Pushing Code to GitHub

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

Learning Goals

- Create a local repository.
- Create a remote repository.
- Connect the local repository and the remote repository with `git remote`.
- Push code up to the remote repo with `git push`.

Introduction

You've seen how valuable *remotes* are for *getting* software. Now we can take a look at the other side of the transaction: how we can create a remote repository for our own projects and synchronize our *local* repository to it using `git remote` and `git push`.

Once your code is on a *remote* repo, it's backed up — which is always a good thing. Also, once you push to a remote, you can choose whether to let others `fork` or `clone` and benefit from it. Let's learn how to `push` our code!

In this lesson, we'll learn how we can set up remote repositories on GitHub for any projects we work on. Specifically, we'll learn how to get a new repo set up on our local drive, create a remote repo for it on GitHub, connect the local and remote repositories, and push our code up to the remote repo on GitHub.

Create a Local Repo

Let's review the process for creating a local repository:

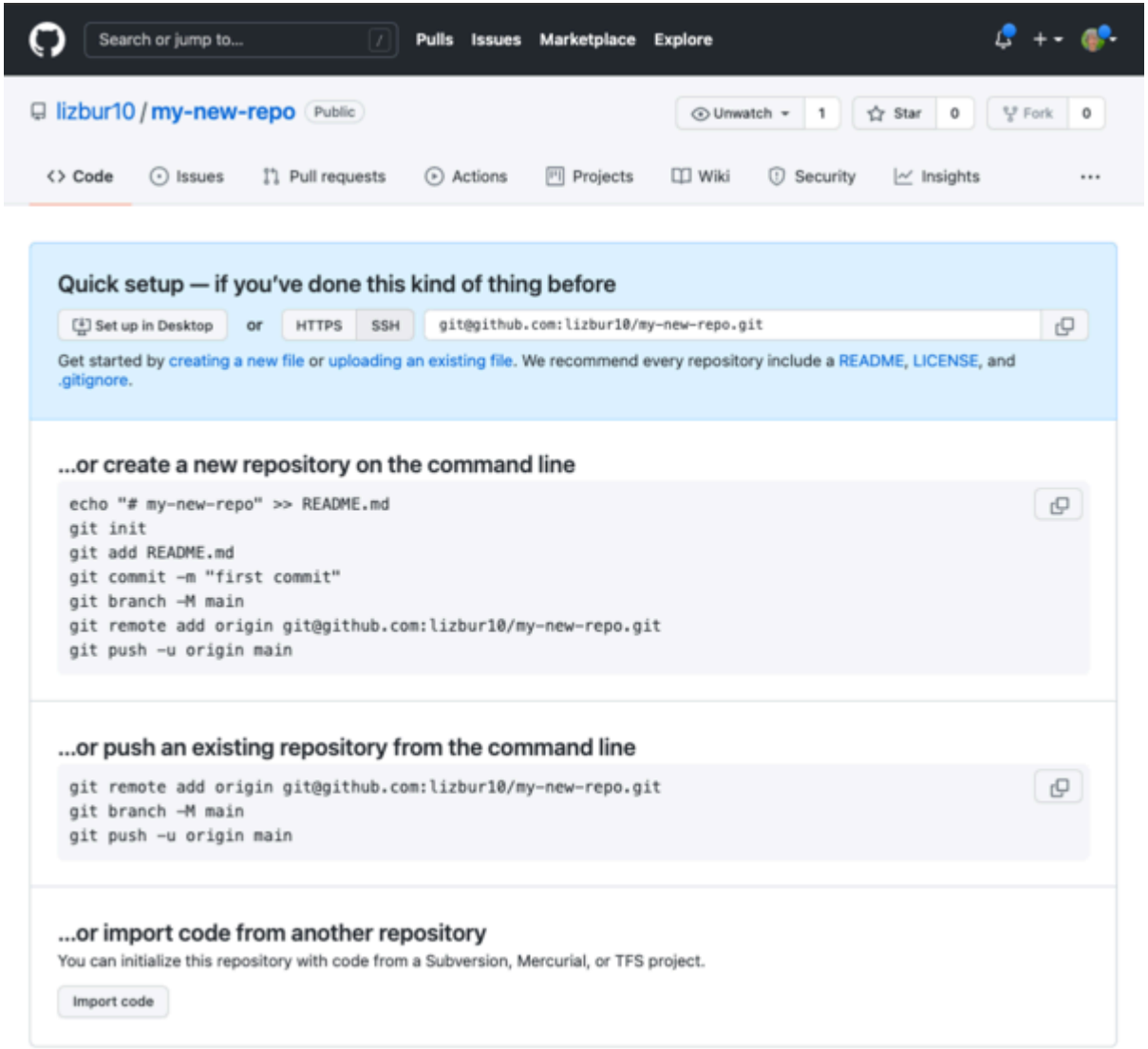
1. First, we want to create a folder for our repository, which we'll call `my_new_project`. In the terminal, navigate to whatever directory you want to store your files in (remembering to make sure that directory isn't a Git repo) and type `mkdir my_new_project`.

