

DevOps Lab

Program -2

Creating a Multi-Stage Dockerfile

Project Structure

program_2/

|

├── Dockerfile

├── package.json

├── package-lock.json

├── node_modules/

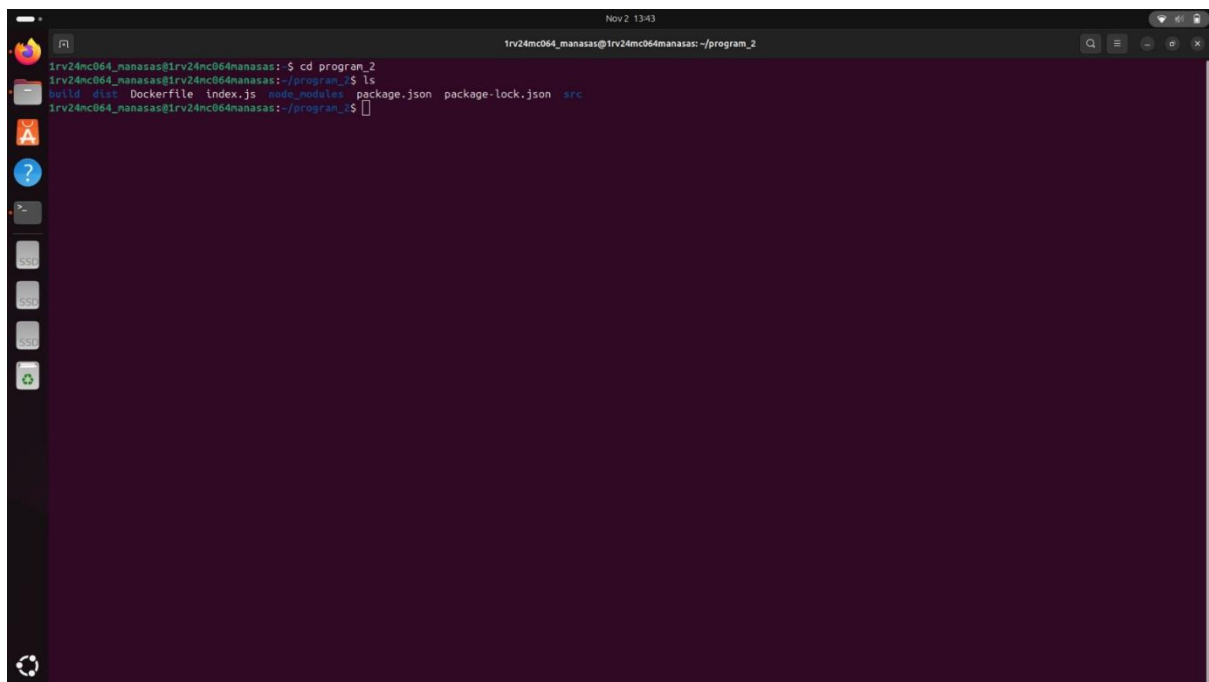
└── src/

└── index.js

Step 1: Create Project Folder

```
mkdir program_2
```

```
cd program_2
```

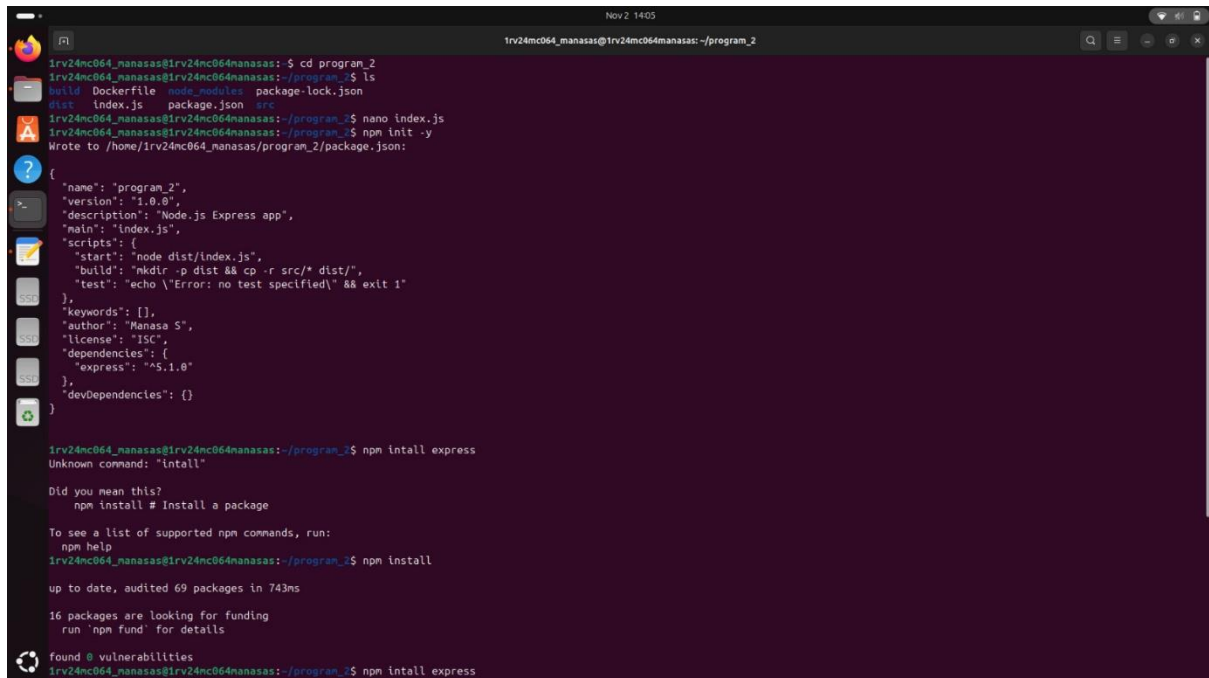


Step 2: Initialize Node.js Project

```
npm init -y
```

Step 3: Install Express Framework

npm install express



A terminal window showing the process of installing the Express framework. The user is in a directory named 'program_2'. They run 'ls' and see files: 'build', 'Dockerfile', 'node_modules', 'package-lock.json', 'dist', 'index.js', 'package.json', and 'src'. They then run 'nano index.js' and create a package.json file with the following content:

```
{
