

Program - 2

Develop a Multi-Stage Dockerfile for Container Orchestration

Description

The objective of this problem is to develop a multi stage dockerfile for container orchestration. This approach helps optimize image size, improve security, and streamline the build and deployment process.

Project Structure

```
program2/
    → Dockerfile
    → package.json
    → package-lock.json
    → src/ index.js
    → node_modules
```

Step 1: Create the Dockerfile

```
jeevanc@jeevanc-IdeaPad-Slim-3-15AMN8:~/Devopslab/program2$ cat 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
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 2: Make a folder -> src and create a file -> index.js

```
jeevanc@jeevanc-IdeaPad-Slim-3-15AMN8:~/Devopslab/program2$ cat src/index.js
const express = require('express');

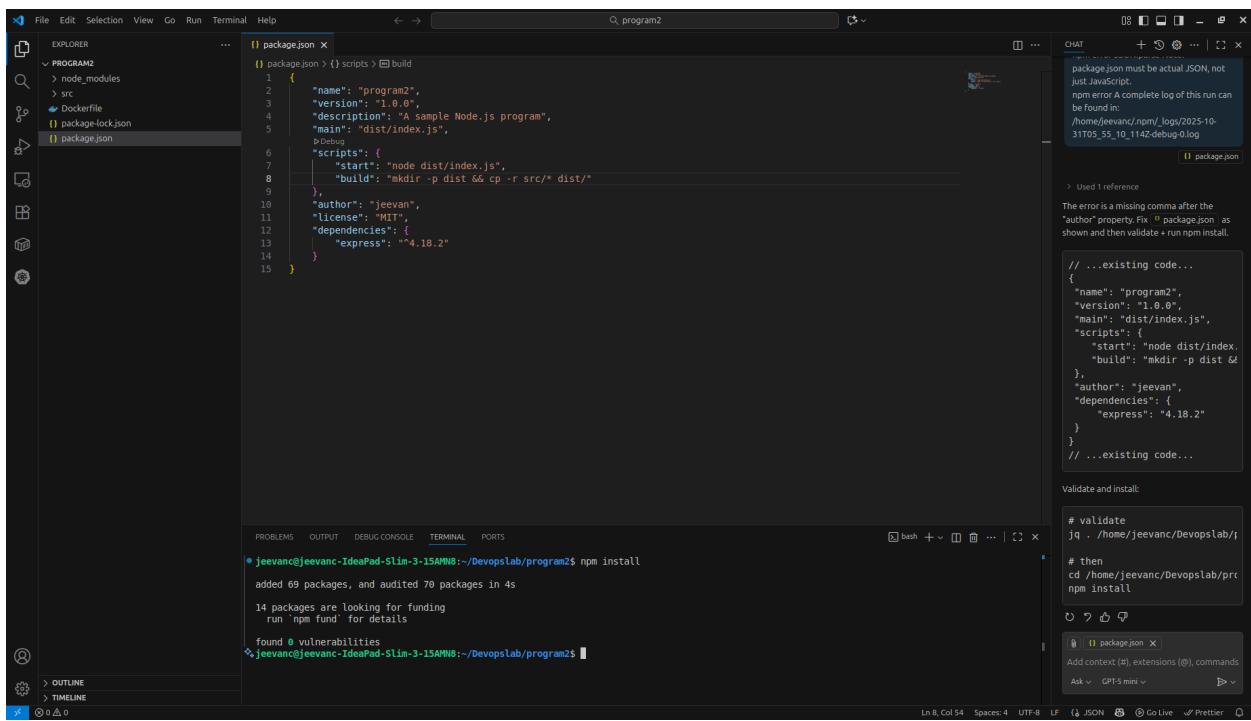
const app = express();

const PORT = 3000;

app.get('/', (req,res) => {
    res.send("Hello");
});

app.listen(PORT, () =>{
    console.log("Server is running");
});
```

Step 3: Create a file -> package.json



Step 4: Install Node modules with the following command

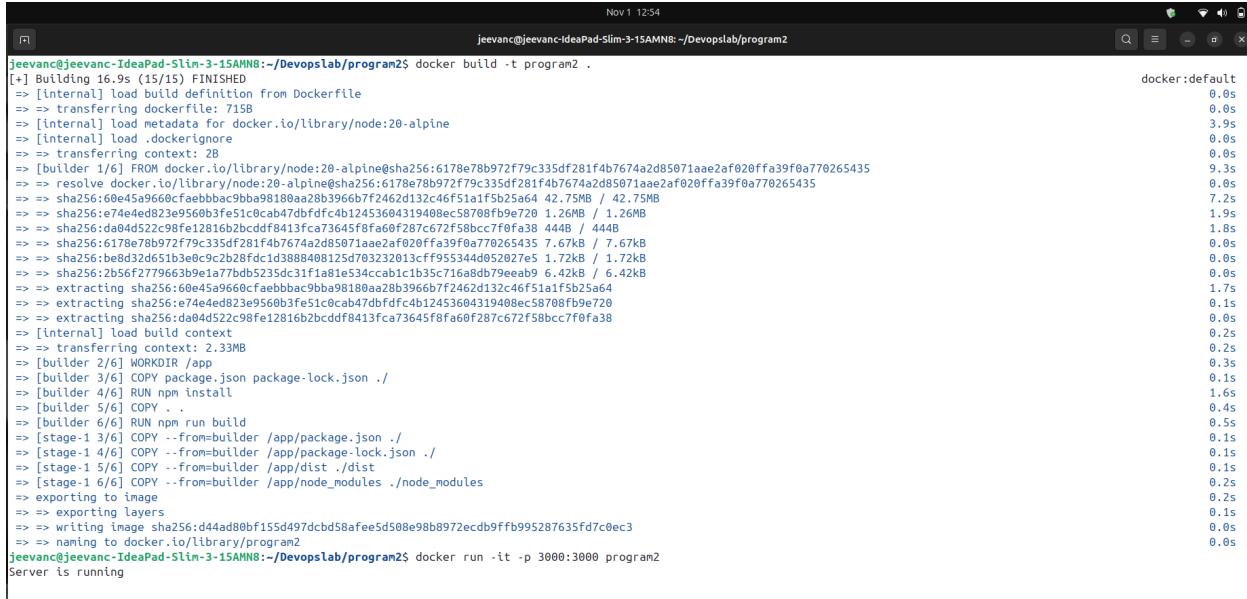
```
npm install
```

Step 5: Execute the Docker Build Command

