

DevOps Lab – Program 1

CREATING A DOCKER FILE

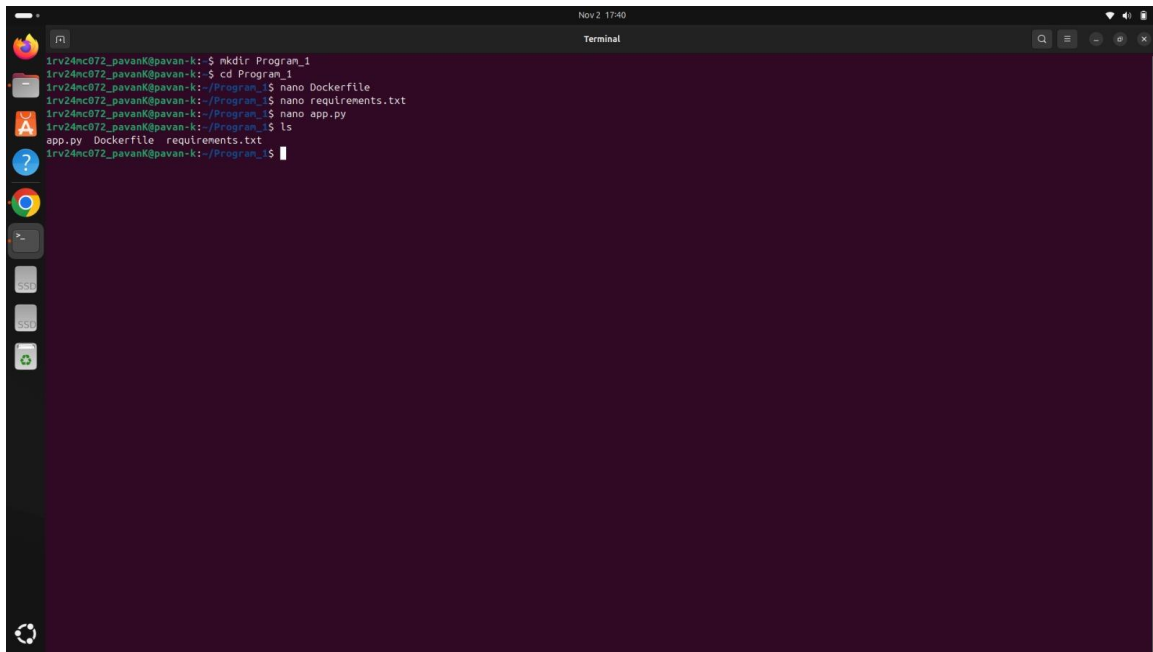
Project Structure

```
program_1/  
├──  
├── Dockerfile  
├── app.py  
└── requirements.txt
```

