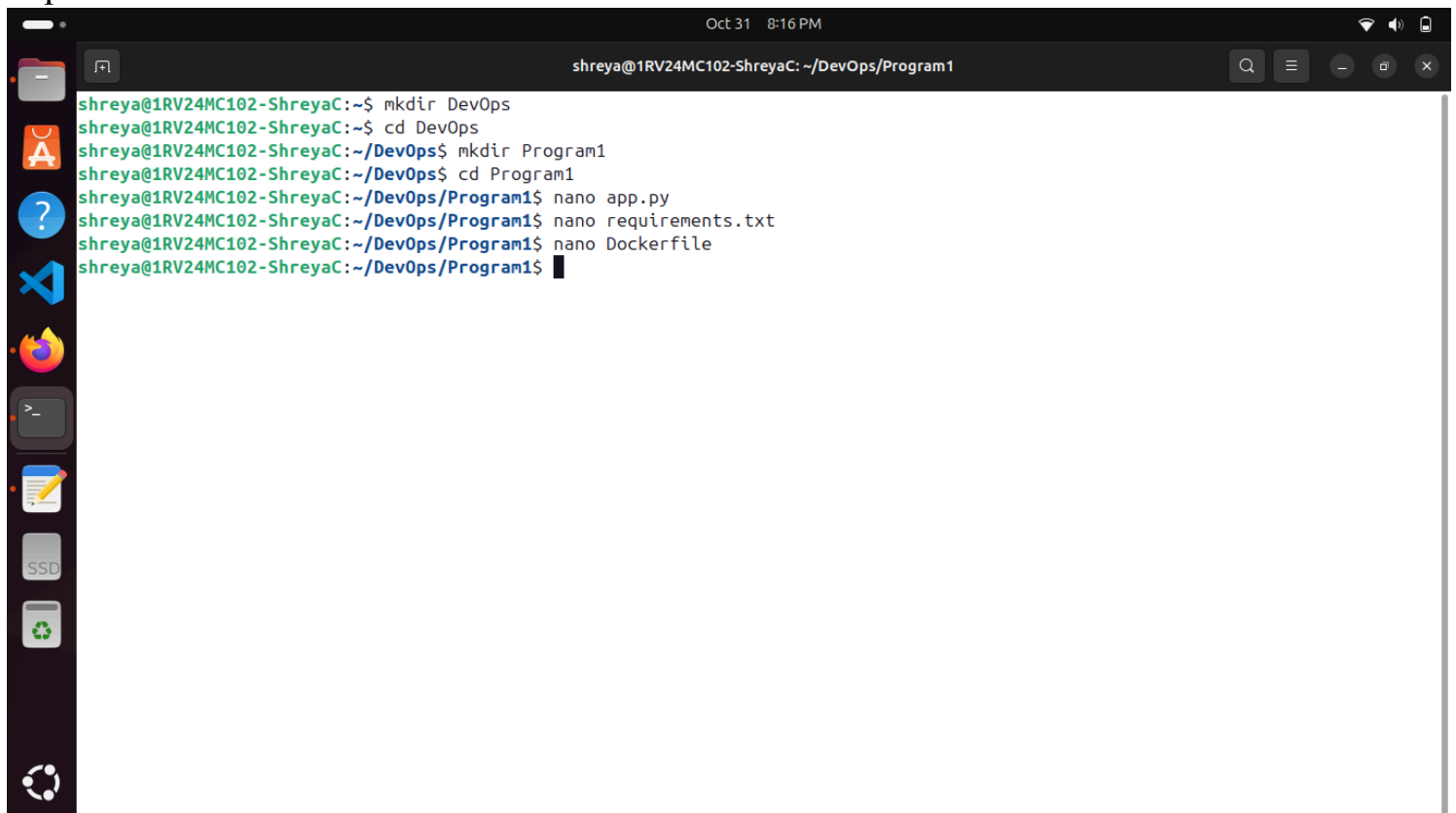


Program 1 : Build a Docker Container from a Custom Dockerfile

Folder Structure

