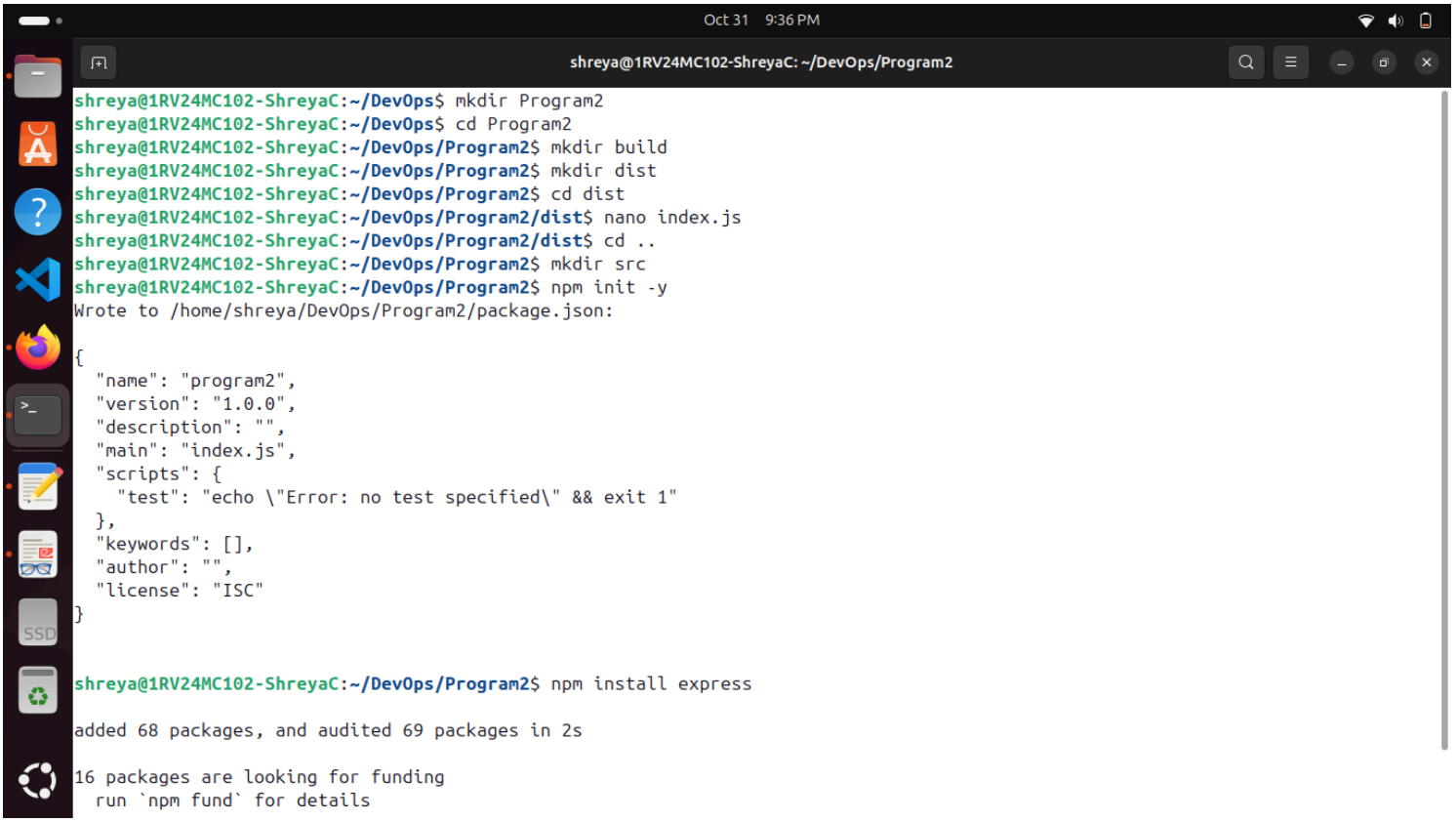
Shreya Rajesh Chadichal
1RV24MC102

Program 2: Develop a Multi-Stage Dockerfile for Container Orchestration.

