

Program3

Code a Dockerized python Flask or Node.js application

Step 1 - Create the following file structure

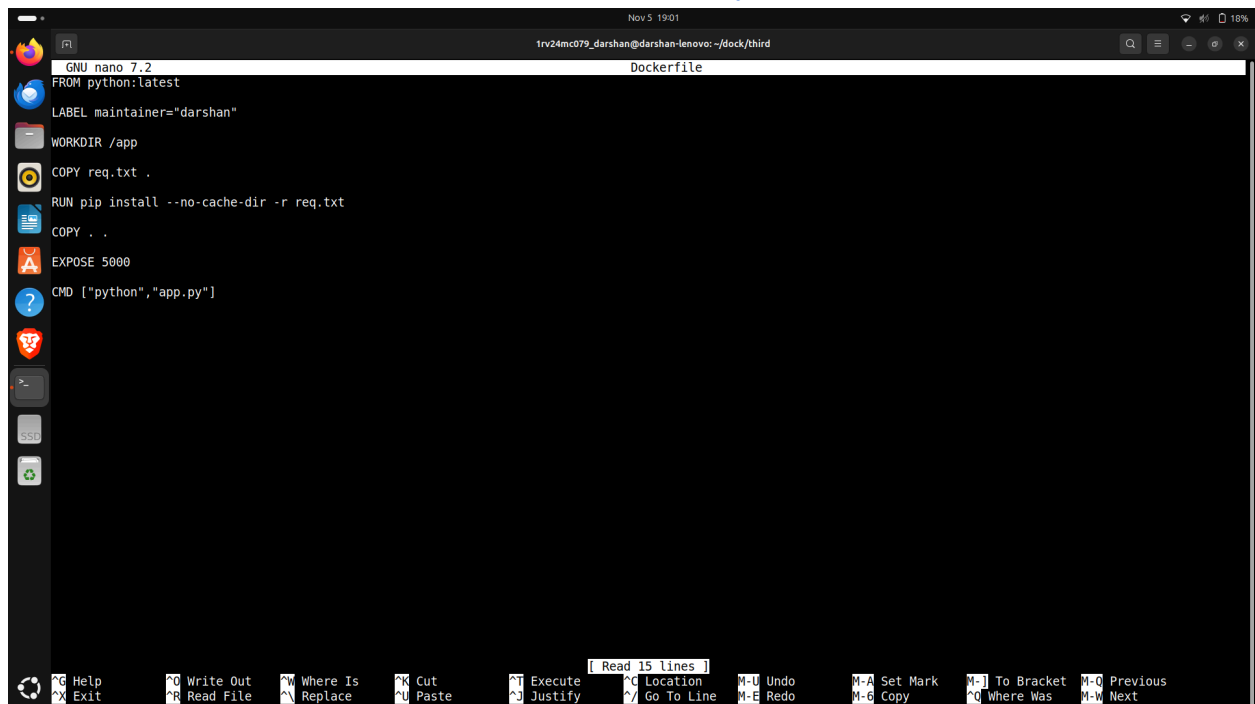
prgm3

|----- Dockerfile

|----- app.py

|----- requirements.txt

Step 4 - Write the Dockerfile, requirements.txt and [app.py](#)



The screenshot shows a terminal window with the nano 7.2 text editor open, editing a file named 'Dockerfile'. The terminal title bar indicates the user is 'trv24mc079_darshan@darshan-lenovo: ~/dock/third' and the date is 'Nov 5 19:01'. The Dockerfile content is as follows:

```
GNU nano 7.2 Dockerfile
FROM python:latest

LABEL maintainer="darshan"

WORKDIR /app

COPY req.txt .

RUN pip install --no-cache-dir -r req.txt

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]
```

The bottom of the terminal shows the nano editor's command palette with various shortcuts like 'Help', 'Exit', 'Write Out', 'Read File', 'Where Is', 'Replace', 'Cut', 'Paste', 'Execute', 'Justify', 'Location', 'Go To Line', 'Undo', 'Redo', 'Set Mark', 'Copy', 'To Bracket', 'Where Was', 'Previous', and 'Next'. A 'Read 15 lines' indicator is also visible.