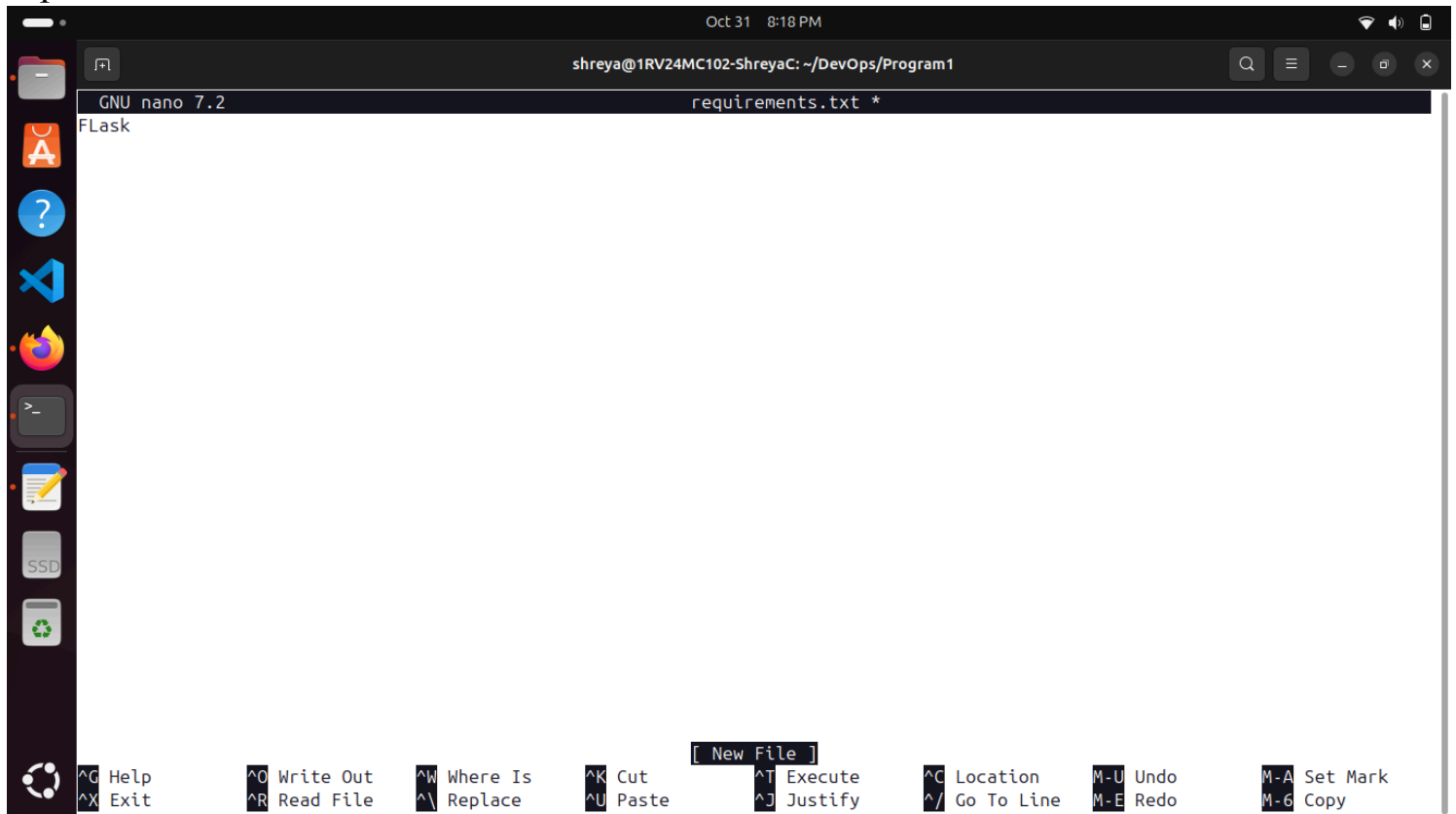
```
DevOps
|-- Program1
|   |-- app.py
|   |-- requirements.txt
|   |-- Dockerfile
```

