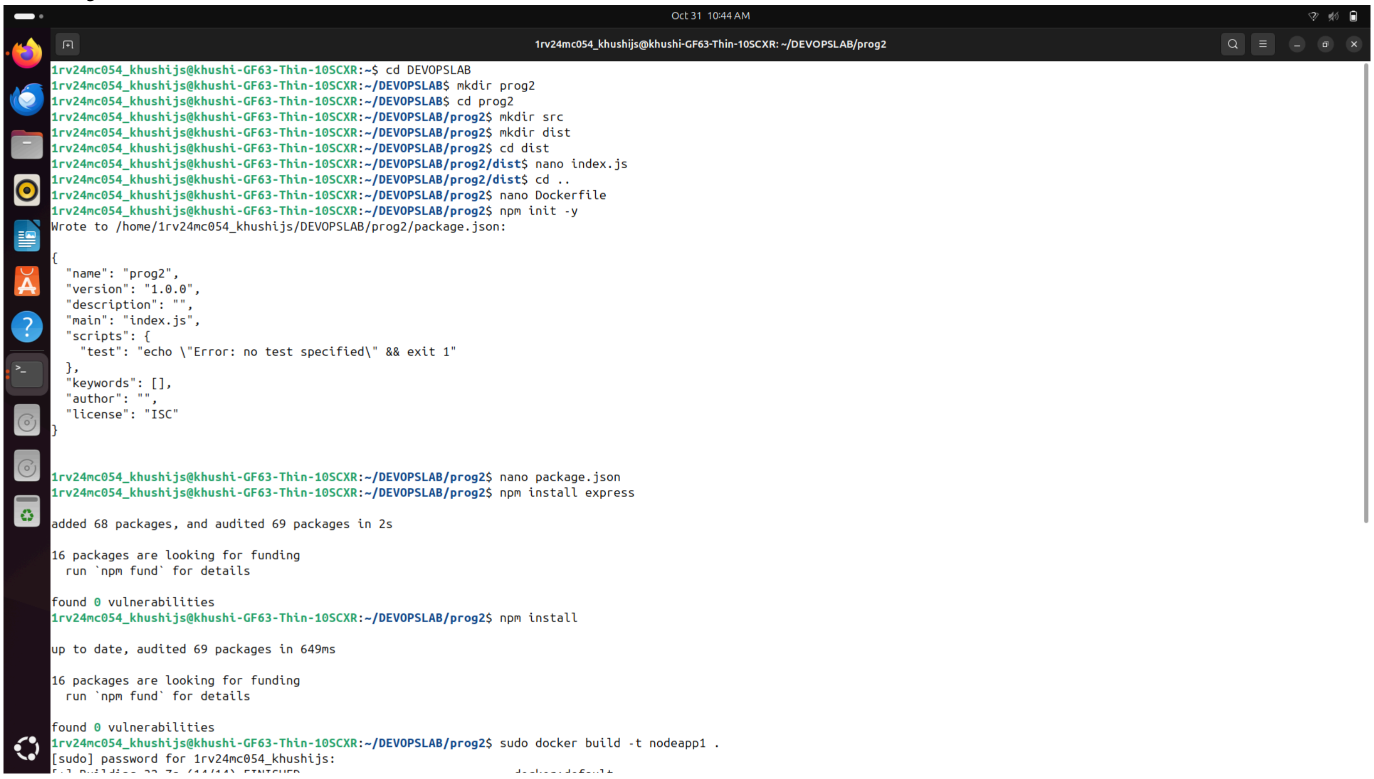
Program 2: Develop a Multi-Stage Dockerfile for Container Orchestration.

