**NAME :ANUSHA S**

**USN:1RV24MC019**

**PROGRAM -2**

**TITLE :**Develop a multi-stage Dockerfile for container orchestration

Structure of the Program:

Program -2:

Dockerfile

src/index.js

package.json

package-lock.json

node_modules

Step-1 : Create a Dockerfile

```
### Start building the multistage Docker file -1 and 2
#Build and Production

#Step 1:Building stage
FROM node:20-alpine AS builder

#Set the working directory in the container
WORKDIR /app

#Copy the files and install the dependencies
COPY package.json package-lock.json ./
RUN npm install

#Copy the application code
COPY . .

#Build the application
RUN npm run build


#Stage 2-Production Stage
FROM node:20-alpine

#Set the working directory
WORKDIR /app

#Copy the necessary files from the builder
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 port on which the app runs
EXPOSE 3000

#Define the command on which the app runs
CMD ["node","dist/index.js"]
```

Step-2: You can write package.json or create it using npm init -y

```
{
  "name": "program-2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start":"node dist/index.js",
    "build":"mkdir -p dist && cp -r src/* dist/"
  },
  "keywords": [],
  "author": "Anusha",
  "license": "ISC",
  "dependencies": {
    "express": "^5.1.0"
  }
}
```

Step-3: Create a src folder and write index.js file inside it

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

app.get("/",(req,res)=>{
res.send("Hello!From the docker");
});

app.listen(PORT,()=>{
console.log(`Server runs on http://localhost:${PORT}`);
});
```

Step-4:Inside the terminal create Docker image

sudo docker build -t express-image .

Step-5: Inside the terminal run the container

sudo docker run -d -p 3000:3000 –name second-container program-2

Step-6 : Check in your browser and go to http://localhost:3000


# Hello!From the docker

Step-7:Verify the containers in the terminal

sudo docker container ls -a

sudo docker container stop second-container

sudo docker container rm second-container

sudo docker image rm express-image

```
_1RV24MC019_anusha_s@anu-Vivobook-Go-E1504FA-E1504FAB:~/Program-2$ sudo docker container ls -a
[sudo] password for _1RV24MC019_anusha_s:
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                                       NAMES
26e892e5d9de   express-image   "docker-entrypoint.s…"   46 minutes ago   Up 46 minutes   0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp   second-container
_1RV24MC019_anusha_s@anu-Vivobook-Go-E1504FA-E1504FAB:~/Program-2$ sudo docker container stop second-container
second-container
_1RV24MC019_anusha_s@anu-Vivobook-Go-E1504FA-E1504FAB:~/Program-2$ sudo docker container rm second-container
second-container
_1RV24MC019_anusha_s@anu-Vivobook-Go-E1504FA-E1504FAB:~/Program-2$ sudo docker image rm express-image
Untagged: express-image:latest
Deleted: sha256:35052908aae04b424c14f46cdd059fed0a5eb67363b5c7a68e7b4ab9c747bf9b
_1RV24MC019_anusha_s@anu-Vivobook-Go-E1504FA-E1504FAB:~/Program-2$
```