Step 1: To build a docker container from custom Dockerfile, I've created a directory called DevOps navigate to DevOps directory and within that create a sub-directory with name Program1, navigate to Program1 directory and create the files such as [app.py](#), requirements.txt and Dockerfile.



```
shreya@1RV24MC102-ShreyaC: ~/DevOps/Program1
shreya@1RV24MC102-ShreyaC:~$ mkdir DevOps
shreya@1RV24MC102-ShreyaC:~$ cd DevOps
shreya@1RV24MC102-ShreyaC:~/DevOps$ mkdir Program1
shreya@1RV24MC102-ShreyaC:~/DevOps$ cd Program1
shreya@1RV24MC102-ShreyaC:~/DevOps/Program1$ nano app.py
shreya@1RV24MC102-ShreyaC:~/DevOps/Program1$ nano requirements.txt
shreya@1RV24MC102-ShreyaC:~/DevOps/Program1$ nano Dockerfile
shreya@1RV24MC102-ShreyaC:~/DevOps/Program1$
```

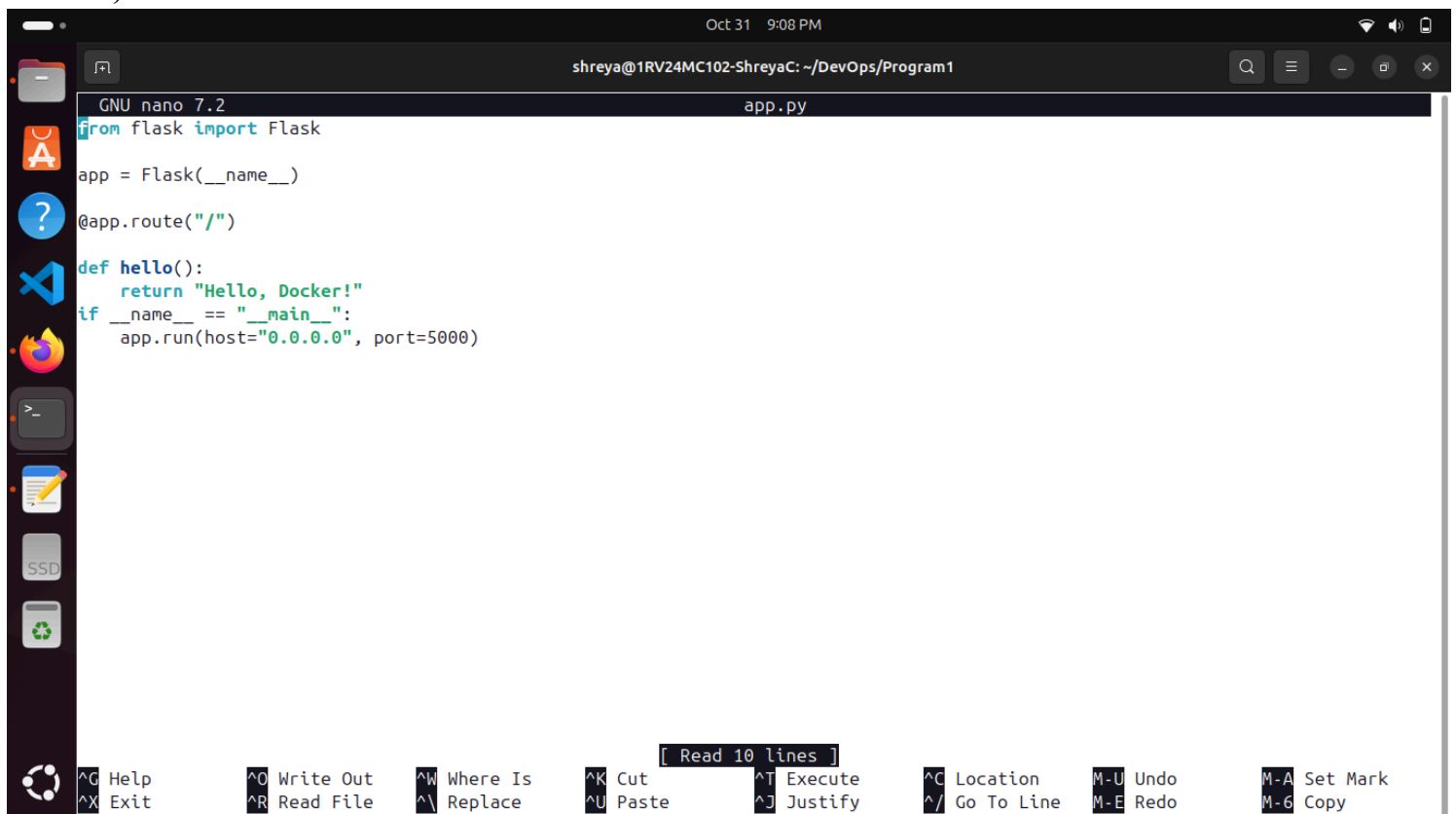
Step 2: Add the following code in requirements.txt file, which is used to install the dependencies.



```
GNU nano 7.2 requirements.txt *
Flask
```

