

DevOps Lab

Documentation

Program 3:

Code a Dockerized Python Flask

Project Structure:

```
program_3/  
    -->Dockerfile  
    -->requirements.txt  
    -->app.py
```

Procedure:

Create a folder (ex: program_3)

Step 1: Create a **Dockerfile** without any extensions and add the following content
nano Dockerfile

```

GNU nano 7.2                               Dockerfile *
1RV24MC062_lubna_tabassum@Inspiron:~/DevOps/program_3
#Use Official Python Image
FROM python:latest

#Maintainer Information
LABEL maintainer="lubna@rvce.edu.in"

#Version Information
LABEL version="1.0"

#Set working directory
WORKDIR /app

#Copy the requirements file and install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
#Copy the application code
COPY . .
#Expose to port
EXPOSE 5000
#Run the application
CMD ["python", "app.py"]

^G Help          ^O Write Out    ^W Where Is    ^K Cut           ^T Execute      ^C Location
^X Exit         ^R Read File   ^\ Replace     ^U Paste        ^J Justify      ^/ Go To Line

```

Step 2: Create a [app.py](#) file with following code

nano app.py

```

GNU nano 7.2                               app.py
1RV24MC062_lubna_tabassum@Inspiron:~/DevOps/program_3
from flask import Flask
app=Flask(__name__)
@app.route("/")
def home():
    return "Hello from simple flask Docker!"

if __name__=="__main__":
    app.run(host="0.0.0.0",port=5000)

[ Read 9 lines ]
^G Help          ^O Write Out    ^W Where Is    ^K Cut           ^T Execute      ^C Location
^X Exit         ^R Read File   ^\ Replace     ^U Paste        ^J Justify      ^/ Go To Line

```

Step 3: Create requirements.txt file

The screenshot shows a terminal window titled "requirements.txt" with the command "GNU nano 7.2". Inside the editor, there is one line of text: "Flask==2.3.3". At the bottom of the screen, there is a menu bar with various keyboard shortcuts for file operations like Help, Exit, Read, Write, Find, Replace, Cut, Paste, Execute, Justify, Location, and Go To Line.

Step 4: Execute the Docker Build command

```
docker build -t program_3 .
```

The screenshot shows a terminal window with the command "docker build -t program_3 ." being run. The output shows the build process, which took 14.3s. It lists various steps such as loading the Dockerfile, transferring metadata, loading .dockerignore, transferring context, pulling Python from docker.io, and finally exporting the image. The total time for the build was 0.0s.

```
1RV24MC062_lubna_tabassum@Inspiron:~$ cd DevOps
1RV24MC062_lubna_tabassum@Inspiron:~/DevOps$ cd program_3
1RV24MC062_lubna_tabassum@Inspiron:~/DevOps/program_3$ docker build -t program_3 .
[+] Building 14.3s (10/10) FINISHED
   docker:default
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 480B                         0.0s
=> [internal] load metadata for docker.io/library/python:latest 14.1s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                           0.0s
=> [1/5] FROM docker.io/library/python:latest@sha256:92b51c6fa86611bbf2000abf360a206 0.0s
=> => resolve docker.io/library/python:latest@sha256:92b51c6fa86611bbf2000abf360a206 0.0s
=> [internal] load build context                          0.0s
=> => transferring context: 93B                         0.0s
=> CACHED [2/5] WORKDIR /app                            0.0s
=> CACHED [3/5] COPY requirements.txt .                  0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> CACHED [5/5] COPY . .                                0.0s
=> exporting to image                                    0.0s
=> => exporting layers                                 0.0s
=> => writing image sha256:b1d2ce2a103af0fc9c27ea1cb569b6c9738e36852e12bf1bafa30c38d 0.0s
=> => naming to docker.io/library/program_3            0.0s
```

Step 5: Run the docker command and specify the port number

```
docker run -d -p 3000:3000 program_3
```

```
1RV24MC062_lubna_tabassum@Inspiron:~/DevOps/program_3$ docker run -p 5000:5000 program_3
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

Step 6: Verify localhost details on browser

<http://localhost:5000>

