Hands-on Lab: Get familiar with fork and pull requests

Estimated time needed: 30 mins

Objectives

After completing this lab, you will be able to:

1. Use git commands to manage upstream repositories
2. Create a personal access token
3. Fork existing repository using the UI
4. Clone forked repository in the lab environment
5. Create a new branch
6. Make changes locally
7. Add and commit to the local branch
8. Push changes to the forked repository
9. Create a pull request to the upstream repository

Pre-requisites

This lab is designed to run on Skills Network - Cloud IDE which runs on a Linux system in the cloud and already has git installed.
If you intend to run this lab on your own system, please ensure you have git (on Linux or macOS) or GitBash (on Windows) installed.

> Note: While the lab allows you to copy-paste the commands, the best way to learn is to type the command yourselves. The instructors highly recommend the same.

Exercise 1: Generate personal access token

The first step is to generate an access token from GitHub.com. Follow the lab named **Generate GitHub personal access token** and copy the access token to use as a password in the upcoming exercises.

Exercise 2: Fork the repository

To fork a source repository, complete the following steps:

1. Log in to GitHub and go to this project's sample source repository. This is the upstream repository for your project.
2. At the top right of the screen, click `Fork` and select your own GitHub account as the destination for the fork.

A copy of the source repository has now been added as one of your GitHub repositories. This is the origin repository.

Exercise 3: Clone the forked repository

A clone is a local copy of a repository. Before you can clone the forked repository, you first need its HTTPS URL, which provides secure access to it.

To clone the forked repository, complete the following steps:

1. In your list of repositories, click the forked repository. On the repository's main page, click the **Code** button.
2. Click the clipboard icon to copy the URL. Make sure the `HTTPS` tab is active.
3. Open the terminal in the lab environment by using the menu in the editor: Terminal > New Terminal.
4. Let's export the copied URL in an environment variable so it's available for us to use in the later steps, run the following command in terminal:

   ```
   export ORIGIN=<your repository HTTPS URL>
   ```

   Replace <your repository HTTPS URL> with the URL you copied in step 2.
5. Run the following command with the `HTTPS` URL you copied earlier:

   ```
   git clone $ORIGIN
   ```

The command clones the repository that is on GitHub into your current directory.

Exercise 4: Explore the cloned repo

To become familiar with the cloned repo, complete the following steps:

1. Click the Explorer icon as shown in the following image:

2. Click Project and expand the folder of the project you just cloned. You can open the files in the editor, on the right side, by clicking on the file name.

Exercise 5: Create the `feature-circle-500` branch

We will now add a new feature to the source code.We will increase the circle's size to 500x500 pixels.Before we make this change, we will create a new branch.

1. Navigate to our repository using this command `cd gkpbt-css-circle`
2. Create a new branch using the `git checkout -b feature-circle-500` command. Notice that we used a single command instead of creating a branch and then checking it out. The `-b` flag creates the branch if it does not already exist.
3. You can check that you are in the new branch by using the `git branch` command.

Exercise 6: Make required code changes

1. Let's change the width and height to 500px each. Open the `style.css` file from the file explorer and change the code as follows:

```
        .blue {
                background-color:blue
        }
        .circle{
                border-radius:50%;
                width:500px;
                height:500px;
        }
```

2. If you do a `git status` at this point, you will see a change is shown. This change is not staged at this point, but Git is aware of it.

   ```
   git status
   ```

3. Optionally, you can use the `git diff` command to see the detailed changes:

   ```
   git diff ./style.css
   ```

Notice the text in red was deleted and the text in green was added. Essentially, we changed the height and width from 300px to 500px each.

> Note: To exit the git diff command, simply press the "Q" key.

Exercise 7: Add and commit your changes

A commit is Git's way of recording your file changes, similar to how you might save an edited document. To commit the change that you made in the previous exercise, you first need to add it to a staging area. Git will then take the staged snapshot of changes and commit them to the project. Remember, Git will never change files unless you explicitly ask it to.

To commit your new file, complete the following steps:

1. To move the changes from your working project directory to the staging area, type the following command in the Terminal window:

   ```
   git add .
   ```

   The `git add` command has several options. The single `.` adds all untracked files in the current directory and subdirectories to the staging area. Alternatively, you can add the single file you created by using the `git add style.css` command. Finally, you can use `git add -A` to recursively add all files from the top level git folder.