Help Exit Write Out Read File Where Is Replace Cut Paste [New File] Execute Justify Location Go To Line Undo Redo Set Mark Copy

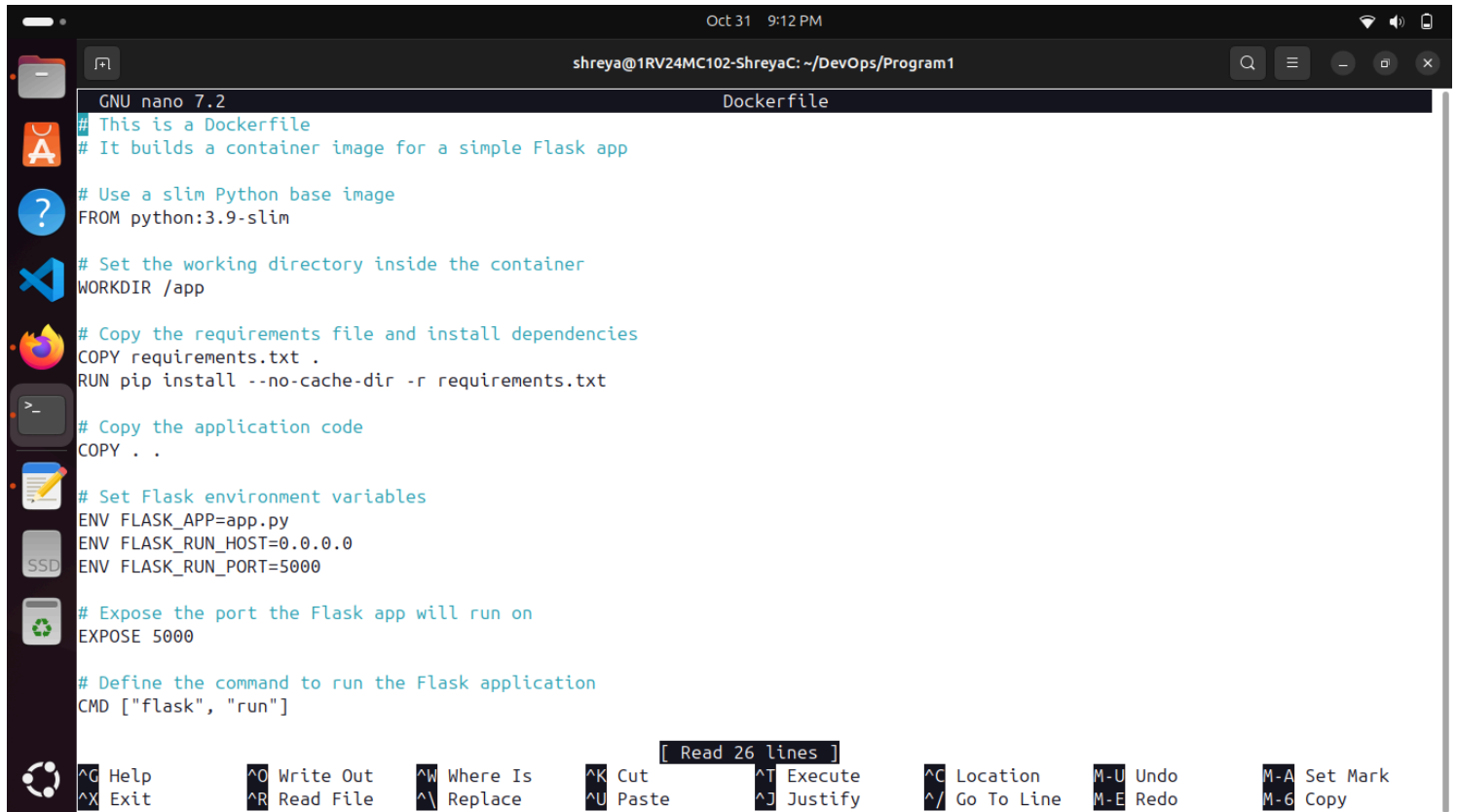
Step 3: Code the following in [app.py](#) file, this code runs a simple web server that shows **“Hello, Docker!”** when we visit.



