Getting Code with GitHub

(https://github.com/learn-co-curriculum/git-github-getting-code-from-github) (https://github.com/learn-co-curriculum/git-github-getting-code-from-github/issues/new)

Learning Goals

- Explain the different ways that GitHub can be useful.
- Copy a repository to your local machine with `git clone` .
- List remotes with `git remote` .
- Duplicate other organizations' repositories into your own GitHub account with the "Fork" button.

Introduction

In the previous lesson, we learned how Git allows us to create logged histories of all the versions of the files we "track." Another great benefit of using Git is that it makes it easy for developers to "share" code by "pushing" a copy of their repo to the internet (specifically, to GitHub). From there, other developers can **clone** down the code onto their own machine and use it.

For those of you in the Software Engineering program, you've been taking advantage of this feature of Git for a while with the labs you've been working on in this course! The important thing to understand is that the forking and cloning workflow that you've been using with labs can be used with *any* public repo on GitHub. In this lesson, you'll learn how to accomplish those steps "by hand."

How GitHub can be Useful

There are lots of ways GitHub can be useful. A few examples are:

- For personal projects:
  - It allows you to keep a backup copy of your project in the cloud.
  - It makes your projects available to other people (potential employers, for example) so they can see your work.

- For working as part of a team: the GitHub repo serves as the central source of truth, and team members pull the code down to their local computers to work on a particular issue or feature. When they're done, they push their solution up to be incorporated into the central repo.
- To contribute to open-source projects: There are lots of open-source projects available on GitHub that anybody can contribute to (hence "open-source"). You can find one of these projects, locate an issue or need that you think you can address, write the code on your local machine, and submit a request to the repo's owners to incorporate your solution into the main code base. This is a great thing to have in your GitHub profile when you start looking for a job!
- To use other developers' code as a starting point that you can expand on or modify according to your needs.

In several of these instances, we would want to copy (or **clone**) a repo that already exists in GitHub onto our local machine so we can work on it. We'll learn how to do that in this lesson. Then, in the next lesson, we'll learn how to **push** our work back up to GitHub.

Copy a Repository to Your Local Machine with `git clone`

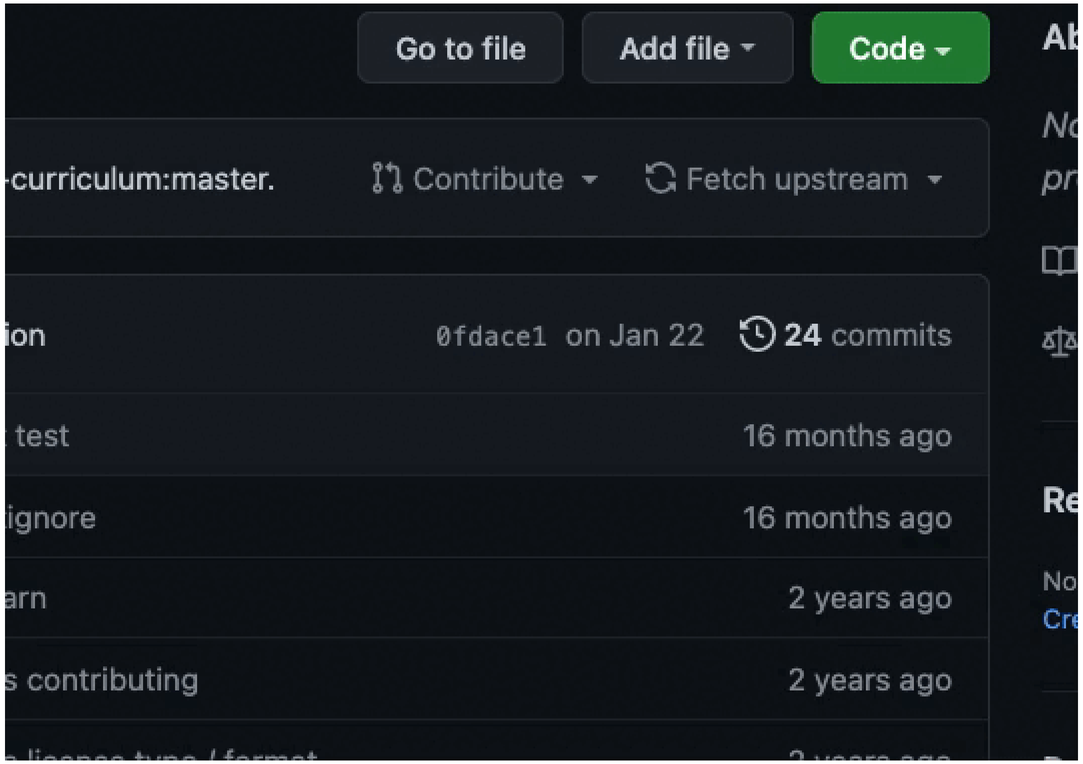
Let's go through the cloning process using the repo for the Python programming language:

1) Navigate to the **Python repository** (https://github.com/python/cpython) .

