Experiment 10 - Web Application using Docker

Roll No.	24
Name	Iyer Sreekesh Subramanian
Class	D15-A
Subject	DevOps Lab
LO Mapped	LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements LO5: To understand the concept of containerization and Analyze the Containerization of OS images and deployment of applications over Docker

Aim: To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.

Dockerfile:

What is a Dockerfile?

- A Dockerfile is a text file that contains a series of commands or instructions.
- These instructions are executed in the order in which they are written.
- Execution of these instructions takes place on a base image.
- On building the Dockerfile, the successive actions form a new image from the base parent image.

Syntax

# comments

command argument argument1...

Example

# Print "Hello World"

Run echo "Hello World"

List of Docker Commands for Creating a Dockerfile

Dockerfile consists of specific commands that guide you on how to build a specific Docker image.

The specific commands you can use in a dockerfile are:

FROM, PULL, RUN, and CMD

FROM - Creates a layer from the ubuntu:18.04
PULL - Adds files from your Docker repository
RUN - Builds your container

CMD - Specifies what command to run within the container
Mentioned below is an example of the dockerfile with the important commands

```
FROM ubuntu:18.04

PULL. /file

RUN make /file

CMD python /file/file.py
```

ENTRYPOINT allows specifying a command along with the parameters

Syntax

ENTRYPOINT application "arg, arg1".

Example

```
ENTRYPOINT echo "Hello, $name".
```

ADD command helps in copying data into a Docker image

Syntax

ADD /[source] /[destination]

Example

```
ADD /root_folder /test_folder
```

ENV provides default values for variables that can be accessed within the container

Syntax

ENV key value

Example

```
ENV value_1
```

MAINTAINER declares the author field of the images

Syntax

MAINTAINER [name]

Example

```
MAINTAINER author_name
```

How to build an image using a Dockerfile?

Use the following command -

```
docker build -t <dockerfile_name> .
```

You can use `docker images` to take a look at the created image.

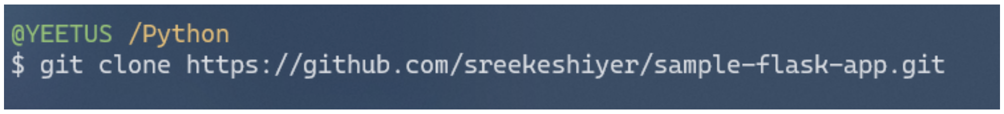
After that, you can use `docker container run -d <image_name>` to run a container using the image.

<u>Sample Web Application</u>:

In this experiment, we will build a docker image for a sample flask API which on connection returns a greeting. You can clone this application from this GitHub repository.

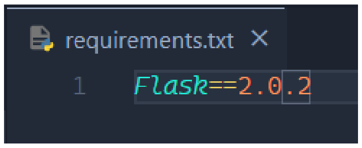
Steps:

1. Clone the GitHub Repository from this URL.



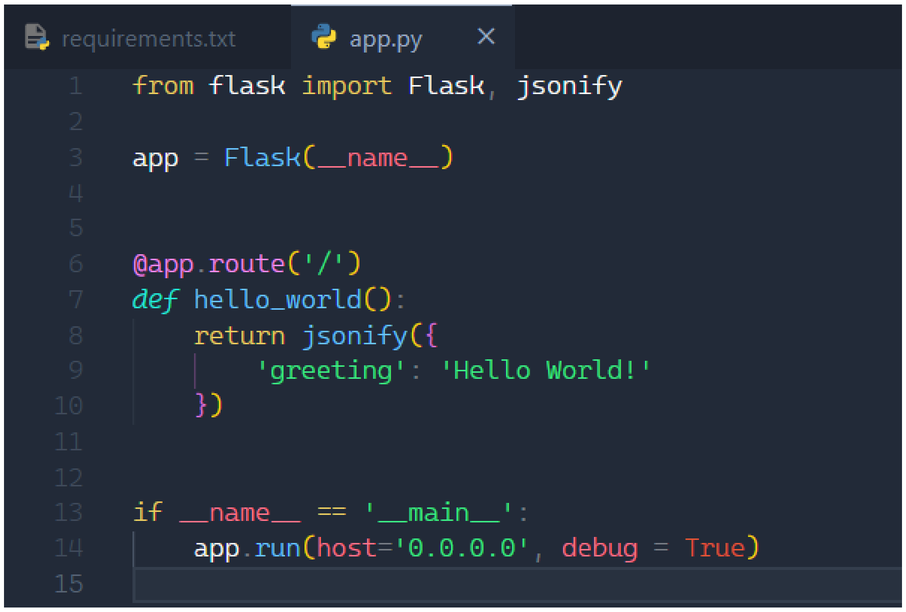
```
@YEETUS /Python
$ git clone https://github.com/sreekeshiyer/sample-flask-app.git
```

2. Check the *requirements.txt* file to confirm that you are installing the latest flask version.



```
Flask==2.0.2
```