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

Help Exit Write Out Read File Where Is Replace Cut Paste [Read 10 lines] Execute Justify Location Go To Line Undo Redo Set Mark Copy

Step 4: Code the following in Dockerfile, it creates a lightweight container that installs Python and Flask dependencies, copies the code to /app and runs our Flask server on port 5000.



```
GNU nano 7.2 Dockerfile
# This is a Dockerfile
# It builds a container image for a simple Flask app

# Use a slim Python base image
FROM python:3.9-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the requirements file and install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

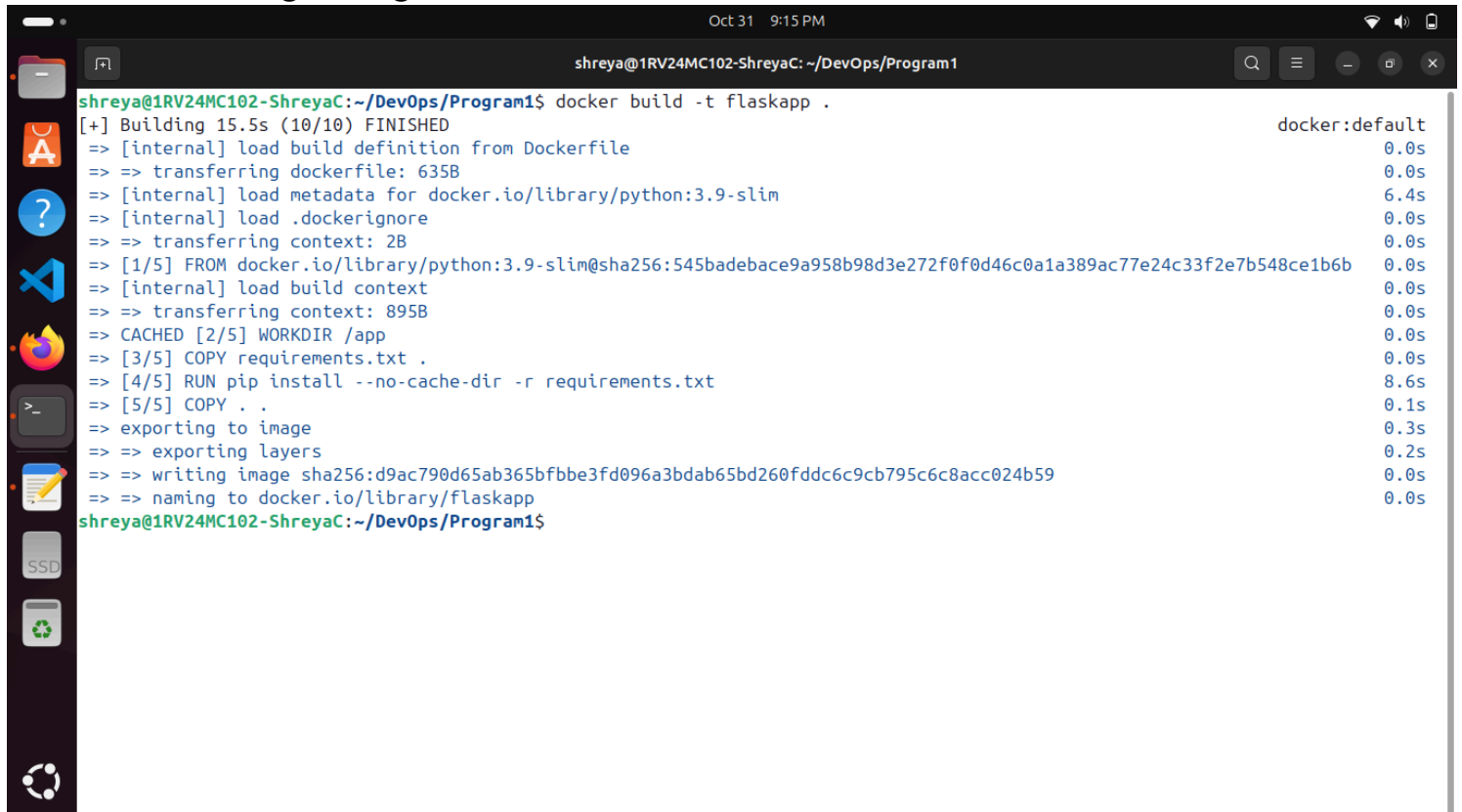
# Copy the application code
COPY . .

# Set Flask environment variables
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0
ENV FLASK_RUN_PORT=5000

# Expose the port the Flask app will run on
EXPOSE 5000

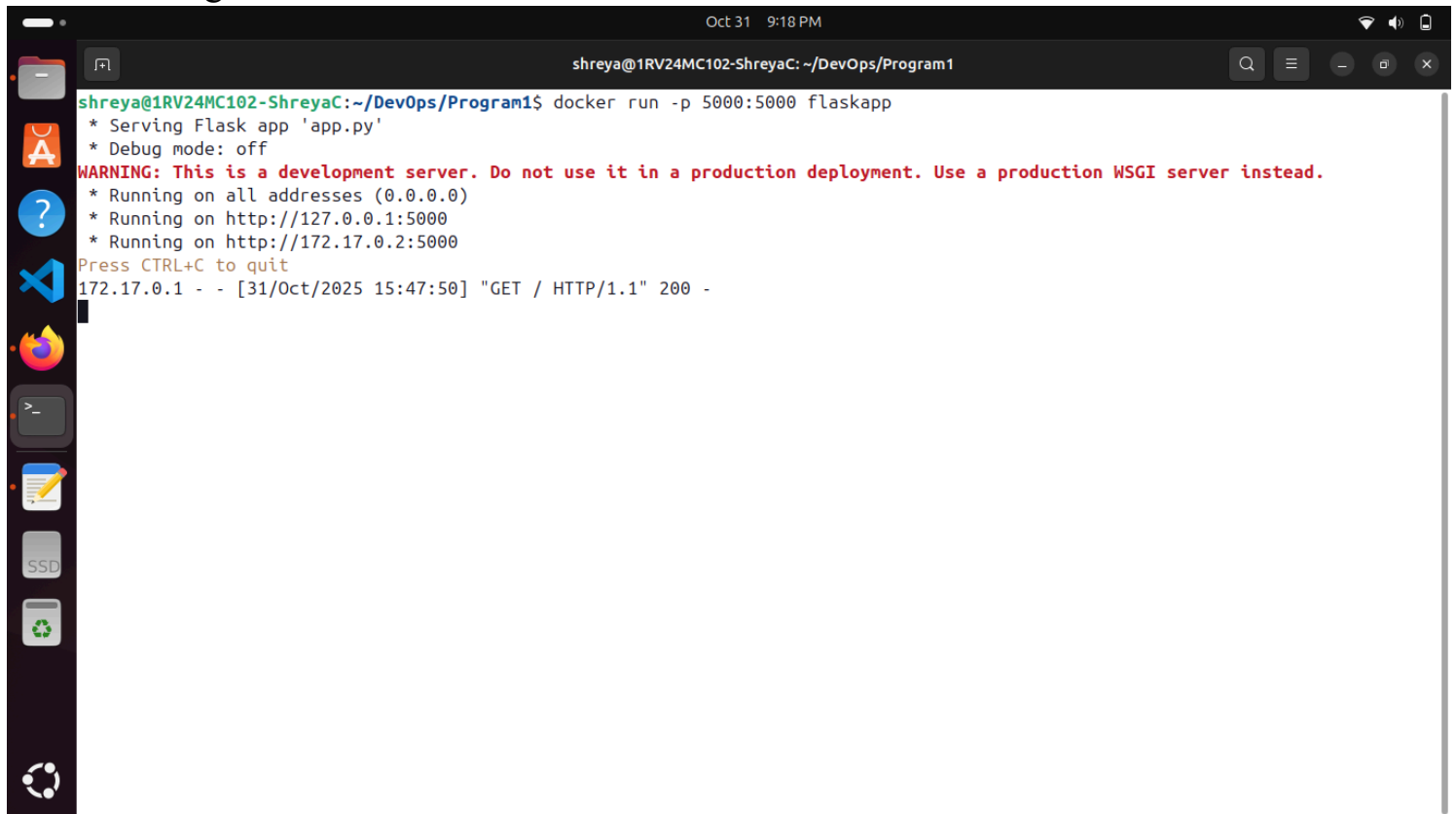
# Define the command to run the Flask application
CMD ["flask", "run"]
```