2) Click the green "Code" button on the right.

3) Select `SSH` if you have set up an SSH key and `HTTPS` otherwise. (See the first lesson in this module for more information about setting up an SSH key.)

4) Click the copy button.



5) In the terminal, navigate to where you want to put the repo. Type `git clone` and a space, then paste in the copied link from GitHub. The pasted URL should start with `git@github.com` if you're using SSH, and `https://github.com` if you're using HTTPS. Running this command will create a *local* copy of the GitHub repository on your machine.

Remember: Any time you are cloning an existing repo from Github, you should always make sure you're not inside a directory that is already initialized as a Git repo. You can do this by running `git status`. If you see the message below, it's safe to proceed.

```
$ git status
fatal: not a git repository (or any of the parent directories): .git
```

List Remote Repos with `git remote`

Recall that a `remote` repository is a repository on GitHub, and a `local` repo is one you have on your local machine. Once you've set up a local repository, if you want to either *fetch* code from the remote repository (e.g., if the code has been updated and you want to get the most recent version) or *push* code to the remote repository (e.g, to save your local changes to the cloud), Git needs to know which remote repo to connect to. To verify that our local repo is hooked up properly to the remote repo, we can use the `git remote` command.

If you type the `ls` command in your terminal, you'll see that, when you cloned down the Python repo, Git created a directory called `cpython` . Use `cd` to enter that directory.

```
$ cd cpython
```

The `git remote` command will return the names of each remote repository (or, "remote") that is associated with a local repo. If, as in this case, you've cloned down a repo from GitHub, the remote will be that repo.

Go ahead and run the command; you should see a remote named `origin` returned. This is the "nickname" that Git assigns by default to whatever remote you cloned from:

```
$ git remote
origin
```

We can use the `origin` nickname rather than typing out the entire URL when we want to fetch or push code. To prove that the `origin` name points to the repo we cloned from GitHub, we can run `git remote -v` (the `v` flag stands for "verbose"). You should see something like this, with the appropriate change if you used HTTPS instead of SSH when you cloned the repo:

```
$  git remote -v
origin  git@github.com:python/cpython.git (fetch)
origin  git@github.com:python/cpython.git (push)
```

Here you can see that the "remote address" (`git@github.com:python/cpython.git`) assigned to the "remote name" (`origin`) is the same thing you copied from the GitHub web interface. This confirms that the *remote repository* you *cloned* automatically set up a *remote name* called `origin` .

Note: As you may have noticed, the Python repo is quite large. You might want to delete it before you move on by running:

```
$ cd ..
$ rm -rf cpython
```

In the next lesson, when we learn how to **push** code up to GitHub, the `git remote` command is what we'll use to ensure that the code is being pushed to the right place.

Copy Other Organizations' Repositories into Your GitHub Account with the "Fork" Button

While we can clone down any public repo on GitHub and make changes on our local machine, we can't push our changes back up to GitHub unless we have access rights to the remote repo. If you want to make changes to a public repo you cloned down from GitHub **and** push those changes back up to GitHub, you'll need to **fork** a copy of the repo to your own GitHub account first.

Let's go back to the Python repository (https://github.com/python/cpython) on GitHub. In the upper right corner of the page, you'll see three buttons, including a Fork button:

If you were to click the Fork button on the page for this repo (or any other public repo on GitHub), a copy of the repo would be made and stored in *your* GitHub account. Having your own copy gives you the ability to update its `main` (or `master`) branch locally and save those changes to your fork on GitHub without affecting the original repo.

This "fork and clone" workflow is what you would use to submit a contribution to an open source project. Once you've pushed your code from your local machine up to your fork of the repo on GitHub, you can submit a *pull request* to the repo you forked from. Someone in that organization will (hopefully) review your pull request and decide whether to *merge* it in to the official repo. You will learn more about pull requests and merging a bit later in this module.

Check for Understanding

Before moving on to the next lesson, check for your understanding of this material by answering the following questions in your own words:

- What are three things using GitHub enables you to do?
- Under what circumstances might you fork a repo before cloning it to your local machine?

Conclusion

GitHub gives developers many ways to collaborate. Using GitHub's "Fork" button and `git clone` together allows you to make copies of others' code and work on it on your local machine.

Often, the original authors will include license information regarding how you can use their repository, so make sure to check before you publish, sell or distribute any material you've forked, cloned and modified.