Folder Structure

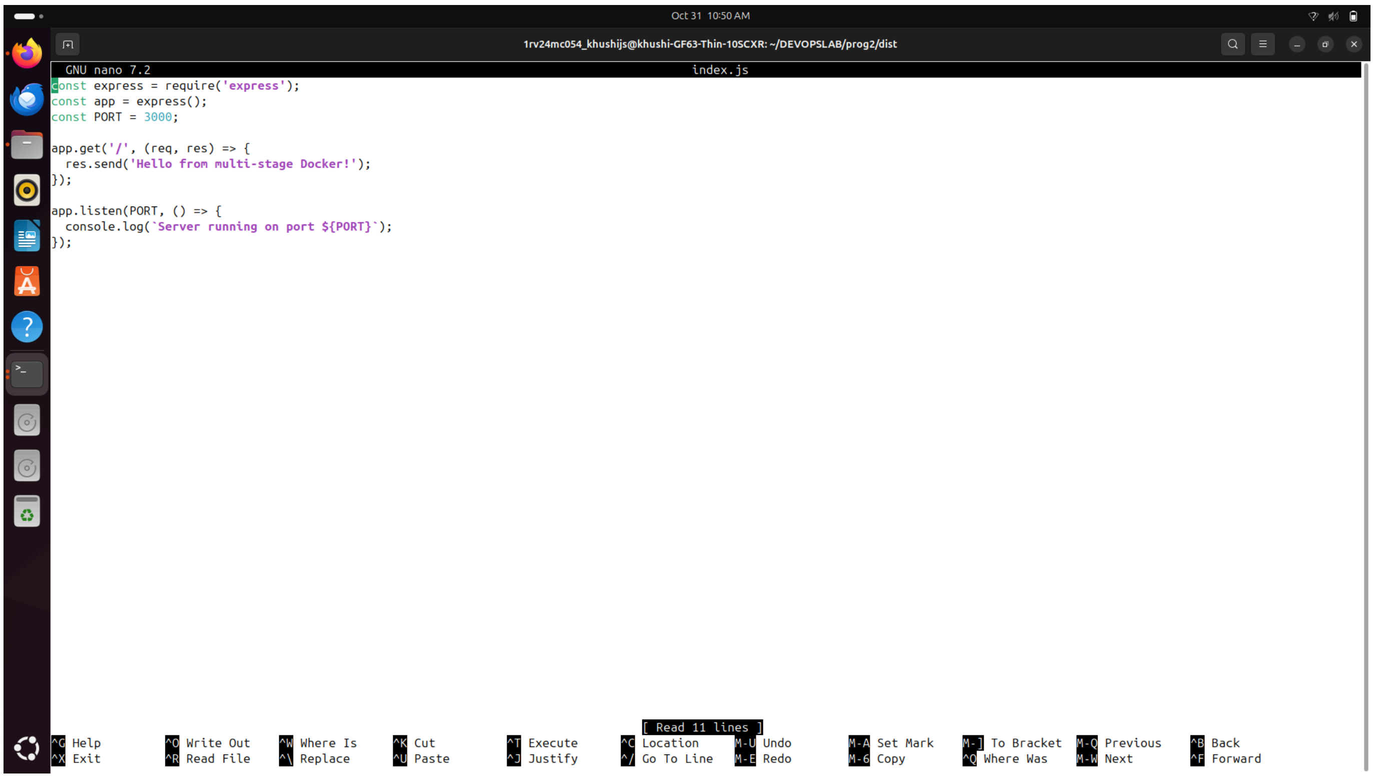
```
DEVOPSLAB
        |--Program2
              |--src
              |--dist
                  |--index.js
              |--Dockerfile
              |--build
              |--package.json
              |--package-lock.json
              |--node_modules
```

Step 1: I initiated a new Node.js project by creating a folder named "Program2." Within this folder, I established three subfolders: build, dist, and src, for organized code management. Inside the dist folder, I created index.js to house the primary server logic and Dockerfile to manage dependencies. Subsequently, I returned to the main project directory and ran npm init -y to generate a default package.json file, which tracks project details and dependencies. To facilitate web server development, I then installed Express, a widely used Node.js framework.


Oct 31 10:44 AM
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR: ~/DEVOPSLAB/prog2