2. To navigate into this new folder, type `cd my_new_project`. Your terminal should display `my_new_project`, indicating that you are now inside of the new folder. Open the directory in VS Code by typing `code .`. (Your command will be different if you're using a different text editor.)
3. Next, we need to create a new file named `README.md`. You can do this in the terminal, by typing `touch README.md`, or in VS Code, by choosing `File -> New File`.
4. We can directly type in content for our README file in VS Code, but we can also use our terminal skills to add content. So, in the terminal, type `echo "This is my readme file" >> README.md`. If you've got the README file open in VS Code, the new text will appear!
5. Now that we've got some basic content, let's initialize our local repository. In your terminal, type `git init`. Your terminal should show that an 'empty Git repository' has been initialized.
6. Type `git add README.md` (or `git add .`) to stage the new `README.md` file so it will be tracked by Git.
7. Now, type `git commit -m "Initialize git"`. This will create the first commit for this local repository.

Create a Remote Repository on GitHub

There are a few steps to follow to create a *remote* repository on GitHub.

1. Go to: **github.com/new (https://github.com/new)**, while logged in to GitHub.
2. Enter a name for your repository in the "Repository name" field. You can name it whatever you'd like, but to avoid confusion you may want to use the same name you used locally, `my_new_project`. You can leave the remaining options as they are — the default options are fine. Click the green "Create repository" button.
3. After you create the *remote* repository, you should see a "Quick setup" box at the top of the page. Make sure the correct option is selected (SSH or HTTPS), then click the copy button next to the repo URL to copy the URL. (We'll use this in the next section.)



Behind the scenes, GitHub has essentially `git init` 'd a blank directory.

Connect the Local Repository to the Remote Repository

We learned in the previous lesson that the `git remote` command will list the **remotes** available to our Git repo. If you run that command now, ...nothing happens. This is because we haven't set our remote repo as the **remote** for our local repo yet. Doing so is a simple matter of assigning

the URL we copied above to a **remote name**.

We also learned that Git uses the remote name `origin` by default when we clone an existing repo from GitHub. We verified that by running `git remote -v`, which showed that the remote name `origin` was pointing to the repo on GitHub that we cloned. We'll stick to that convention here.

You should still have your remote Git info copied from GitHub. To set the remote, type `git remote add origin` followed by a space, then paste in the URL. It will look something like this:

```
$ git remote add origin git@github.com:your-github-username/my_new_project.git
```

The command above creates a remote called `origin` and points it to your new remote repo on GitHub.

Let's use `git remote -v` (recall that the `-v` is for "verbose") to verify that we successfully set our remote:

```
$ git remote -v
View existing remotes
origin  git@github.com:your-github-username/my_new_project.git (fetch)
origin  git@github.com:your-github-username/my_new_project.git (push)
```

We did it! We're now set up to ***push*** our code.

Send Code to the Remote Repo with `git push`

Now that we have added a remote repo, we need to send our latest work to it using `git push`. This command will push all the local, committed work to the remote repository.

The `git push` command takes two arguments. The first is the name of the remote repo. Remember, `origin` is just an alias or "short name" that refers to the repository URL. You don't actually have to enter the repository URL. Instead, you can just use `origin`. The second argument is the name of the remote **branch** you want to send code to. We're going to push to our remote repository's **default branch**. The command to push the code (assuming `main` is the default branch) is:

```
$ git push -u origin main
```

The first time you push code up to a newly-added remote repository, using the `-u` (short for `--set-upstream`) flag will tell Git to "save" the remote repository as the default push destination for your current branch. What this means is that, for every subsequent push from the `main` branch, you will only need to run `git push`.

If you go back to GitHub and refresh the page, you should now see the current version of your project directory (containing just the `README`) saved in the cloud!

Exercise

Let's go back to our todo's project that we created earlier and get it backed up on GitHub. Start by navigating back into that directory, then complete the steps below. If you need a reminder of how to do any of the steps, it is totally fine to go back and review the lesson or use Google!

1. Create a remote repo for the project on GitHub.
2. Connect the local repo to the remote repo.
3. Push the code up from the local repo to the remote.
4. Refresh the page on GitHub to verify that the code was pushed up.
5. Open the `todo-list.txt` file in VS Code and add one or two more todo's.
6. Add and commit your changes.
7. Push the updated code to the remote (remember, you can use a simpler command this time!).
8. Check to make sure the latest changes are saved on GitHub.

Conclusion

Being able to add Git remotes allows you to back up your local repository to a remote server. To review, the process is:

- Create a local repo and run `git init` to start tracking it.
- Create a remote repo on GitHub.
- Run `git remote add origin your-remote-repository-URL` to tie your local repo to the remote repo on GitHub.
- Use `git add` and `git commit` to save your changes.
- Use `git push -u origin main` to push the changes up to the remote repo the first time, and `git push` subsequently.

For existing repos that you've cloned down to your local machine, the process is even easier: once you've added and committed your changes, you simply need to run `git push`.

Resources

- **GitHub guide on pushing (https://help.github.com/articles/pushing-to-a-remote/)**