  "name": "program_2",
  "version": "1.0.0",
  "description": "Node.js Express app",
  "main": "index.js",
  "scripts": {
    "start": "node dist/index.js",
    "build": "mkdir -p dist && cp -r src/* dist/",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "Manasa S",
  "license": "ISC",
  "dependencies": {
    "express": "^5.1.0"
  },
  "devDependencies": {}
}
```

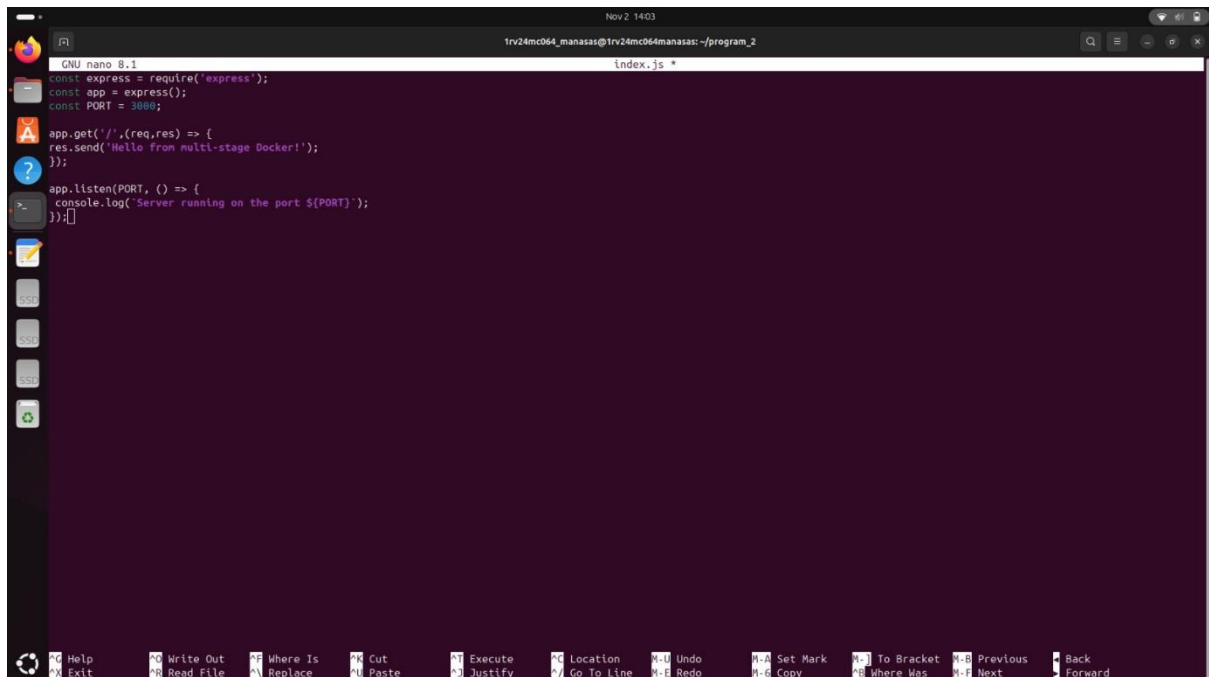
Next, they run 'npm install express'. The terminal shows an error: 'Unknown command: "install"'. It then suggests 'npm install # Install a package'. The user runs 'npm help' and then 'npm install'. The terminal shows the installation progress: 'up to date, audited 69 packages in 743ms' and '16 packages are looking for funding'. Finally, they run 'npm install express' again, which successfully installs the package.