Folder Structure:

```
DevOps
    |-- Program2
        |-- build
        |-- dist
            |-- index.js
        |-- src
        |-- Dockerfile
        |-- package.json
        |-- package-lock.json
        |-- node_modules
```

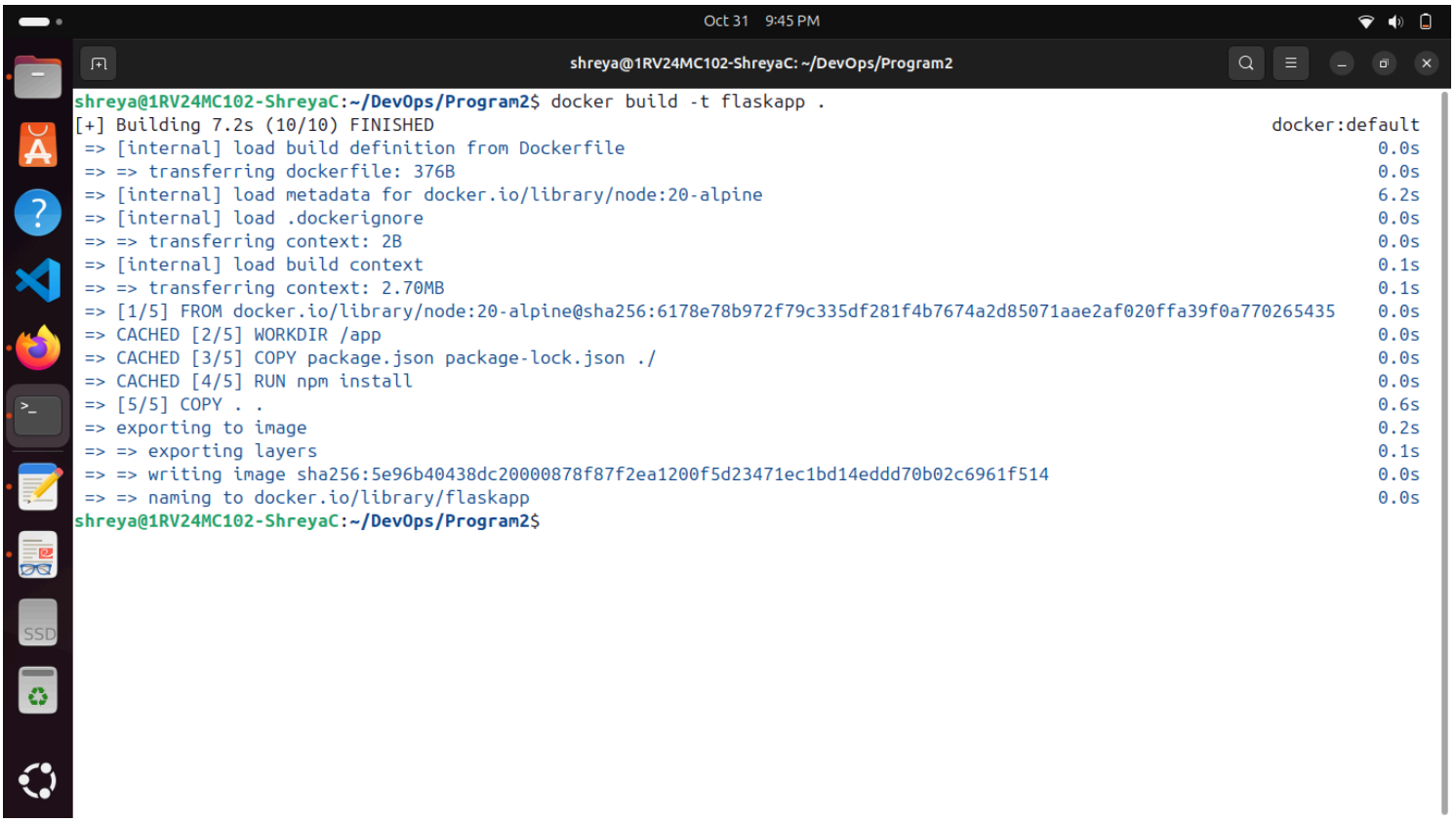
Step1 : I've created a new folder called Program2 to hold Node.js project, navigated inside it, and made three subfolders build, dist, and src to organize the code. Then I've opened the dist folder and created a file named index.js where the main server code will go. Also, created a file name Dockerfile which installs the dependencies. After that, navigated back to the main project folder and ran npm init -y to quickly create a default package.json file that keeps project info and dependencies. Finally, installed Express, a popular Node.js framework, so that we can easily build and run web server.



Step 2: This code creates a simple web server using Express that runs on port 8000. When we open http://localhost:8000

Step 3: This Dockerfile creates a lightweight Node.js container for our app. It starts with the Node 20 Alpine image, sets /app as the working folder, installs dependencies from package.json, copies the rest of our files, opens port 5000 for access, and finally runs the app using node dist/index.js.

Step 4: Now build the docker container using docker build command which tells the Docker to create new image using Dockerfile.



Step 5: Run the Docker container using docker run command, which creates the container from the image we built.