App Explanation

How to set up your project on a new machine with just one step

We'll cover the following

In the last section, we covered data handling in and out of Docker. We also covered the basic commands used in Docker. It's time to get started with Docker instructions that will help us automate our manual project setup on any desktop.

> Docker is not limited to the functionality of setting up a project in one go. There are a lot of other functionalities provided by Docker. However, Dockerfile is the stepping stone for all of it.

Dockerfile for Flask app

To start with Dockerfile we need a project. We will use the Flask framework which is a very minimalistic framework that can be understood by any newbie.

It will be a very basic app since we are looking at the fundamentals of Docker and not programming. Also, it will let us focus more on Docker and less on programming. Otherwise, we will spend more time on understanding the code and less time on our main topic. So, our app will have a normal login functionality as it is implemented in every web app.

The app will have a login/sign in form and in the backend, we will store user credentials in a dictionary for now. A user will log in using credentials. If login is successful, he will be prompted with a greeting or else, he will be asked to log in again.

So, let's code.

> *From now, you will be working with code so,* `git clone https://github.com/venky8283/Docker.git` *project and type* `git log` *to checkout to the specific commit relevant to the lesson.* **For this lesson,** `git checkout 114b58be8ce05029d397b5206e3a1ece9720fdfd` **.**

```
flask
python-dotenv
```

App implementation

In the file explorer, you will see three files and one folder.

.flaskenv: This is a hidden Linux file used for environment variables declaration specifically for Flask.

> Windows users use 'set FLASK_APP=app.py' and 'set FLASK_ENV=development' command to set an environment variable in the same window from where you will run the Flask code. Use this method if you haven't installed the python-dotenv package.

requirements.txt: This file will have all the requirements we will require to run our app. Later, we will install all with one command.

templates/form.html: You might have understood from the name itself that this will have our form HTML code for login. Flask looks for HTML templates in a folder called templates, hence it is good practice to have templates in the templates folder.

At this point, if you have hit the run button for the app, you might have seen something going on in the console. If you know what is happening and how the code is running, you can skip the next chapter.

> If you try to access the demo site using the link provided in the output window, you won't be able to access it now. We will see how to do that as we move forward in this course.

Let’s grab a cup of coffee and try to understand how we are running this code. If you don’t understand, don’t worry, we will cover that in the next chapter.