# Lab: (Command Line) Branches

Estimated time: 20 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 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. 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. Use `git branch` to verify that you have a single branch in your local repository, and its name is `master`. Use `git log --oneline --graph` to verify that you are currently on the most recent commit. You should see `HEAD -> master` on the most recent commit.

4. Create and checkout a branch off of the latest master commit named "featureX". You can do this with two command or one command:

   ```
   # two command approach
   $ git branch featureX
   $ git checkout featureX

   # one command approach
   $ git checkout -b featureX
   ```

5. Execute `git branch` to verify that you have created a `featureX` branch, and that it is the currently checked out branch. Execute `git log --oneline --graph` to verify that the `featureX` branch is the current branch- you should see `HEAD -> featureX`. Notice that the latest commit now has both the `master` and the `featureX` branch labels. Because `featureX` is the current branch, the next commit you make will be to this branch.

> 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"
     (If you need a refresher on how to use `git add` and `git commit` to create a commit, see the previous labs.)

2. Execute `git log --oneline --graph` and view your commit graph (the asterisks). 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 (`HEAD -> featureX`).

3. Execute `git checkout master` to checkout the master branch. 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.

4. Execute `git log --oneline --graph`. Notice that only information about the current branch is listed. Also notice that the current branch is the `master` branch `HEAD -> master`. If you changed the working tree and committed right now, the commit would be to the `master` branch.

5. Execute `git log --oneline --graph --all`. Add `--all` shows all of the local branches. Now you can see your `featureX` branch, and the `featureX` branch has a commit more current than the commit at the tip of the `master` branch.

6. **Change back** to the `featureX` branch by checking it out.

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

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

8. Execute `git log --oneline --graph --all` 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 executing `git checkout HEAD~`. The appended ~ means "parent of the commit".
   Git will warn you that you are entering a detached HEAD state. This is because your HEAD
   reference points directly at the SHA-1 of a commit, instead of to a branch label. Read
   Git's message, it is informative.

2. Verify that you are seeing the older version of `fileA.txt` ("feature mistake") in your
   working tree. Execute `git log --oneline --graph --all` and 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. Try to delete the `featureX` branch using `git branch -d featureX`. 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, notice that it says that if you are sure that
   you want to delete the branch, use the `-D` option with the `git branch` command.

2. **Delete** the `featureX` branch again, but this time use the `-D` option. The `featureX`
   branch is deleted.

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

4. (If you are interested) Want to "undo" the deleting of the `featureX` branch? Execute `git reflog`. This shows the local history of HEAD references. Since Git doesn't immediately
   delete commits, you can find the SHA-1 of your most recent `featureX` branch there.
   **Copy** the SHA-1 of the "add feature bigger mistake" commit. Execute `git checkout -b featureX [SHA-1 YOU COPIED]`. View your commit graph and verify that your `featureX`

branch has returned. **Delete** the `featureX` branch again.

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