

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

```
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop$ mkdir programm2
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop$ cd programm2
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ npm init -y

1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ mkdir src
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ nano src/index.js
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ ls
node_modules package.json package-lock.json src
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ nano package.json
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ npm run build
```

```
GNU nano 6.2 Dockerfile
#multistage docker 182
#stage 1 build
FROM node:20-alpine AS builder

WORKDIR /app

#copy package and install
COPY package.json package-lock.json ./
RUN npm install

#copy
COPY . .
#BUILD
RUN npm run build
#stage 2
FROM node:20-alpine

WORKDIR /app
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 3000

CMD ["node","dist/index.js"]
```

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

```
GNU nano 6.2 package.json
{
  "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": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

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 multi-stage Docker!');
});

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

Step-4: install node dependencies

```
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ npm run build
```

```
> program_2@1.0.0 build
> mkdir -p dist && cp -r src/* dist/
```

```
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ ls
dist  node_modules  package.json  package-lock.json  src
```

Step-5: Create a docker image

➤ Sudo docker build -t prg-2 .

```
1RV24MC036_gayatri_v_k@ubuntu:~/Desktop/programm2$ sudo docker build -t node-multistage-app .
[sudo] password for 1RV24MC036_gayatri_v_k:
[+] Building 2.7s (14/14) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 406B                             0.0s
=> [internal] load metadata for docker.io/library/node:20-alpine 2.1s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.4s
=> => transferring context: 2.70MB                               0.4s
=> [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e78b97 0.0s
=> CACHED [builder 2/6] WORKDIR /app                             0.0s
=> CACHED [builder 3/6] COPY package*.json ./                   0.0s
=> CACHED [builder 4/6] RUN npm install                          0.0s
=> CACHED [builder 5/6] COPY . .                                  0.0s
=> CACHED [builder 6/6] RUN npm run build                        0.0s
=> CACHED [stage-1 3/5] COPY --from=builder /app/package*.json ./ 0.0s
=> CACHED [stage-1 4/5] COPY --from=builder /app/node_modules ./node_mod 0.0s
=> CACHED [stage-1 5/5] COPY --from=builder /app/dist ./dist    0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:b80d8b1b8d22adbe8deaa8305497cc96e125d4bf38c75 0.0s
=> => naming to docker.io/library/node-multistage-app           0.0s
```

Step-6: Run the container

- Sudo docker run -it -p 3000:3000 prg-2

Step-7: go to browser and type <http://localhost:3000>

