

DevOps Lab

Program -2

Creating a Multi-Stage Dockerfile

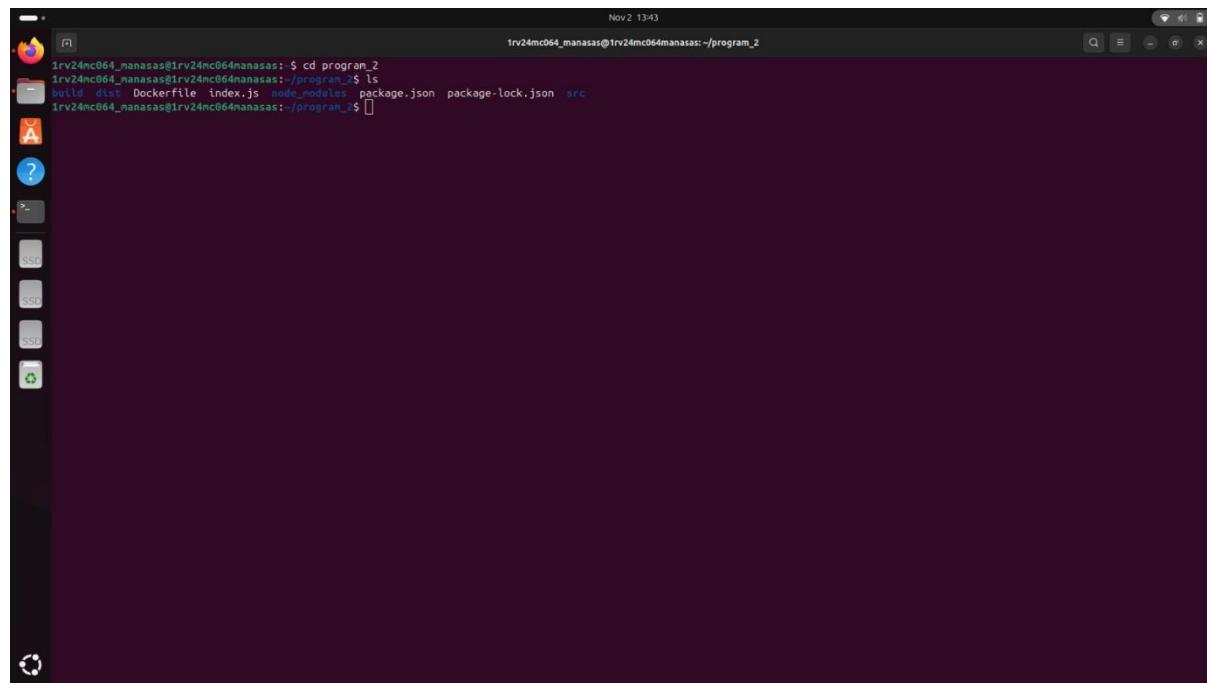
Project Structure

```
program_2/  
|  
|--- Dockerfile  
|--- package.json  
|--- package-lock.json  
|--- node_modules/  
└--- src/  
    └--- index.js
```

