Lab: (Sourcetree) Branches

Estimated time: 20 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 and checkout a branch.
2. Create commits on the branch.
3. Checkout an old commit.
4. Delete a branch.

1: Create and checkout a branch.

1. Using Sourcetree, create a local repository named `projectc`. If you need help, please refer to the previous labs.

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

3. **(Mac)** Create a branch off of the latest master commit named "featureX". In Sourcetree's History tab, make sure that the latest commit is highlighted and its details show that it is the current commit (`HEAD -> master`). Its commit message should say "add feature 1". **Click** the Branch icon to start creating the branch.

   (Windows) Create a branch off of the latest master commit named "featureX". Select Sourcetree's Log/History tab (at the bottom) and make sure that the latest commit has a hollow circle next to it. This means that this is the currently checked out commit. Its commit message should say "add feature 1". **Click** the Branch icon to start creating the branch.

4. In the window that appears, make sure that **New Branch** is selected at the top. **Name** the branch `featureX`. Under Commit, "Working copy parent" should be selected. This means that you will be branching off of the current commit. Make sure the "Checkout new branch" checkbox is **selected**. **Click Create Branch** to create the `featureX` branch.

5. Under the BRANCHES tab on the left, you should now see your `featureX` branch. There is a circle next to the branch name, indicating that it is checked out. This means that working tree contains the files from this version of the project, and the next commit that you make will be to this branch.

6. Notice that the latest commit now has both the `master` and the `featureX` branch labels.

   (Mac) In the commit details, you should see `HEAD -> featureX`, indicating that the next commit you make will be to the `featureX` branch. (Note, you might need to click on a different commit and click back to refresh the details.)

   > Congratulations, you have created and checked out a branch.

2: Create commits on the branch.

1. Now that you have created and checked out the `featureX` branch, you can do some work on the project without affecting the `master` branch. In your local repository, **create a commit** on the `featureX` branch with the following:

   - modify `fileA.txt`, adding "feature mistake" directly under the line "feature 1"
   - add a commit message of "add feature mistake"
     (As a reminder, you can use any tool that you want to modify the `fileA.txt` file. You can usually double-click on the file in Sourcetree to edit it. Once you have edited and saved `fileA.txt`, click on the "File status" tab. `fileA.txt` should show as unstaged. Stage the file. Click Commit (if necessary) and add the message "add feature mistake".)

2. Click on the History (or Log/History for Windows users) tab and view your commit graph. You should see a straight line, with your `featureX` branch label and "add feature mistake" commit message on the most recent commit. You should see that the `featureX` branch is checked out (under BRANCHES).

3. Under BRANCHES, double-click the `master` branch to check it out. Your working tree will be updated with the older version of `fileA.txt`. View the contents of that file and verify that you do not see your "feature mistake" content. The `master` branch is unaware of the work that you did on the `featureX` branch. Notice in the commit graph that the second commit is the current commit, as indicated by the hollow circle. If you changed the working tree and committed right now, the commit would be to the `master` branch.

4. **Change back** to the `featureX` branch by checking it out (double-click under BRANCHES).

5. **Create another commit** on the `featureX` branch with the following:

   - in `fileA.txt`, under "feature 1", change the line "feature mistake" to "feature bigger mistake"
   - add a commit message of "add feature bigger mistake"

6. Click on the History (or Log/History) tab and view your commit graph. You should again see a straight line, with two commits on the `featureX` branch.

> Congratulations, you have created commits on the `featureX` branch.

3: Checkout an old commit.

1. Let's say you want to view the first change that we made on the `featureX` branch. **Checkout** the first commit you made on the `featureX` branch ("add feature mistake"). Do this by double-clicking on the commit in the commit graph. You will be shown a confirmation dialog about entering a detached HEAD state. This means that there is no branch label associated with this commit, and that your HEAD reference will point directly to the commit. **Click OK** to checkout the commit.

2. Verify that you are seeing the older version of `fileA.txt` ("feature mistake") in your working tree. Also notice that the current commit has a HEAD tag with no branch label. You are in a detached HEAD state. We are only viewing the old commit, so we are OK. If we wanted to create new commits based on this commit, we should create a branch right now. We don't need to do that though.

3. **Checkout** the `master` branch to get out of the detached HEAD state.

> Congratulations, you have checked out an old commit.

4: Delete a branch.

> Well, our great featureX idea might not have been so great after all. We went from "big mistake" to "bigger mistake". We will delete the `featureX` branch without merging it into `master` (you will learn about merging later).

1. Under BRANCHES, right-click on the `featureX` branch and select **Delete > featureX**. A confirmation dialog appears. Try deleting the branch now without selecting Force delete- **click OK**. Sourcetree opens a dialog, showing the response from Git as if you were on a command line. You will see that Git won't let you delete this branch, because it has not been merged. Your two commits on the `featureX` branch would become "dangling commits" and would eventually be garbage-collected by Git. In the Git message, it says that if you are sure that you want to delete the branch, use the `-D` option with the `git branch` command. You don't have to do that in Sourcetree, as we will see. Click **Close** to dismiss the Git message.

2. **Delete** the `featureX` branch again, but this time select the **Force delete** checkbox. Behind the scenes, Sourcetree uses the `-D` option with the `git branch`, as suggested

by Git. The `featureX` branch is deleted.

3. View the commit graph and verify that you are back to having only a `master` branch.

4. **(Mac)** (If you are interested) Want to "undo" the deleting of the `featureX` branch? In Sourcetree, select View > Command History. Expand the entry for Deleting branch…. You should see something like "Deleted branch featureX (was 345263e)". That SHA-1 is the commit ID of the commit that the `featureX` branch label pointed to. **Copy** that SHA-1. Click on **Branch**, select **New Branch**. Name the branch `featureX` and paste the SHA-1 into the textbox related to "Specified commit". You should now see your `featureX` branch again! Git doesn't really delete commits right away. Another way to obtain that commit's SHA-1 is to use the `git reflog` command at the command line. Please delete the `featureX` branch again.

   (Windows) (If you are interested) Want to "undo" the deleting of the `featureX` branch? Git doesn't really delete commits right away. You can obtain the `featureX` branch tip's SHA-1 using the `git reflog` command at the command line. You could then checkout that commit and create another `featureX` branch.

> Congratulations, you have deleted a branch and completed this lab.