Step 1 – Set Up Project Directory

```
mkdir program_1
```

```
cd program_1
```

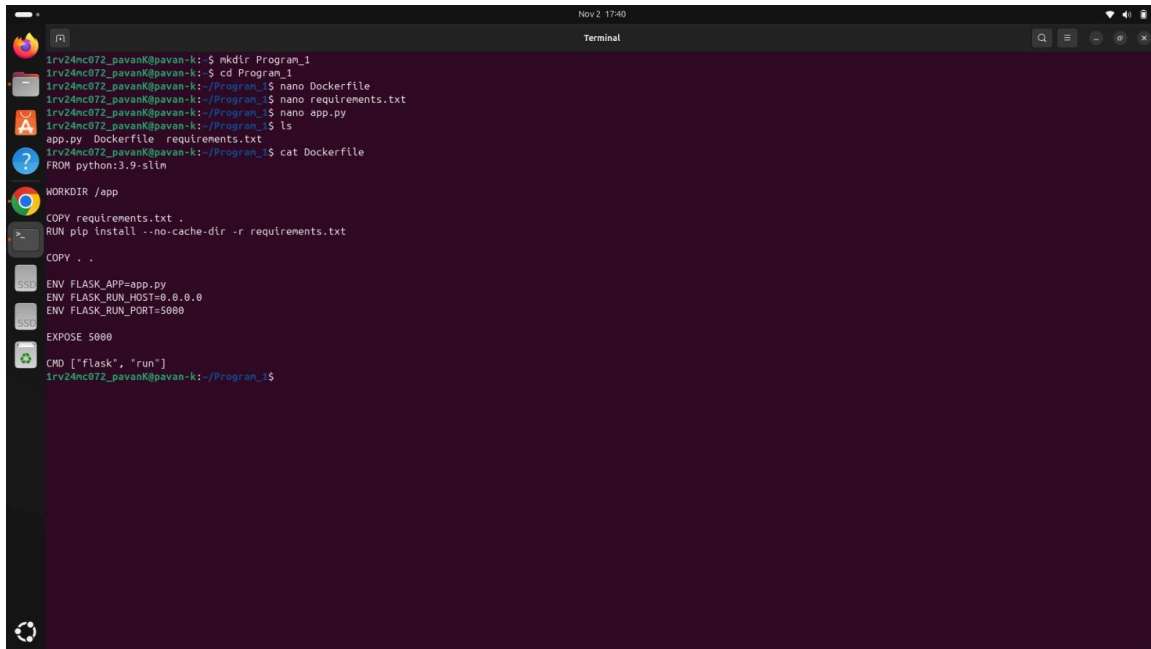
A terminal window titled "Terminal" with a dark background and light text. The window shows a series of commands and their outputs. The user is in a directory named "Program_1". The commands executed are: "mkdir Program_1", "cd Program_1", "nano Dockerfile", "nano requirements.txt", "nano app.py", and "ls". The output of "ls" shows the files "app.py", "Dockerfile", and "requirements.txt". The terminal window has a standard Ubuntu-style top bar with the date "Nov 2, 17:40" and system icons on the right. On the left, there is a vertical dock with icons for the Dash menu, Home, Files, Applications, and several other applications.

```
Nov 2, 17:40  
Terminal  
1rv24nc072_pavan@pavan-k:~$ mkdir Program_1  
1rv24nc072_pavan@pavan-k:~$ cd Program_1  
1rv24nc072_pavan@pavan-k:~/Program_1$ nano Dockerfile  
1rv24nc072_pavan@pavan-k:~/Program_1$ nano requirements.txt  
1rv24nc072_pavan@pavan-k:~/Program_1$ nano app.py  
1rv24nc072_pavan@pavan-k:~/Program_1$ ls  
app.py  Dockerfile  requirements.txt  
1rv24nc072_pavan@pavan-k:~/Program_1$
```

Step 2 – Create the Dockerfile

Open a new Dockerfile:

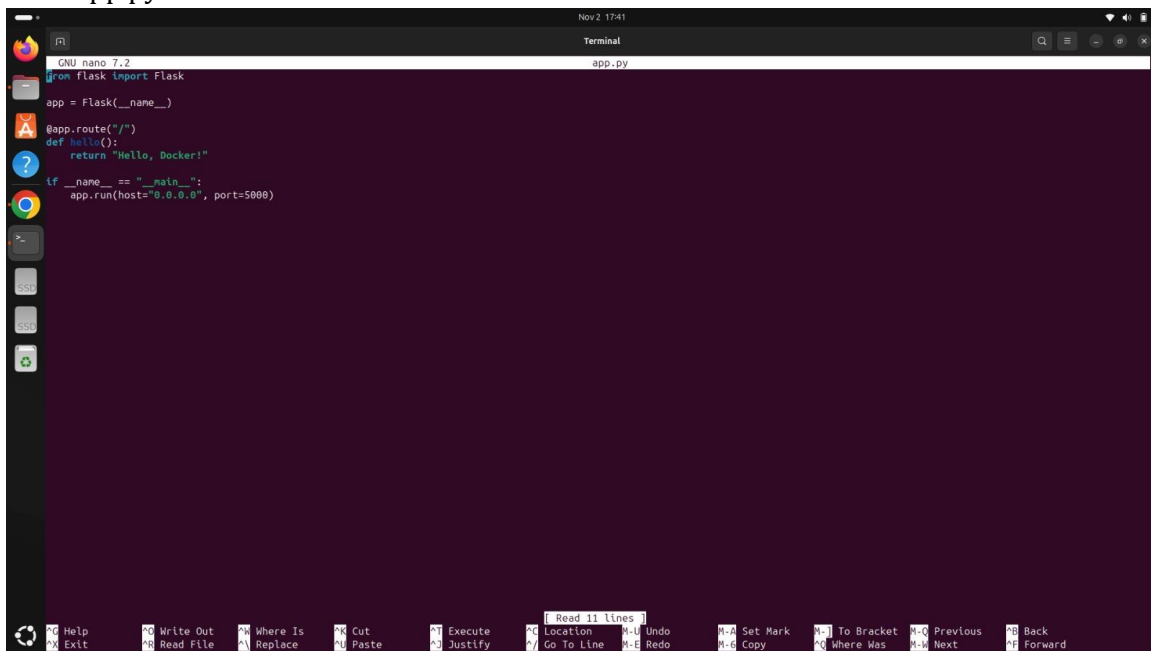
```
sudo nano Dockerfile
```

A terminal window titled 'Terminal' showing the creation and building of a Docker image. The user creates a directory 'Program_1', enters it, and creates a 'Dockerfile' and 'requirements.txt' using 'nano'. The 'Dockerfile' contains instructions to copy 'app.py' and 'requirements.txt', run 'pip install', and expose port 5000. The 'requirements.txt' contains 'flask'. The user then runs 'docker build -t flask .', which successfully builds the image 'flask' with ID 'sha256:11'.

Step 3 – Write the Flask Application

Create the main Python file:

nano app.py

A terminal window titled 'Terminal' showing the content of 'app.py' in nano editor. The code defines a Flask application with a single route '/' that returns 'Hello, Docker!'. The application is run on host '0.0.0.0' and port '5000'. The terminal output shows the application running successfully.

Step 4 – Specify Dependencies

Create the requirements file:

nano requirements.txt

```
Nov 2 17:40
Terminal

1rv24nc072_pavan@pavan-ki:~$ mkdir Program_1
1rv24nc072_pavan@pavan-ki:~$ cd Program_1
1rv24nc072_pavan@pavan-ki:~/Program_1$ nano Dockerfile
1rv24nc072_pavan@pavan-ki:~/Program_1$ nano requirements.txt
1rv24nc072_pavan@pavan-ki:~/Program_1$ nano app.py
1rv24nc072_pavan@pavan-ki:~/Program_1$ ls
app.py  Dockerfile  requirements.txt
1rv24nc072_pavan@pavan-ki:~/Program_1$ cat Dockerfile
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0
ENV FLASK_RUN_PORT=5000

EXPOSE 5000

CMD ["flask", "run"]
1rv24nc072_pavan@pavan-ki:~/Program_1$ cat requirements.txt
flask
1rv24nc072_pavan@pavan-ki:~/Program_1$
```

Step 5 – Build the Docker Image

Execute:

`docker build -t program_1 .`

```
Nov 2 17:44
Terminal

1rv24nc072_pavan@pavan-ki:~/Program_1$ docker build -t flask-demo:1.0 .
[+] Building 17.0s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 268B
=> [internal] load metadata for docker.io/library/python:3.9-slim
=> [internal] load .dockerignore
=> transferring context: 2B
=> resolve docker.io/library/python:3.9-slim@sha256:2d97f6910b16bd33bd3860f261f53f144965f755599aabb1acd4e13cf1731b1b
=> sha256:fc74430849022d13bd44b0969a953f842f59c6e9d1abc2c83d71baffa286c8b 13.80MB / 13.80MB
=> sha256:2d97f6910b16bd33bd3860f261f53f144965f755599aabb1acd4e13cf1731b1b 10.36kB / 10.36kB
=> sha256:0ad0b29e58ec35ef7d22736f444e725d71b19acd07377b4465996c0e0d 1.74kB / 1.74kB
=> sha256:0d5d4630c10b449524c3f6a07f50a3bffe4418b090cfc09f372871f497 5.40kB / 5.40kB
=> sha256:38513bd7256313495cdd83b3b0915a633cfa75dc2a07072ab2c8d191826c5d 29.78MB / 29.78MB
=> sha256:b3ec19b3aee8c83a3e09854de4ac4aa08381dfed84a9da075048c2e3df3881d 1.29MB / 1.29MB
=> sha256:ee56f685404adfb1680322f152d2cfce2115b30dd4a81c2c450078315bb5b08 251B / 251B
=> extracting sha256:38513bd7256313495cdd83b3b0915a633cfa75dc2a07072ab2c8d191826c5d 0.75
=> extracting sha256:b3ec19b3aee8c83a3e09854de4ac4aa08381dfed84a9da075048c2e3df3881d 0.13
=> extracting sha256:fc74430849022d13bd44b0969a953f842f59c6e9d1abc2c83d71baffa286c8b 0.55
=> extracting sha256:ee56f685404adfb1680322f152d2cfce2115b30dd4a81c2c450078315bb5b08 0.06
=> [internal] load build context
=> transferring context: 529B
=> [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt .
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY . .
=> exporting to image
=> exporting layers
=> writing image sha256:27f05579f440bc40ed72e55c5b410a3c6057403ee7f3edcc847a7a0759837d
=> naming to docker.io/library/flask-demo:1.0
1rv24nc072_pavan@pavan-ki:~/Program_1$
```

Step 6 – Run the Container

Launch the container:

`docker run -d -p 5000:5000 program_1`

```
Nov 2 17:45
Terminal

1rv24nc072_pavanK@pavan-k:~/Program_1$ docker build -t flask-demo:1.0 .
[+] Building 17.0s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load metadata for docker.io/library/python:3.9-slim
=> [internal] load .dockerignore
=> transferring context: 2B
=> [1/5] FROM docker.io/library/python:3.9-slim@sha256:2d97f6918b16bd338d1868f261f53f144965f755599aeb1acd4e13cf1731b1b
=> resolve docker.io/library/python:3.9-slim@sha256:2d97f6918b16bd338d1868f261f53f144965f755599aeb1acd4e13cf1731b1b
=> sha256:fc74318049822d138045b8969a953f942f59c6e9d1a8c2c83d71bffa286c80 / 13.08MB
=> sha256:2d97f6918b16bd338d1868f261f53f144965f755599aeb1acd4e13cf1731b1b 10.36kB / 10.36kB
=> sha256:dad5b29e3506c35e8f0222736f44ef25d21b219acdd73f7bb41d59996cae8d 1.74kB / 1.74kB
=> sha256:085da638e1b8a49514c3fda83ff58a3bffa4410b05bfcac087e5722871f497 5.40kB / 5.40kB
=> sha256:1851b07256113495cd083b300915a631cfa75dc2a07872a05c8d191809c45d 29.78MB / 29.78MB
=> sha256:b3ec39b16a6c8b3a409854de4ec4a0831dfed84a9da0f5840c2e3d0f3081d 1.29MB / 1.29MB
=> sha256:ea56f6b504a4f01608322f152d2cfec2115b300d4081c2c45078315b05500 251B / 251B
=> extracting sha256:1851b07256113495cd083b300915a631cfa75dc2a07872a05c8d191809c45d
=> extracting sha256:b3ec39b16a6c8b3a409854de4ec4a0831dfed84a9da0f5840c2e3d0f3081d
=> extracting sha256:fc74318049822d138045b8969a953f942f59c6e9d1a8c2c83d71bffa286c80
=> [internal] load build context
=> transferring context: 329B
=> [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt .
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY .
=> exporting layers
=> writing image sha256:27f0db579f44bbc40ed72e55c5b410a3c60574039ee7f3adcc847a7a8759837d
=> naming to docker.io/library/flask-demo:1.0
1rv24nc072_pavanK@pavan-k:~/Program_1$ docker run --rm -p 5000:5000 --name flask-demo-container flask-demo:1.0
* Serving Flask app 'app.py'
* 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 7 – Test the Application

Open a web browser and visit:

<http://localhost:5000>