Step 1: Create Project Folder

```
mkdir program_2
```

```
cd program_2
```



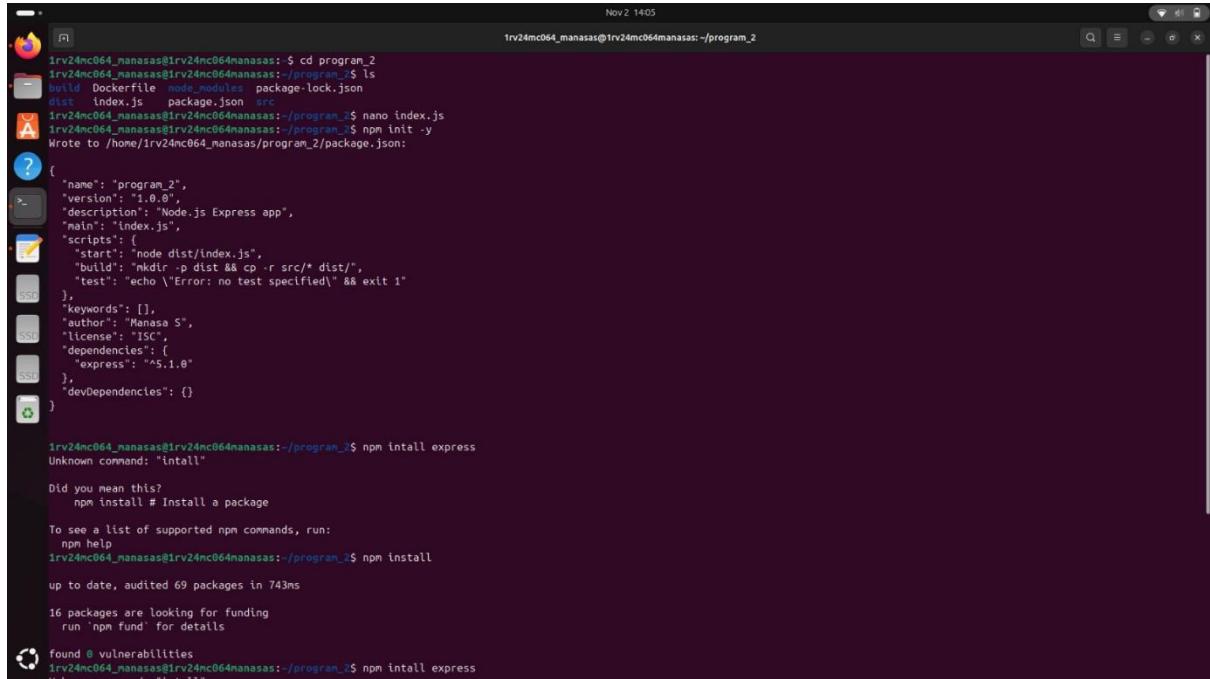
A screenshot of a terminal window titled "Nov 2 13:43". The command "cd program_2" is entered, followed by "ls" which lists the contents of the directory: Dockerfile, index.js, node_modules, package.json, package-lock.json, and src. The terminal has a dark theme with light-colored text.

Step 2: Initialize Node.js Project

```
npm init -y
```

Step 3: Install Express Framework

npm install express



```
Nov 2 14:05
irv24mc064_manasas@irv24mc064manasas:~/program_2
irv24mc064_manasas@irv24mc064manasas:~/program_2$ ls
build Dockerfile node_modules package-lock.json
dist index.js package.json src
irv24mc064_manasas@irv24mc064manasas:~/program_2$ nano index.js
irv24mc064_manasas@irv24mc064manasas:~/program_2$ npm init -y
Write to /home/irv24mc064_manasas/program_2/package.json:
{
  "name": "program_2",
  "version": "1.0.0",
  "description": "Node.js Express app",
  "main": "index.js",
  "scripts": {
    "start": "node dist/index.js",
    "build": "mkdf -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": {}
}

irv24mc064_manasas@irv24mc064manasas:~/program_2$ npm intall express
Unknown command: "intall"

Did you mean this?
  npm install # Install a package

To see a list of supported npm commands, run:
  npm help
irv24mc064_manasas@irv24mc064manasas:~/program_2$ npm install
up to date, audited 69 packages in 743ms

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

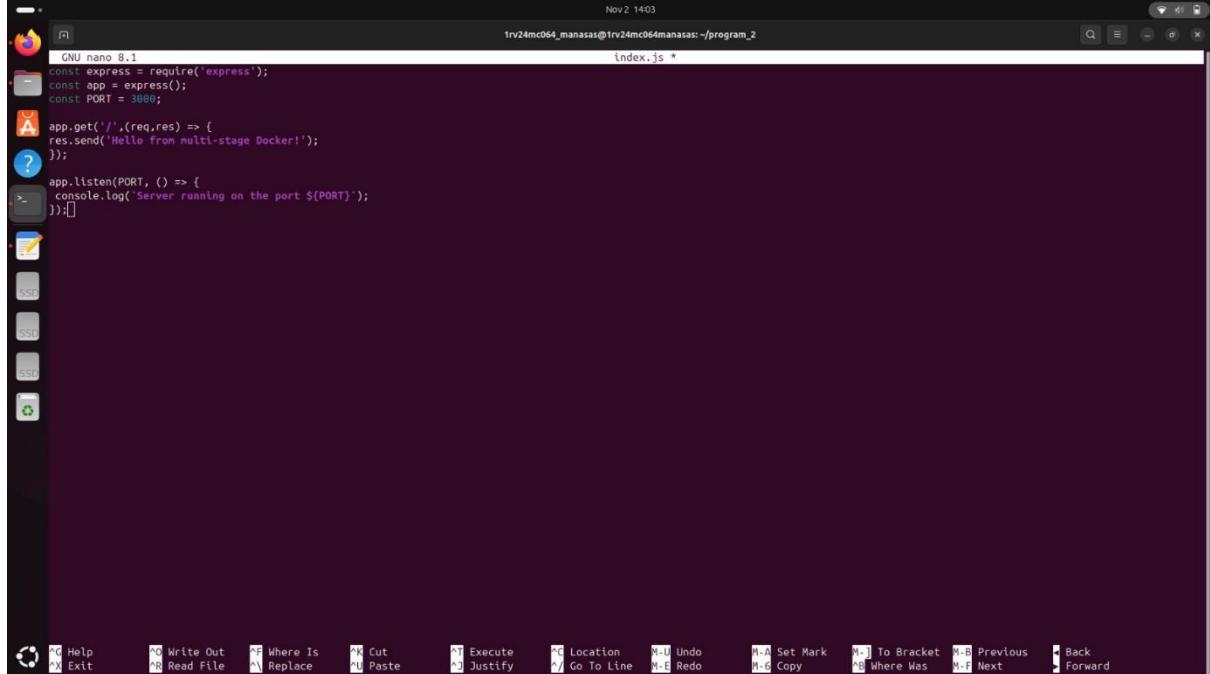
found 0 vulnerabilities
irv24mc064_manasas@irv24mc064manasas:~/program_2$ npm intall express
...  
...
```