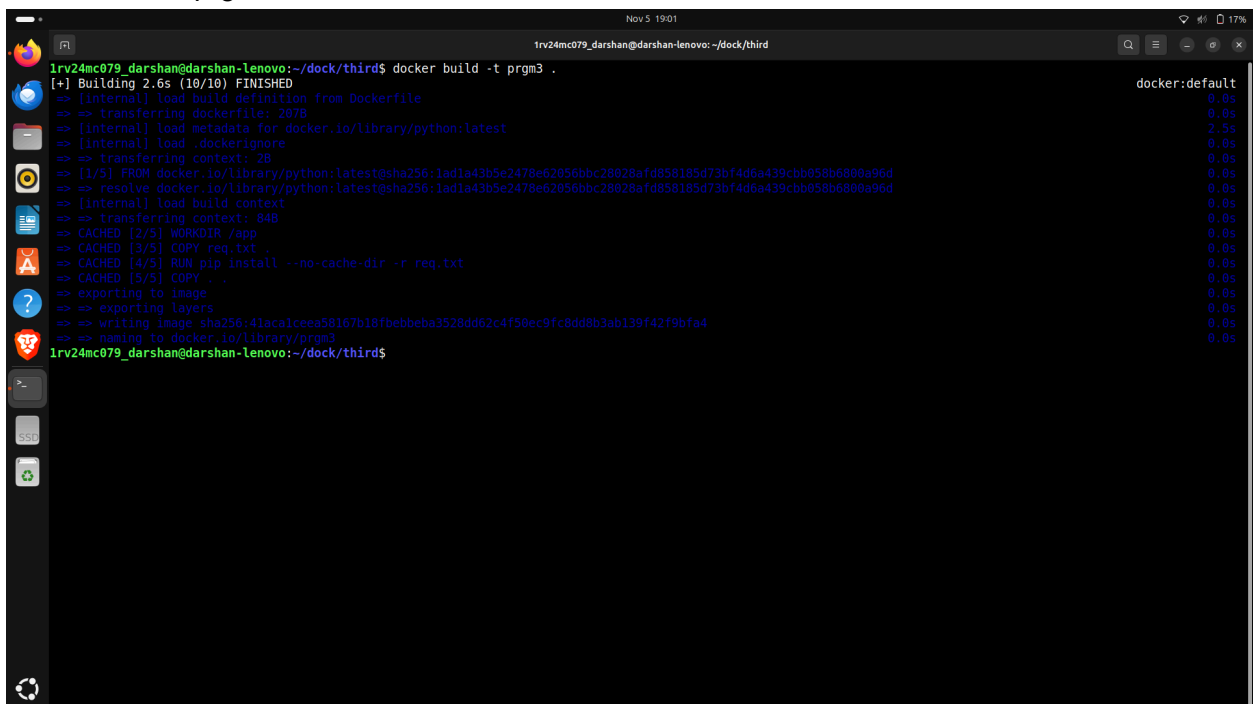


The screenshot shows a terminal window with a nano editor open. The editor is editing a file named `app.py`. The code in the file is a simple Flask application that returns "Hello from python " when the root path is accessed. The terminal window title is `Nov 5 19:01` and the user is `trv24mc079_darshan@darshan-lenovo: ~/dock/third`. The nano editor's status bar at the bottom shows various keyboard shortcuts like `Ctrl+G Help`, `Ctrl+X Exit`, etc.

```
GNU nano 7.2 app.py
from flask import Flask
app = Flask(__name__)
@app.route("/")
def home():
    return("Hello from python ")
app.run(host="0.0.0.0",port=5000)
```

Step 5 - Build the Docker image

`docker build -t prgm3 .`



The screenshot shows a terminal window where the command `docker build -t prgm3 .` has been executed. The output shows the progress of building the Docker image, including downloading the base image (python:latest) and installing dependencies. The final output is `docker:default` and the image is named `prgm3`. The terminal window title is `Nov 5 19:01` and the user is `trv24mc079_darshan@darshan-lenovo: ~/dock/third`.

```
trv24mc079_darshan@darshan-lenovo:~/dock/third$ docker build -t prgm3 .
[+] Building 2.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 287B
=> [internal] load metadata for docker.io/library/python:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/python:latest@sha256:1ad1a43b5e2478e62856bbc28028afd858185d73bf4d6a439cbb050b6800a96d
=> => resolve docker.io/library/python:latest@sha256:1ad1a43b5e2478e62856bbc28028afd858185d73bf4d6a439cbb050b6800a96d
=> [internal] load build context
=> => transferring context: 84B
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY req.txt .
=> CACHED [4/5] RUN pip install --no-cache-dir -r req.txt
=> CACHED [5/5] COPY . .
=> => exporting to image
=> => exporting layers
=> => writing image sha256:41acalceea58167b18fbebba3528dd62c4f50ec9fc8dd8b3ab139f42f9bfa4
=> => naming to docker.io/library/prgm3
trv24mc079_darshan@darshan-lenovo:~/dock/third$
```

Step 6 - Build and Run the container

`docker run -p 5000:5000 prgm3`

Step 7 - Verify the output

<https://localhost:5000>

