Experiment no 4

Title: Running a python application over docker.

Aim: To run a python program over docker.

Theory:

A Quick Overview Of The Concepts

In a nutshell, we are going to create a docker file that we will use to build a docker image which we will then run in a docker container.

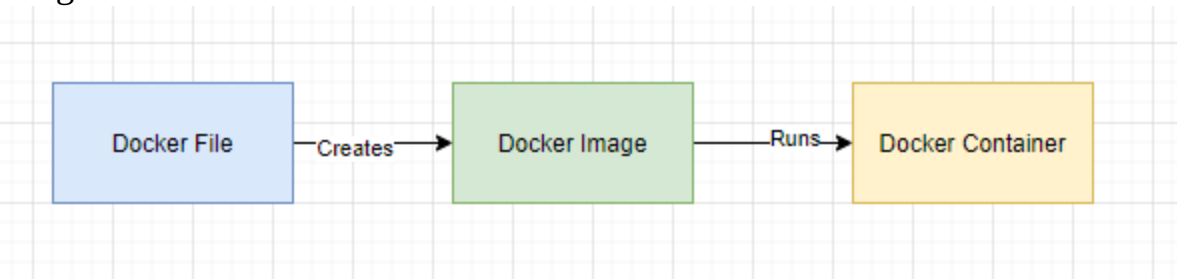


Image By Author

What is Docker?

Docker is a software platform. It enables software developers to develop, ship and run applications within its containers. Containers are lightweight software applications. We are going to build a Docker image in this experiment.

What is a docker file, image and container?

A docker file is a text file that contains the set of instructions for the Docker platform. Therefore, it can be versioned and committed to a code repository.

An image includes everything needed to run an application — the code or binary, runtime, dependencies, and any other file system objects required.

Docker containers run the application code.

Now that the theory is done, let's get started with a practical exercise.we are going to build a Python program and then run it inside a Docker container.

There are essentially 5 steps:

1. Create your python program (skip if you already have a Python program code)
2. Create a docker file
3. Build the docker file into an image
4. Run the docker image in a container
5. Test the Python program running within a container

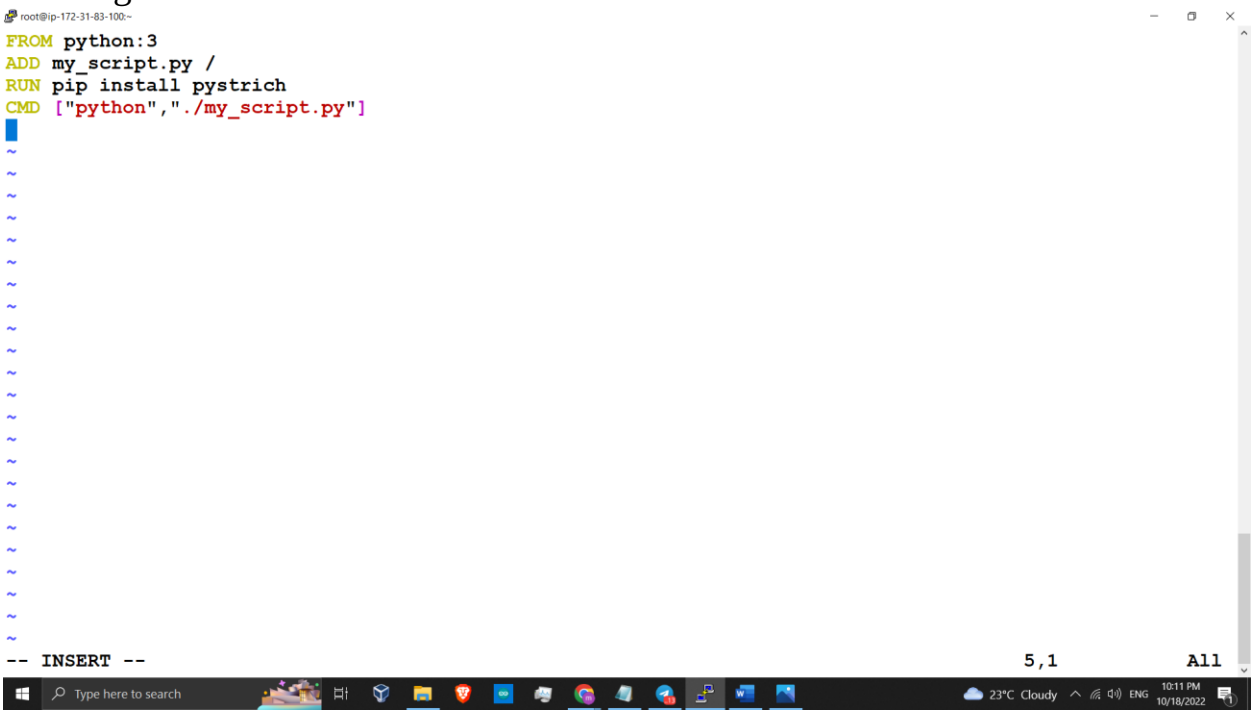
Containers are lightweight software applications. They offer a number of benefits including a reduction in time to deliver an application, scalability, ease of application management, support and better user experience.

While building Python applications, your working directories (workdir) play key roles. Pointing your application towards critical configuration files and other resources during runtime is critical. This allows your Python app to run effectively and predictably. Custom directories let your applications run in a well-defined state, which is perfect for testing and deployment.

Practical:

Step 1:

Creating a Dockerfile



- FROM tells Docker which image you base your image on (in the example, Python 3).
- RUN tells Docker which additional commands to execute.
- CMD tells Docker to execute the command when the image loads.

