**Pushing to Github**

**Pushing to GitHub**

What a journey! Congratulations on reaching this far!

One of the most important parts of development is sharing the stuff you work on for other people to see and contribute to. If you don't want to do that you still need a system for version control, a place where you save different stages of your project that can be accessed as simple as pressing 3 clicks. As you've guessed by now the thing we'll introduce now is GitHub.

Before doing any other actions, please verify that your .gitignore contains at least the .env file, to avoid pushing our keys on the internet, and other things that you consider irrelevant for other people, for example, the information about your deployments.

Here is a .gitignore example:

# Compiler files

cache/

out/

# Ignores development broadcast logs

!/broadcast

/broadcast/\*/31337/

/broadcast/\*\*/dry-run/

/broadcast/

# Docs

docs/

# Dotenv file

.env

/lib

.keystore

Following this point, we will assume that you have your own GitHub account. If not, please read [this page](https://docs.github.com/en/get-started/start-your-journey/creating-an-account-on-github) to find out how to get one. Having a GitHub account opens up a ton of possibilities in terms of developing your project or contributing to existing open-source projects. You have the option of forking an existing project and building on top of it, enhancing or adding extra functionality. You can open up issues on existing projects and make contributions to the codebases of other people, there are a lot of stories about people who contributed to other people's code and formed a strong bond and found business partners and friends that way. Moreover, GitHub profiles are crucial when applying for jobs. Safe to say, that having a great GitHub profile that's interesting and stands out can open up a world of opportunities.

If you want to get started or want a quick start, [GitHub](https://docs.github.com/en) docs provide numerous sets of documentation that you can refer to. In this lesson, we will learn how to do all these using our terminal and not the GitHub website interface.

Before proceeding further, try running git version in your terminal.

If you receive an output similar to this git version 2.34.1 then you have correctly installed Git on your device. If not please follow the directions on how to install Git. You can find them [here](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git).

Ok, let's add our project to GitHub. Use the [following page](https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github) as a support material.

The first thing we need to do is to make sure we are in the **root directory of our project**. Then usually one calls git init -b main to init the Git repository. Foundry initiates a Git repository by default.

1. Try calling git status. You should receive an output that's similar to this:

On branch main

Your branch is up to date with 'origin/main'.

Changes to be committed:

Git status shows your current status, i.e. what did you modify, what is staged and not staged. It also shows untracked files.

**IMPORTANT: Do you see a .env file here?**

You shouldn't! The files you see in git status are going to be posted for everyone to see.

1. Try running git init -b main. You should receive an output that's similar to this:

warning: re-init: ignored --initial-branch=main

Reinitialized existing Git repository in

The next step is adding our files. The git add command is a fundamental tool in the Git version control system. It's used to stage changes you've made to files in your working directory for inclusion in the next commit.

In your terminal call git add .. Run git status again and compare its output to the output you received before. You'll see that the section Changes to be committed: is way bigger and green. All these files are staged and are waiting to be committed.

We mentioned earlier that Git is used for version control. But what is that?

Version control is a system that tracks changes to a collection of files over time. It allows you to revert to previous versions of files, see who made changes and when, and collaborate with others on projects.

Run git log in your terminal. This provides a list of commits. You can revert your code to these versions.

Let's commit our code. Run git commit -m 'our first commit!'

Ok, let's run git status again.

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

(commit or discard the untracked or modified content in submodules)

Let's run git log again.

You'll get an output similar to:

commit c3cd23888f84531a9a7a7a0c4e2070039a7a0b63 (HEAD -> main)

Author: InAllHonesty <inallhonesty92@gmail.com>

Date: Wed May 15 12:49:50 2024 +0300

our first commit!

Great!

All this is stored locally. As the git status indicated we will use git push to add this code to our profile.

There are multiple ways to do this, we will use git to push this code to GitHub. We will follow the indications available on this [page](https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github#adding-a-local-repository-to-github-using-git).

Let's go to GitHub.com and create a new repository.

1. Go on GitHub.com;
2. Click on + then on New Repository;
3. Give it a cool name that is available;
4. Add a description if you want and make the repository public;
5. Don't add a README file or a .gitignore template;
6. Click on Create repository;

Great! Now go to the Quick setup — if you’ve done this kind of thing before and copy the HTTPS link (mine looks like this: <https://github.com/inallhonesty/fundMe-lesson.git>)

Run the following command, replacing my link with your link:

git remote add origin https://github.com/inallhonesty/fundMe-lesson.git

Cool! Now run git remote -v.

Here we can see all the sources we can pull and push code from/to.

Next call the following command:

git push -u origin main

This tells git to push all of our code to the URL associated with the origin URL, on the main branch.

Some things can go wrong at this point, all of them are related to access and configuration. We encourage you to paste the error in ChatGPT or any other similar tool you chose. Ai is good at troubleshooting GitHub!

Hopefully, everything went smoothly! If it didn't, and you are unable to find a solution, please come and ask on Cyfrin Discord, in the Updraft dedicated section and channels.

Go back to the repository we created and refresh.

Great, now your code is on GitHub. The first thing we should do is create a better README.md file. Remember the lesson we had about this!

Go inside the README.md file in your VSCode. Create some titles, provide some info about the project, and specify some requirements and a way to quick-start the project. Save and close the file.

Let's repeat the commands we used above:

git add .

git commit -m 'Update the README.md file'

git push origin main

Go back on GitHub and refresh to see your new README! Super nice!

Another important git command is git clone.

The git clone command in Git is used to create a copy of a remote repository on your local machine.

Let's say you find a project that you like on GitHub: <https://github.com/Cyfrin/2023-10-PasswordStore>

You want to build on top of this project. How do you get it on your local system?

When you open the link above, you will find a green button called <> Code v. Press it, then press on Local then press HTTPS. Now if you copy that link and call the following command in the terminal:

git clone https://github.com/Cyfrin/2023-10-PasswordStore.git

you will obtain a fresh copy of the PasswordStore code inside a new folder with the name of the repository. In our case 2023-10-PasswordStore.

**Note: Keep in mind that the new folder will be created in your current working directory. You might not want to create it inside the FundMe project directory**

You can now run the following commands to open up a new VSCode instance starting from the newly created folder:

cd 2023-10-PasswordStore

code .

Amazing! You can now show off your newly created GitHub project! Click [here](https://twitter.com/intent/tweet?text=I%20just%20made%20my%20first%20Smart%20Contract%20repo%20using%20@solidity_lang,%20foundry,%20@chainlink,%20@AlchemyPlatform,%20and%20more!%0a%0aThanks%20@PatrickAlphaC!!) to tweet about this celebratory moment! Make sure to link your repository!