Step 4: Create Source Folder and Application File

mkdir src

cd src

nano index.js



```
Nov 2 14:03
GNU nano 8.1
irv24mc064_manasas@irv24mc064manasas:~/program_2
index.js *

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}`);
});[]

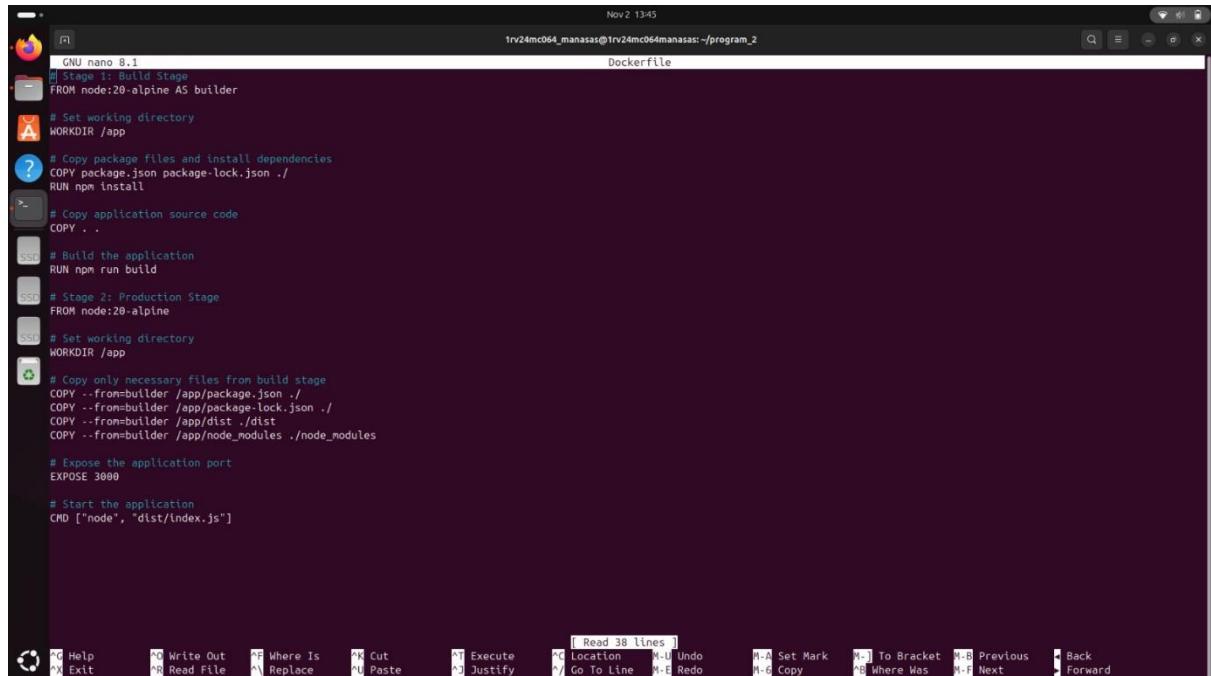
irv24mc064_manasas@irv24mc064manasas:~/program_2$
```

