

Experiment - 10

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

Theory:

What Is a Dockerfile?

When you run the Docker run command and specify WordPress, Docker uses this file to build the image. The Dockerfile is essentially the build instructions to create the image.

The advantage of a Dockerfile over just storing the binary image (or a snapshot/template in other virtualization systems) is that the automatic builds will ensure you have the latest version available. This is a good thing from a security perspective, as you want to ensure you're not installing any vulnerable software.

Building a docker image:

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

Link: [GitHub repository](https://github.com/sreekeshiyer/sample-flask-app)

Implementation:

Step 1: Clone the GitHub Repository.

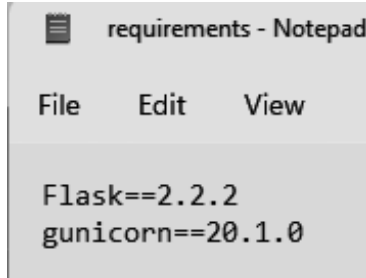
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\mayuri> git clone https://github.com/sreekeshiyer/sample-flask-app.git

Cloning into 'sample-flask-app'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 20 (delta 6), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (20/20), done.
Resolving deltas: 100% (6/6), done.
PS D:\37>
```

Step 2: Check the requirements.txt file to confirm installing the latest flask version



```
requirements - Notepad
File Edit View
Flask==2.2.2
gunicorn==20.1.0
```

Step 3: You can view the code in app.py



```
app.py 1 ●
app.py > ...
1 from flask import Flask, jsonify
2
3 app = Flask(__name__)
4
5
6 @app.route('/')
7 def hello_world():
8     return jsonify({
9         "message": "Hello World."
10    })
11
12
13 if __name__ == '__main__':
14     app.run()
```

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

Step 4: Create a new file in the same folder, named 'Dockerfile'. Add the following contents to the file, like so.

```

Dockerfile - Notepad
File Edit View

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
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.

Step 5: Open the terminal and run `docker build -t`

```

PS C:\Users\mayuri> docker build -t sampleflask .

[+] Building 14.3s (9/9) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 447B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/python:3.9.7-slim-buster       4.4s
=> [1/4] FROM docker.io/library/python:3.9.7-slim-buster@sha256:76eaa9e5bd357d6983a88ddc9c4545ef4ad64c50f84f081b  6.6s
=> => resolve docker.io/library/python:3.9.7-slim-buster@sha256:76eaa9e5bd357d6983a88ddc9c4545ef4ad64c50f84f081b  0.0s
=> => sha256:b380bbd43752f83945df8b5d1074fef8dd044820e7d3aef33b655a2483e030c7  27.14MB / 27.14MB  4.0s
=> => sha256:14063a2781d6ddd50c6afb4a9e22cccf2205cb04a72caff066c71cf3e4d19c76  2.77MB / 2.77MB  2.2s
=> => sha256:5a63271f8164c4a1fd42b70f29316cfe5ec83c34cb6de7289ecb802f6583a063  10.99MB / 10.99MB  2.5s
=> => sha256:76eaa9e5bd357d6983a88ddc9c4545ef4ad64c50f84f081b9a952c7ed08e3bdd6  1.86kB / 1.86kB  0.0s
=> => sha256:1075c6cab453237a9c37116de539158429e8b2e360c8a78ff042ae048ebadc5f  1.37kB / 1.37kB  0.0s
=> => sha256:564b1a543bd3c638c5878c22e852edcf4e1a2166aafb197b355990d63143d835  7.88kB / 7.88kB  0.0s
=> => sha256:8bcf4fd3160ad38c522f2edcf11277e643a6ebc542bde7014eff790bd4baa89  233B / 233B  2.8s
=> => sha256:537014b023270e5e978b116acc4a0d6a92f3f5ee48c2c48480e9d9cd392b3b10  2.64MB / 2.64MB  4.2s
=> => extracting sha256:b380bbd43752f83945df8b5d1074fef8dd044820e7d3aef33b655a2483e030c7  1.2s
=> => extracting sha256:14063a2781d6ddd50c6afb4a9e22cccf2205cb04a72caff066c71cf3e4d19c76  0.2s
=> => extracting sha256:5a63271f8164c4a1fd42b70f29316cfe5ec83c34cb6de7289ecb802f6583a063  0.4s
=> => extracting sha256:8bcf4fd3160ad38c522f2edcf11277e643a6ebc542bde7014eff790bd4baa89  0.0s
=> => extracting sha256:537014b023270e5e978b116acc4a0d6a92f3f5ee48c2c48480e9d9cd392b3b10  0.2s

```

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

```
PS C:\Users\mayuri> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sampleflask	latest	987292ed8aa0	28 seconds ago	126MB
nginx	latest	51086ed63d8c	15 hours ago	142MB
sonarqube	latest	2cf2f2494695	5 weeks ago	534MB

```
PS D:\37\sample-flask-app> |
```

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

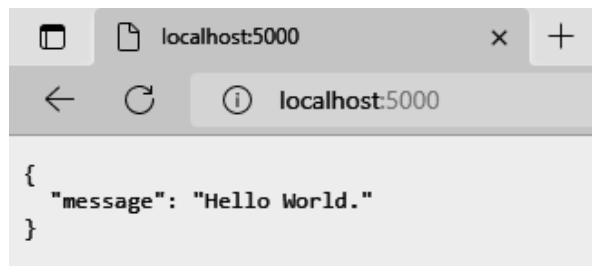
```
PS C:\Users\mayuri> docker container run -p 5000:5000 -d sampleflask
```

```
PS D:\37\sample-flask-app> docker container run -p 5000:5000 -d sampleflask
```

```
2cd668872b86f1df1b49e8e240268e5a46d5e6271d55f99a119424a99fed2c83
```

```
PS D:\37\sample-flask-app>
```

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



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