The Python script my_script.py looks like the following:

```
[root@ip-172-31-83-100 ~]# cat my_script.py
# Sample taken from pyStrich GitHub repository
from pystrich.datamatrix import DataMatrixEncoder

encoder = DataMatrixEncoder('This is a DataMatrix.')
encoder.save('./datamatrix_test.png')
print(encoder.get_ascii())
[root@ip-172-31-83-100 ~]#
```

root@ip-172-31-83-100:~

```
[root@ip-172-31-83-100 ~]# cat my_script.py
# Sample taken from pyStrich GitHub repository
from pystrich.datamatrix import DataMatrixEncoder

encoder = DataMatrixEncoder('This is a DataMatrix.')
encoder.save('./datamatrix_test.png')
print(encoder.get_ascii())
[root@ip-172-31-83-100 ~]# docker build -t python-barcode .
```

root@ip-172-31-83-100:~

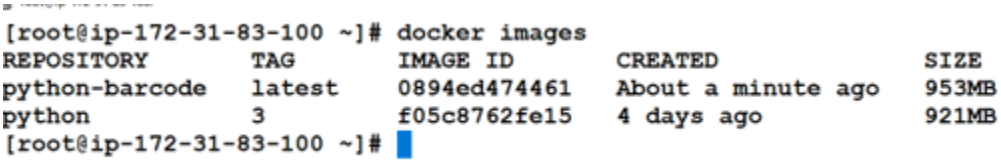
```
[root@ip-172-31-83-100 ~]# cat my_script.py
# Sample taken from pyStrich GitHub repository
from pystrich.datamatrix import DataMatrixEncoder

encoder = DataMatrixEncoder('This is a DataMatrix.')
encoder.save('./datamatrix_test.png')
print(encoder.get_ascii())
[root@ip-172-31-83-100 ~]# docker build -t python-barcode .
```

root@ip-172-31-83-100:~

```
[root@ip-172-31-83-100 ~]# docker build -t python-barcode .
Sending build context to Docker daemon  12.29kB
Step 1/4 : FROM python:3
3: Pulling from library/python
f606d8928ed3: Extracting  557.1kB/55.05MB
47db815c6a45: Download complete
bf4849400000: Download complete
a572f7a256d3: Downloading  29.14MB/54.58MB
8f7d05258955: Downloading  24.81MB/196.8MB
7110f04115ae: Download complete
967c89122295: Waiting
9f28dfdab382: Waiting
5e971b2ac572: Waiting
```

To see created image

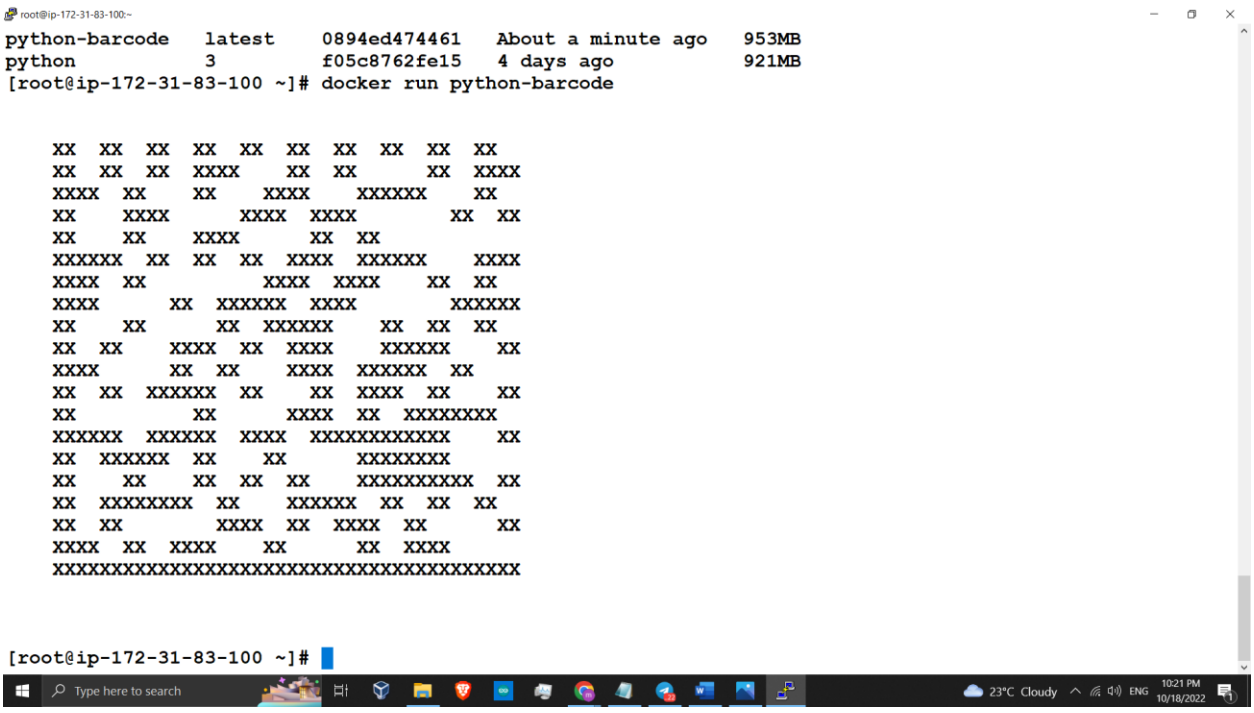
```
$ docker images
```



Run Your Image

After your image has been built successfully, you can run it as a container. In your terminal, run the command `docker images` to view your images. You should see an entry for “python-barcode”. Run the new image by entering:

```
docker run python-barcode
```



Conclusion: Thus we have successfully run the python application over docker.