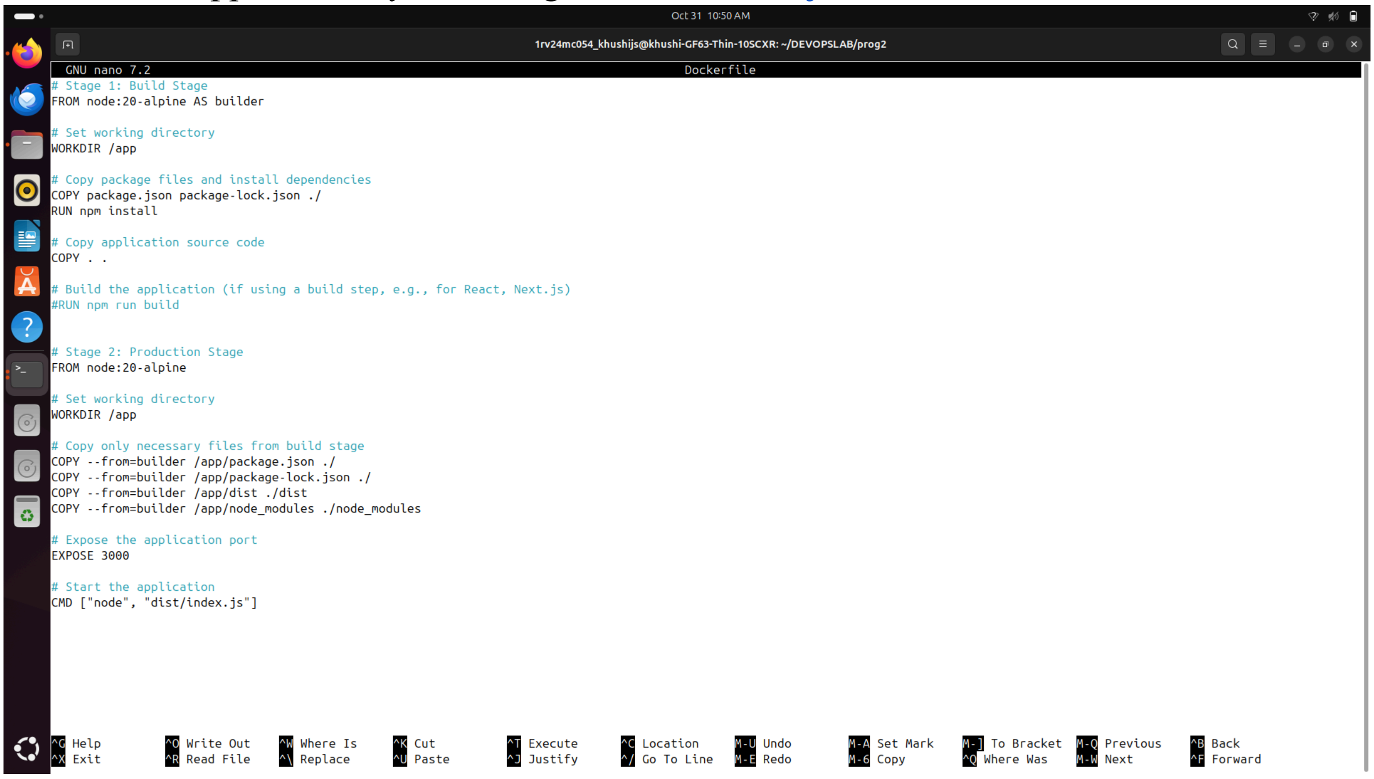
```
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~$ cd DEVOPSLAB
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB$ mkdir prog2
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB$ cd prog2
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ mkdir src
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ mkdir dist
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ cd dist
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2/dist$ nano index.js
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2/dist$ cd ..
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ nano Dockerfile
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ npm init -y
Wrote to /home/1rv24mc054_khushijs/DEVOPSLAB/prog2/package.json:

{
  "name": "prog2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}


1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ nano package.json
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ npm install express

added 68 packages, and audited 69 packages in 2s

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

found 0 vulnerabilities
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ npm install

up to date, audited 69 packages in 649ms

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

found 0 vulnerabilities
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ sudo docker build -t nodeapp1 .
[sudo] password for 1rv24mc054_khushijs:
```

Step 2: This Express-based code sets up a basic web server. It operates on port 8000, accessible via http://localhost:3000.



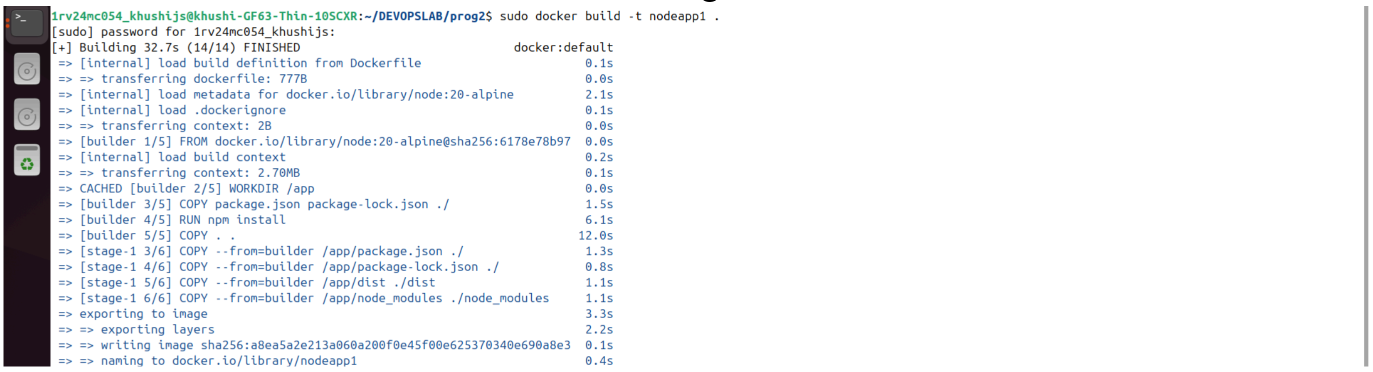
```
GNU nano 7.2                         index.js
const express = require('express');
const app = express();
const PORT = 3000;

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

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

Step 3: This Dockerfile constructs a compact Node.js container for our application. It utilizes the Node 20 Alpine image as its foundation, designates /app as the working directory, and installs dependencies as specified in package.json. Subsequently, it copies the remaining application files, exposes port 3000 for external access, and ultimately launches the application by executing node dist/index.js.



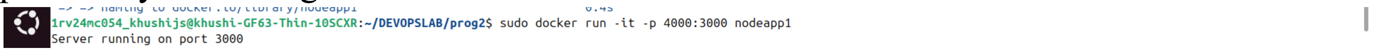
```
GNU nano 7.2                         Dockerfile
# Stage 1: Build Stage
FROM node:20-alpine AS builder