Step 5: Create the Multi-Stage Dockerfile

Go back to your main folder:

```
cd ..
```

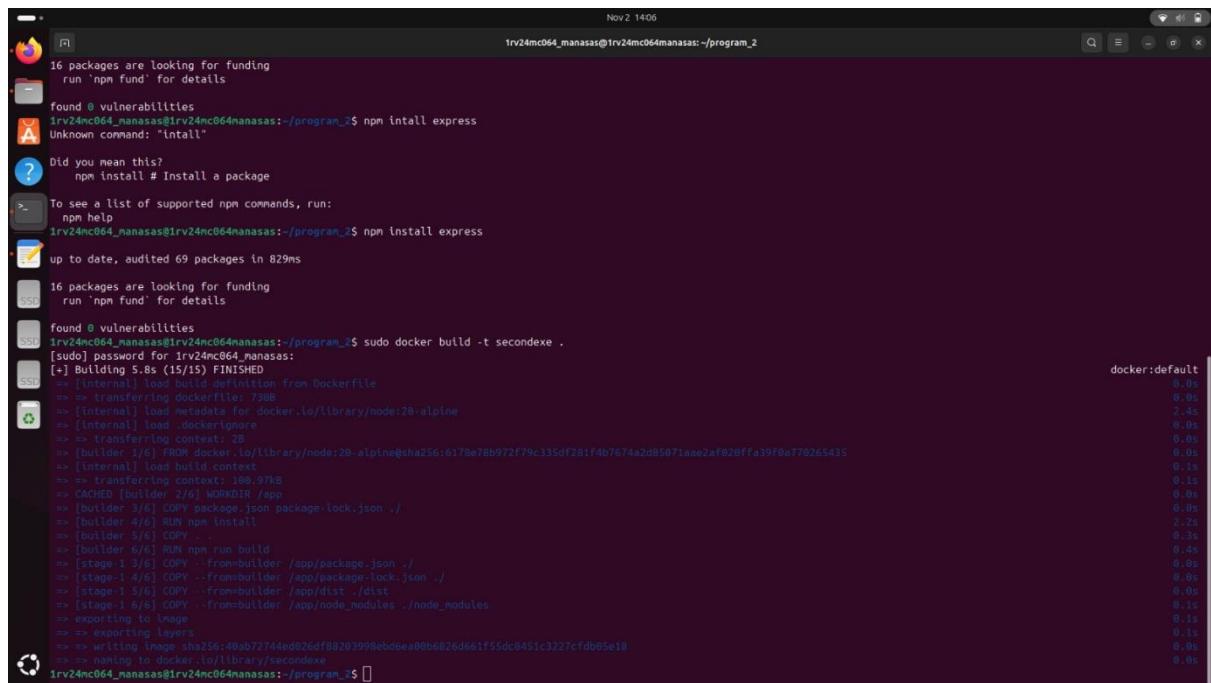
```
sudo nano Dockerfile
```



```
Nov 2 13:45
GNU nano 8.1
1rv24mc064_manasas@1rv24mc064manasas:~/program_2
FROM node:20-alpine AS builder
# Set working directory
WORKDIR /app
# Copy package files and install dependencies
COPY package.json package-lock.json .
RUN npm install
# Copy application source code
COPY .
# Build the application
RUN npm run build
# Stage 2: Production Stage
FROM node:20-alpine
# Set working directory
WORKDIR /app
# 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
# Expose the application port
EXPOSE 3000
# Start the application
CMD ["node", "dist/index.js"]
```

Step 6: Build the Docker Image

```
docker build -t secondexe .
```



```
Nov 2 14:06
16 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$ npm install express
Unknown command: "intall"
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] transfer dockerfile: 730B
   ==> [internal] load metadata for docker.io/library/node:20-alpine
   ==> [internal] load .dockerignore
   ==> [internal] transfer context: 2B
   ==> [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e8b97f79c335df281f4b7674a2d85071aee2af020ffa39f0a770265435
   ==> [internal] load build context
   ==> [internal] transfer context: 109.97kB
   ==> [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 to image
   ==> exporting layers
   ==> writing image sha256:40ab72744ed0260ff38203998ebd6ea00b6626d661f55dc8451c3227cfdb005e18
   ==> saving to docker.io/library/secondexe
1rv24mc064_manasas@1rv24mc064manasas:~/program_2$
```