Step 5: Now build the docker container using docker build command which tells the Docker to create new image using Dockerfile.



```
shreya@1RV24MC102-ShreyaC:~/DevOps/Program1$ docker build -t flaskapp .
[+] Building 15.5s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 635B
=> [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@sha256:545badebace9a958b98d3e272f0f0d46c0a1a389ac77e24c33f2e7b548ce1b6b
=> [internal] load build context
=> => transferring context: 895B
=> CACHED [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:d9ac790d65ab365bfbbe3fd096a3bdab65bd260fddc6c9cb795c6c8acc024b59
=> => naming to docker.io/library/flaskapp
shreya@1RV24MC102-ShreyaC:~/DevOps/Program1$
```

Step 6: Run the Docker container using docker run command, which creates the container from the image we built.



A terminal window titled "shreya@1RV24MC102-ShreyaC: ~/DevOps/Program1" showing the execution of a Docker container. The command `docker run -p 5000:5000 flaskapp` has been run. The output shows the Flask application starting, serving on port 5000, and receiving a GET request from 172.17.0.1. A warning message is displayed: "WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead." The terminal also shows the IP addresses it is running on: 0.0.0.0, 127.0.0.1, and 172.17.0.2. The prompt "Press CTRL+C to quit" is visible.

```
shreya@1RV24MC102-ShreyaC:~/DevOps/Program1$ docker run -p 5000:5000 flaskapp
* 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
172.17.0.1 - - [31/Oct/2025 15:47:50] "GET / HTTP/1.1" 200 -
```