Step 4: Create Source Folder and Application File

mkdir src

cd src

nano index.js



A terminal window showing the creation of the application file. The user is in a directory named 'src'. They run 'nano index.js' and create a file with the following content:

```
const express = require('express');
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
  res.send('Hello from multi-stage Docker!');
});

app.listen(PORT, () => {
  console.log(`Server running on the port ${PORT}`);
});
```

Step 5: Create the Multi-Stage Dockerfile

Go back to your main folder:

```
sudo nano Dockerfile
```



```
Nov 2 14:06
1rv24mc064_manasas@1rv24mc064manasas: ~/program_2

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
1rv24mc064_manasas@1rv24mc064manasas: ~/program_2$ npm install express
Unknown command: "install"

Did you mean this?
  npm install # Install a package

To see a list of supported npm commands, run:
  npm help
1rv24mc064_manasas@1rv24mc064manasas: ~/program_2$ npm install express

up to date, audited 69 packages in 829ms

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
1rv24mc064_manasas@1rv24mc064manasas: ~/program_2$ sudo docker build -t secondexe .
[code] password for 1rv24mc064_manasas:
[*] Building 5.8s (15/15) FINISHED

==> [internal] load build definition from Dockerfile                                0.0s
==> transfering dockerfile: 730B                                                    0.0s
==> [internal] load metadata for docker.io/library/node:20-alpine                 2.4s
==> [internal] load .dockerignore                                                    0.0s
==> transferring context: 2B                                                         0.0s
==> [builder 2/6] FROM docker.io/library/node:20-alpine@sha256:617be70b972f79c335df281f4b7674a2d05871aee2af020ffaf39f0a770265435 0.1s
==> [internal] load build context                                                  0.1s
==> transferring context: 100.97kB                                                 0.1s
==> CACHED [builder 2/6] WORKDIR /app                                              0.0s
==> [builder 3/6] COPY package.json package-lock.json ./                        0.0s
==> [builder 4/6] RUN npm install                                                  2.2s
==> [builder 5/6] COPY ...                                                         0.3s
==> [builder 6/6] RUN npm run build                                                0.4s
==> [stage-1 3/6] COPY --from=builder /app/package.json ./                      0.0s
==> [stage-1 4/6] COPY --from=builder /app/package-lock.json ./                 0.0s
==> [stage-1 5/6] COPY --from=builder /app/dist ./dist                          0.0s
==> [stage-1 6/6] COPY --from=builder /app/node_modules ./node_modules          0.1s
==> exporting to image                                                            0.1s
==> exporting layers                                                            0.1s
==> writing image sha256:48ab72744ed92cdf8820399aebd0ee08b626d661f5dc045c3227cfdb05e18 0.0s
==> naming to docker.io/library/secondexe                                         0.0s
1rv24mc064_manasas@1rv24mc064manasas: ~/program_2$
```