Step 7: Run the Docker Container

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

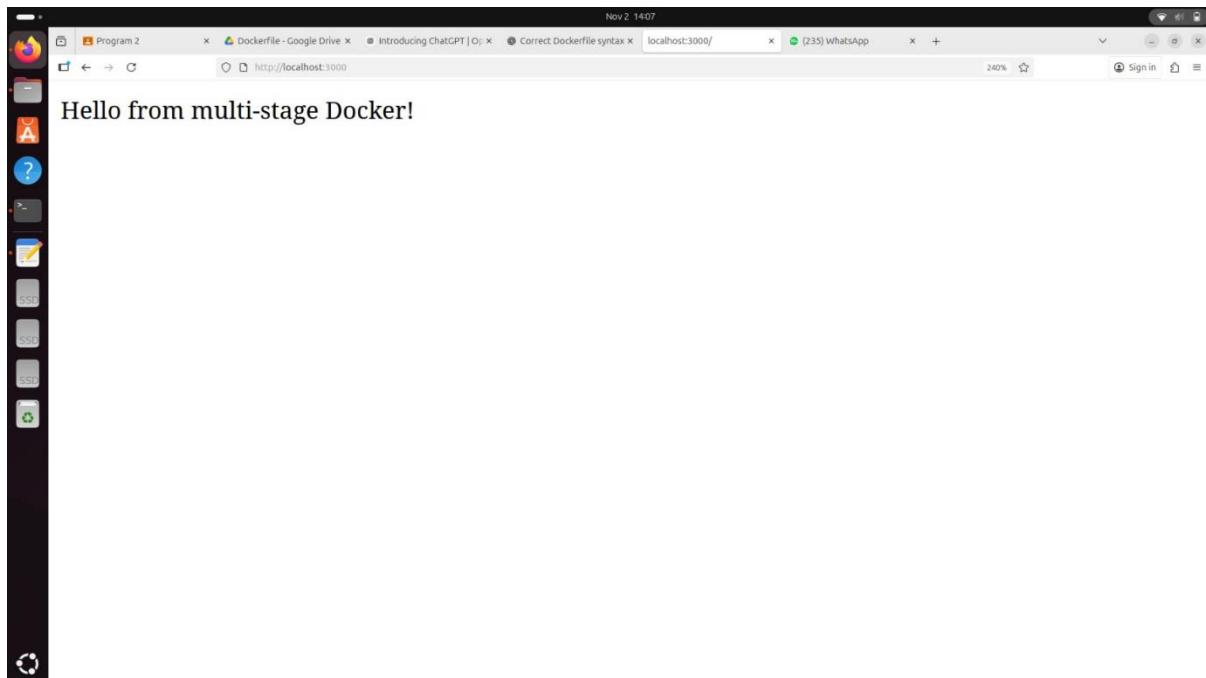
```

Nov 2 14:07
irv24mc064_manasas@irv24mc064manasas:~/program_2$ npm install express
Unknown command: "intall"
Did you mean this?
  npm install # Install a package
To see a list of supported npm commands, run:
  npm help
irv24mc064_manasas@irv24mc064manasas:~/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
irv24mc064_manasas@irv24mc064manasas:~/program_2$ sudo docker build -t secondeixe .
[sudo] password for irv24mc064_manasas:
[+] Building 5.8s (15/15) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 73B
  => [internal] load metadata for docker.io/library/node:20-alpine
  => [internal] load .dockerignore
  => => transferring context: 2B
  => [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:617de78b97f79c335df281f4b7674a2d85071aee2af020ffa39f0a779265435
  => [internal] load build context
  => => transferring context: 100.97KB
  => CACHED [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 to image
  => => exporting layers
  => => writing image sha256:40ab72744ed026df08203998eb06ea0066826d661f55dc8451c3227cfdb085e18
  => => saving to disk /var/lib/docker/overlay2/secondeixe
  => => sending local commit
  => => writing manifest
  => => writing config
  => => writing layer sha256:babe8d280a01681dad84263a859736b25c20973bd51e0627f091969b8384320
irv24mc064_manasas@irv24mc064manasas:~/program_2$ 

```

Step 8: Verify the Application

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



1. Check the Dockerfile Directly

grep -i "from" Dockerfile

```

irv24mc064_manasas@irv24mc064manasas:~/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
irv24mc064_manasas@irv24mc064manasas:~/program_2$ 

```

