix the conflict right now, abort the merge process by selecting **Discard**. Select the **Reset All** tab, then click **Reset All**.

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

6. This time, let's resolve the merge conflict. Click **Merge** again and attempt to merge in the `feature2` branch. Dismiss the merge conflict message.

7. **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".

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

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

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

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

12. 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.