# 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 (if using a build step, e.g., for React, Next.js)
#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 4: Next, build the Docker container using the docker build command. This command instructs Docker to create a new image from the Dockerfile.

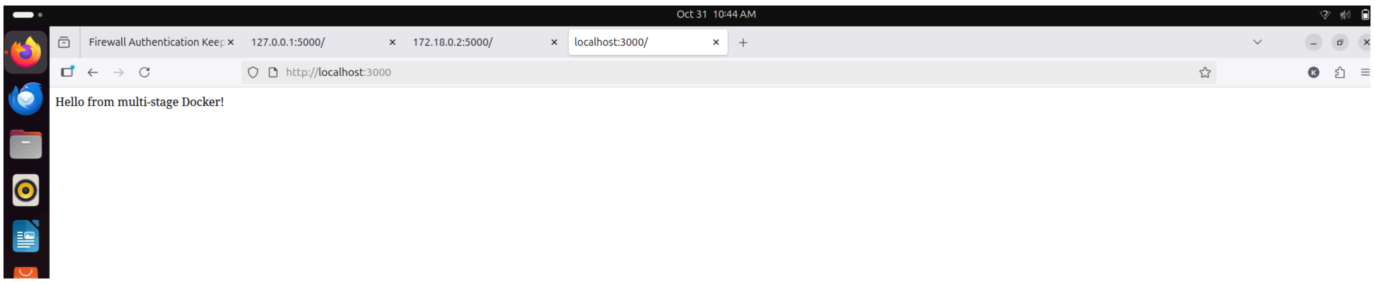


```
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ sudo docker build -t nodeapp1 .
[sudo] password for 1rv24mc054_khushijs:
[+] Building 32.7s (14/14) FINISHED                                 docker:default
 => [internal] load build definition from Dockerfile                           0.1s
 => => transferring dockerfile: 777B                                           0.0s
 => [internal] load metadata for docker.io/library/node:20-alpine              2.1s
 => [internal] load .dockerignore                                              0.1s
 => => transferring context: 2B                                                0.0s
 => [builder 1/5] FROM docker.io/library/node:20-alpine@sha256:6178e78b97      0.0s
 => [internal] load build context                                              0.2s
 => => transferring context: 2.70MB                                            0.1s
 => CACHED [builder 2/5] WORKDIR /app                                          0.0s
 => [builder 3/5] COPY package.json package-lock.json ./                       1.5s
 => [builder 4/5] RUN npm install                                              6.1s
 => [builder 5/5] COPY . .                                                    12.0s
 => [stage-1 3/6] COPY --from=builder /app/package.json ./                     1.3s
 => [stage-1 4/6] COPY --from=builder /app/package-lock.json ./                0.8s
 => [stage-1 5/6] COPY --from=builder /app/dist ./dist                         1.1s
 => [stage-1 6/6] COPY --from=builder /app/node_modules ./node_modules         1.1s
 => exporting to image                                                         3.3s
 => => exporting layers                                                        2.2s
 => => writing image sha256:a8ea5a2e213a060a200f0e45f00e625370340e690a8e3      0.1s
 => => naming to docker.io/library/nodeapp1                                    0.4s
```

Step 5: To create a container, execute the docker run command, which utilizes the previously built image.



```
1rv24mc054_khushijs@khushi-GF63-Thin-10SCXR:~/DEVOPSLAB/prog2$ sudo docker run -it -p 4000:3000 nodeapp1
Server running on port 3000
```



Hello from multi-stage Docker!