3. You can view the code in app.py

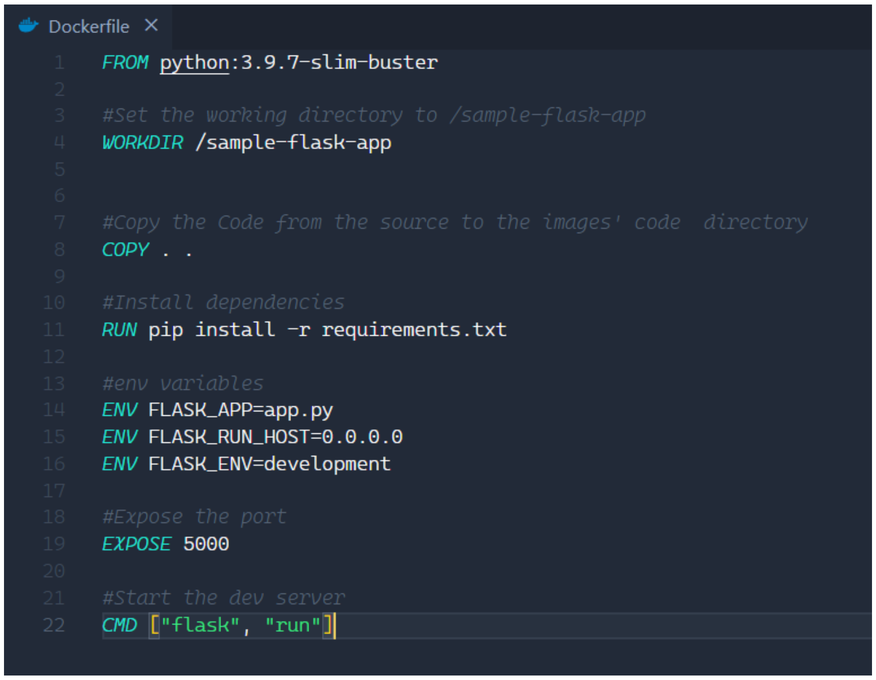


```python
from flask import Flask, jsonify

app = Flask(__name__)


@app.route('/')
def hello_world():
    return jsonify({
        'greeting': 'Hello World!'
    })


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug = True)

```

As you can see, we are creating a flask app that simply returns a greeting when you run it.

4. Create a new file in the same folder, named ‘Dockerfile’.
   Add the following contents to the file, like so.

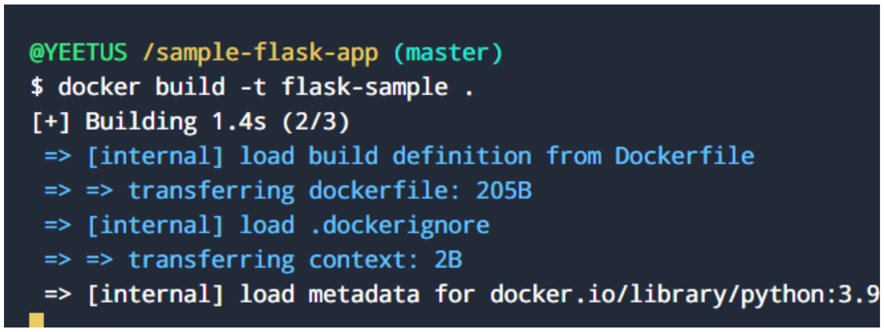
```
FROM python:3.9.7-slim-buster

#Set the working directory to /sample-flask-app
WORKDIR /sample-flask-app
#Copy the Code from the source to the images' code  directory
COPY . .

#Install dependencies
RUN pip install -r requirements.txt

#env variables
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
ENV FLASK_ENV=development

#Expose the port
EXPOSE 5000

#Start the dev server
CMD ["flask", "run"]
```



```
FROM python:3.9.7-slim-buster

#Set the working directory to /sample-flask-app
WORKDIR /sample-flask-app


#Copy the Code from the source to the images' code  directory
COPY . .

#Install dependencies
RUN pip install -r requirements.txt

#env variables
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
ENV FLASK_ENV=development

#Expose the port
EXPOSE 5000

#Start the dev server
CMD ["flask", "run"]
```

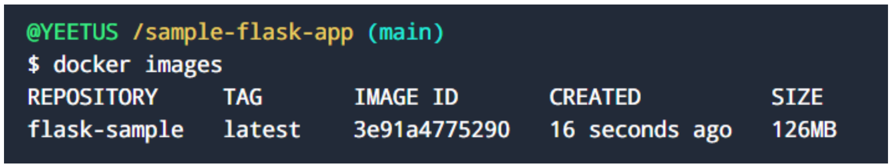
This Dockerfile will be used to create a Docker image for our sample app.

5. Open the terminal and run `docker build -t <dockerfile_name> .`



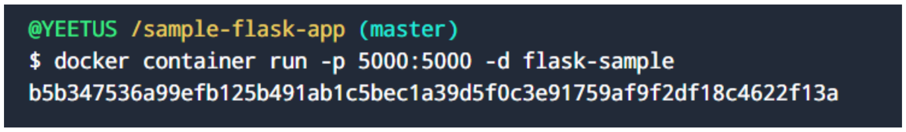
```
@YEETUS /sample-flask-app (master)
$ docker build -t flask-sample .
[+] Building 1.4s (2/3)
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 205B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.9
```

6. Once the image is successfully created, you can use `docker images` to check it.



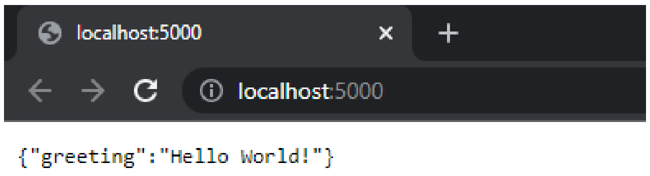
```
@YEETUS /sample-flask-app (main)
$ docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
flask-sample   latest    3e91a4775290   16 seconds ago   126MB
```

7. After that, we can use this image to run a container using -
   `docker container run -p 5000:5000 -d flask-sample`



```
@YEETUS /sample-flask-app (master)
$ docker container run -p 5000:5000 -d flask-sample
b5b347536a99efb125b491ab1c5bec1a39d5f0c3e91759af9f2df18c4622f13a
```

8. Check localhost:5000 in your browser and you can see your app running.



```
{"greeting":"Hello World!"}
```

Conclusion

Thus, we learned about Dockerfile, created and wrote Dockerfile for a sample Flask WebApp and built a Docker Image using it.