2. If you check the status at this point, you will see the file has changed from Untracked to `Changes to be committed`:

3. To commit the new file to the local repository, you need to first tell git who you are. Type in the following commands to set your email and username. The email should be the same as your GitHub email.

Set your email:

```
git config --global user.email "email@example.com"
```

Set your name:

```
git config --global user.name "Your Name"
```

4. Type the following command in the Terminal window to commit the file.
   Note: It's always a good practice to add a description for the commit so you can remember what the change was if you have to refer to it later.

- **`-m` flag**: It is used in Git commit commands to specify the commit message directly in the command line, allowing you to provide a brief description of the changes you are committing.

```
git commit -m "Changing the height and the width of the circle"
```

As you can see, `git status` now says there is nothing to commit and the working tree is clean. The new file is now ready to be pushed from your local system to origin on GitHub.

Exercise 8: Merge your branch back into main branch

If you are happy with your changes in the `feature-circle-500` branch, you can now merge it back into your local `main` branch by following these steps:

1. Confirm that you are currently in the `feature-circle-500` branch.

2. Check out the `main` branch

   ```
   git checkout main
   ```

   If you run `git branch` again, you should see the `*` against the `main` branch.

3. Merge the `feature-circle-500` branch into`main`.

   ```
   git merge feature-circle-500
   ```

4. Confirm the change was merged by using the `git log` command. We are using `--oneline` flag to display logs more concisely.

Note: To exit the `git log` command, simply press the "Q" key. This action will close the log view and bring you back to the command prompt.

Exercise 9: Delete the `feature-circle-500` branch

Since you are done making the change, let's delete the `feature-circle-500` branch by following these steps:

1. Ensure you are on the `main` branch. If not, check it out first

   ```
    git checkout main
   ```

2. Delete the `feature-circle-500` branch, the common flag used is -d (lowercase), which stands for "delete"

   ```
   git branch -d feature-circle-500
   ```

3. You can confirm the branch was deleted by listing all branches

   ```
   git branch
   ```

Exercise 10: Push your changes to origin

This push will synchronize all the changes you made on your local system with your fork repository on GitHub.

To push your update to GitHub, complete the following steps:

1. In the Terminal window, run the following command:

   ```
   git push origin main
   ```

   Once you submit that command, Cloud IDE will display a dialog in the lower right corner, requesting permission to sign in using GitHub. Click "Allow"

   Note : If you don't see the dialog box below, you will be asked to enter your GitHub username and password in the terminal. Your PAT (Personal Access Token) will be hidden when you type or paste it in the terminal for security reasons. So, make sure you enter or paste it correctly before hitting 'Enter'.

▶ Note on Warning Messsages

2. Go to the fork repository in your GitHub account and verify that the local changes have now been added to the main branch.

Exercise 11: Create a pull request

The final step is to request the original project pull in the changes you've made to your fork. To merge your changes to the original repository, you need to create a pull request.

To create a pull request, complete the following steps:

1. Ensure you are on the **Code** tab. Click **Contribute** and then **Open pull request**.

2. In the "Comparing changes" panel, GitHub shows you that it is comparing the main branch of your fork to that of the original repository, and that your changes can be merged. Click the **Create pull request** button.

3. You are taken to the **Open pull request** screen. Notice that your commit message appears as the title of the pull request. Click the **Create pull request** button.

Note: For the purposes of this lab, your pull request will be processed and closed automatically.

You should see the following message in your pull request after a few minutes:

Exercise 12: Practice on your own

1. Create a new branch called `feature-add-color`.

   ▶ Click here for the solution

2. Make `feature-add-color` the active branch.

   ▶ Click here for the solution

3. Add another css rule as follows:

```
.red {
     background-color:red
 }
```

4. Stage this change.

   ▶ Click here for the solution

5. Commit the changes in your `feature-add-color`.

   ▶ Click here for the solution

6. Merge the changes in `feature-add-color` into `main`.

   ▶ Click here for the solution

7. Delete the `feature-add-color` branch.

   ▶ Click here for the solution

8. Push your changes to origin.

   ▶ Click here for the solution

9. Create a new pull request for this feature in the upstream repository using the GitHub UI.

Summary

In this lab, you have learned how to fork an upstream repository into your own account and then clone it locally in the lab environment. You then learned how to synchronize changes in your local repository with remote GitHub repositories using pull requests.

Author(s)

Upkar Lidder

Other Contributor(s)

Richard Ye