**Step 1 - Application structure**

Our application will look like this at the end:

|---- docker-compose.yml

|---- app

|---- app.py

|---- requirements.txt

Create a new folder for our application named docker & create Docker Compose .

mkdir docker && cd docker

touch docker-compose.yml

**Step 2 - Building our Flask app**

Create a new folder called flask and run the command below to open it.

cd flask

Within the flask directory, create a new folder and call it app. Inside the app directory, create an app.py file. The code for our Flask application will be in this file.

Here is the code for our simple Flask application.

app.py

from flask import Flask # importing the flask class

app = Flask(\_\_name\_\_) # creating an instance of the Flask class

@app.route('/') # The primary url for our application

def hello\_world(): # This method returns 'Flask Dockerized', which is displayed in our browser.

return 'Flask Dockerized'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, host='0.0.0.0') # This statement starts the server on your local machine.

Step 3 - Outlining key requirements

In this step, we need to create a requirements.txt file. It contains the dependencies or packages required to run our application.

To create this file, we’ll use the command below.

pip freeze > requirements.txt

**Step 4 - Creating a Dockerfile**

A Dockerfile is used to create an image for our application. This image will run on any host or environment with Docker installed. Our web application can, therefore, be deployed everywhere using an image.

touch Dockerfile

Include the following instructions and arguments in the generated Dockerfile.

Dockerfile

FROM python:3.8

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

ENTRYPOINT ["python"]

CMD ["app.py"]

The above Dockerfile contains the commands needed to assemble our image.

Here is the command that is executed when the container is launched:

docker run python app.py

**Step 5 - Docker Compose**

docker compose up

Open the docker-compose.yml file we created and then add the following services:

docker-compose.yml

version: "3.7"

services:

helloworld:

build:

context: ./

ports:

- 5000:5000

**Step 6 - Building and testing**

Run the command below in the same folder that contains **docker-compose.yml**:

docker-compose up

The above command displays the output of each container.

This can also be achieved using the following command:

docker-compose up --build

Step 6 - Docker Compose commands

docker-compose

docker-compose ps command will list any running containers.