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

```
Nov 2 14:07
1rv24mc064_manasas@1rv24mc064manasas: ~/program_2

found 0 vulnerabilities
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$ npm install express
Unknown command: "install"

Did you mean this?
  npm install # Install a package

To see a list of supported npm commands, run:
  npm help
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$ npm install express

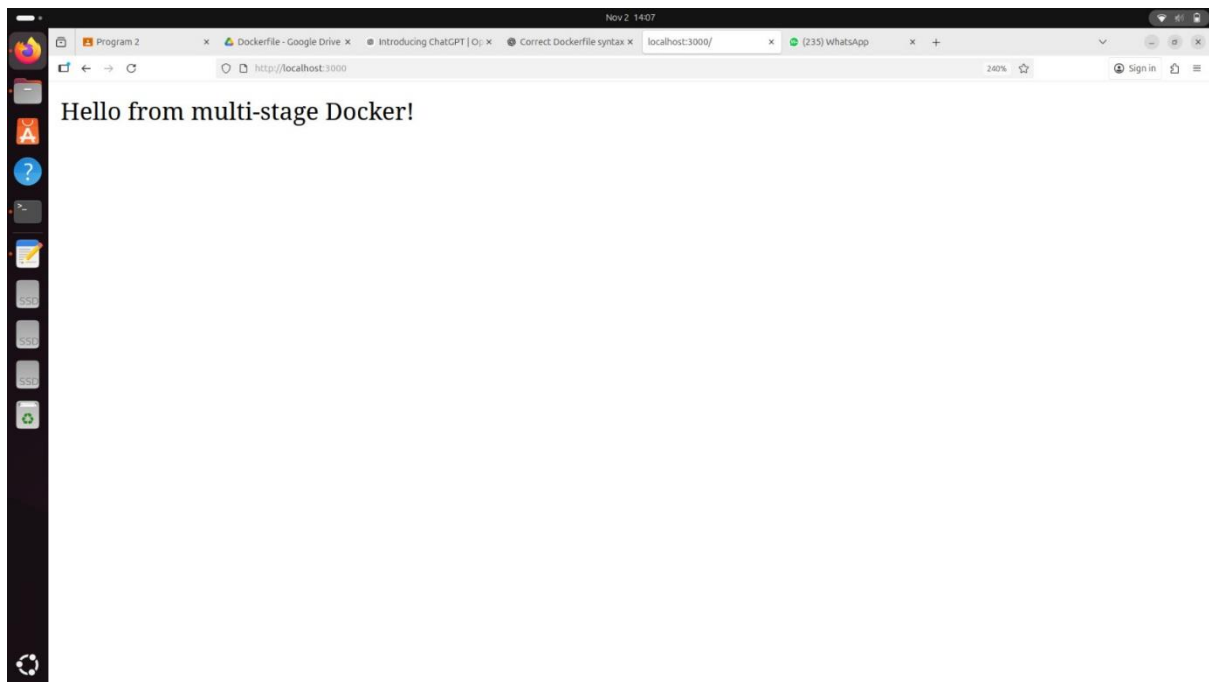
up to date, audited 69 packages in 829ms

16 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$ sudo docker build -t secondexe .
[sudo] password for 1rv24mc064_manasas:
[*] Building 5.8s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load metadata for docker.io/library/node:20-alpine
=> [internal] load .dockerignore
=> [internal] load context: 20
=> [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6170e78b72f79c335df201f4b7674a2d5871aacc1af920ffa39f6e779265435
=> [internal] load build context
=> [builder 2/6] WORKDIR /app
=> [builder 3/6] COPY package.json package-lock.json ./
=> [builder 4/6] RUN npm install
=> [builder 5/6] COPY .
=> [builder 6/6] RUN npm run build
=> [stage:1 3/6] COPY --from=builder /app/package.json ./
=> [stage:1 4/6] COPY --from=builder /app/package-lock.json ./
=> [stage:1 5/6] COPY --from=builder /app/dist ./dist
=> [stage:1 6/6] COPY --from=builder /app/node_modules ./node_modules
=> exporting layers
=> writing image sha256:40ab72744e026df88202990ebd6e0806826d61f55dc9451c327c7db05e18
=> naming to docker.io/library/secondexe
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$ sudo docker run -d -p 3000:3000 secondexe
8abe8d200a01601dad84263a059736b25c20973bdb51e0627f091969b8384320
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$
```

Step 8: Verify the Application

Open your browser and visit: <http://localhost:3000>



1. Check the Dockerfile Directly

grep -i "from" Dockerfile

```
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$ grep from Dockerfile
# Copy only necessary files from build stage
COPY --from=builder /app/package.json ./
COPY --from=builder /app/package-lock.json ./
COPY --from=builder /app/dist ./dist
COPY --from=builder /app/node_modules ./node_modules
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$
```