```
docker build -t program2
```

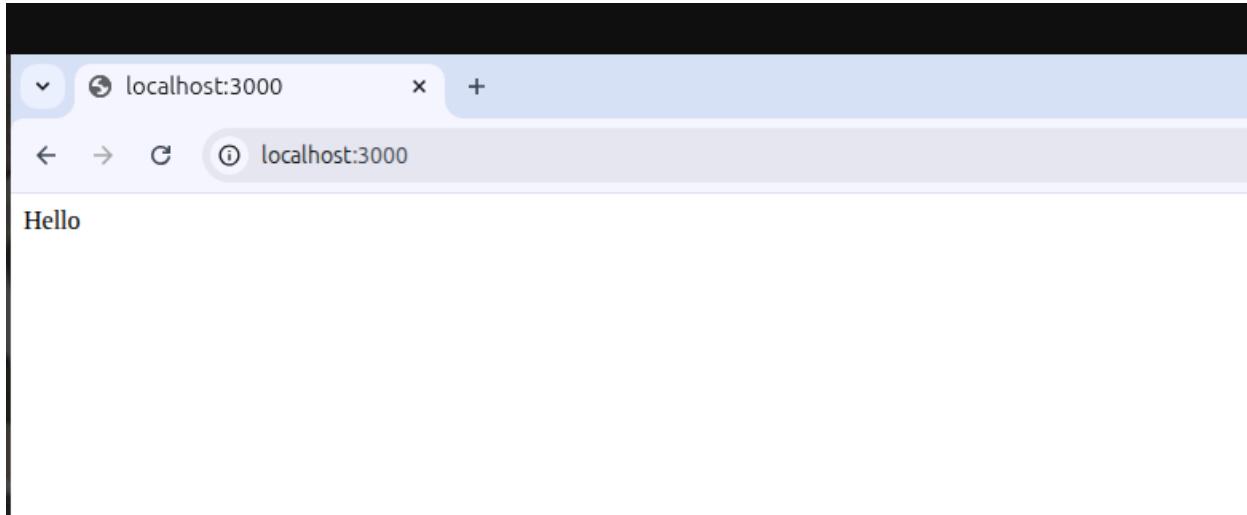
Step 6: Run the Docker Run command specifying the port number

```
docker run -it -p 3000:3000 program2
```



```
jeevanc@jeevanc-IdeaPad-Slim-3-15AMN8:~/Devopslab/program2$ docker build -t program2 .
[j+] Building 16.9s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 71B
=> [internal] load metadata for docker.io/library/node:20-alpine
=> [internal] load .dockignore
=> => transferring context: 2B
=> [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af020ffa39f0a770265435
=> => resolve docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af020ffa39f0a770265435
=> => sha256:60e45a9660cfaeabbac9ba98180aa2bb3966b7f72462d132c46f51a1fsb25a64 42.75MB / 42.75MB
=> => sha256:e744e0d823e9560b3fe51c0cab47dbdfc4b12453604319408ec5870bf9e720 1.26MB / 1.26MB
=> => sha256:da94d522c98fe12816b2bcdff8413fc73645f8fa0f287c672f58bbc7ff0fa38 444B / 444B
=> => sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af020ffa39f0a770265435 7.67kB / 7.67kB
=> => sha256:be8d32d651b3e0c9cb28fd1c1d3888408125d7032328013cff95534d95852027e5 1.72kB / 1.72kB
=> => sha256:2b56f2779663b9e1a77bdb5235dc31f1a81e534ccab1c1b35c716a8db79eeab9 6.42kB / 6.42kB
=> => extracting sha256:60e45a9660cfaeabbac9ba98180aa2bb3966b7f72462d132c46f51a1fsb25a64
=> => extracting sha256:e744e0d823e9560b3fe51c0cab47dbdfc4b12453604319408ec5870bf9e720
=> => extracting sha256:da94d522c98fe12816b2bcdff8413fc73645f8fa0f287c672f58bcc7ff0fa38
=> [internal] load build context
=> => transferring context: 2.33MB
=> [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:d44ad80bf155d497dcdb58afee5d508e98b8972ecdb9ffb995287635fd7c0ec3
=> => naming to docker.io/library/program2
jeevanc@jeevanc-IdeaPad-Slim-3-15AMN8:~/Devopslab/program2$ docker run -it -p 3000:3000 program2
Server is running
```

Step 7: Test the container by verifying the localhost details on web server



Step 8: Stop the Docker container, remove the container

```
docker container ls -a
```

```
docker container stop <container-number>
```

```
docker rm <container-number>
```

```
docker rmi <image-id>
```