2. Develop a Multi-stage Dockerfile for container Orchestration

Directory Structure:

```
P2
|---- Dockerfile
|---- package.json
|---- / src/ index.js
|---- / dist / index.js
|---- node_modules
|---- / build
```

Step-1 : Create a Dockerfile using nano in the pwd

→ **Dockerfile**

```
// Stage-1
FROM node:20-alpine AS builder
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm install
COPY . .
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 5000
CMD ["node", "dist/index.js"]
```

Step-2: Create a express script inside src folder

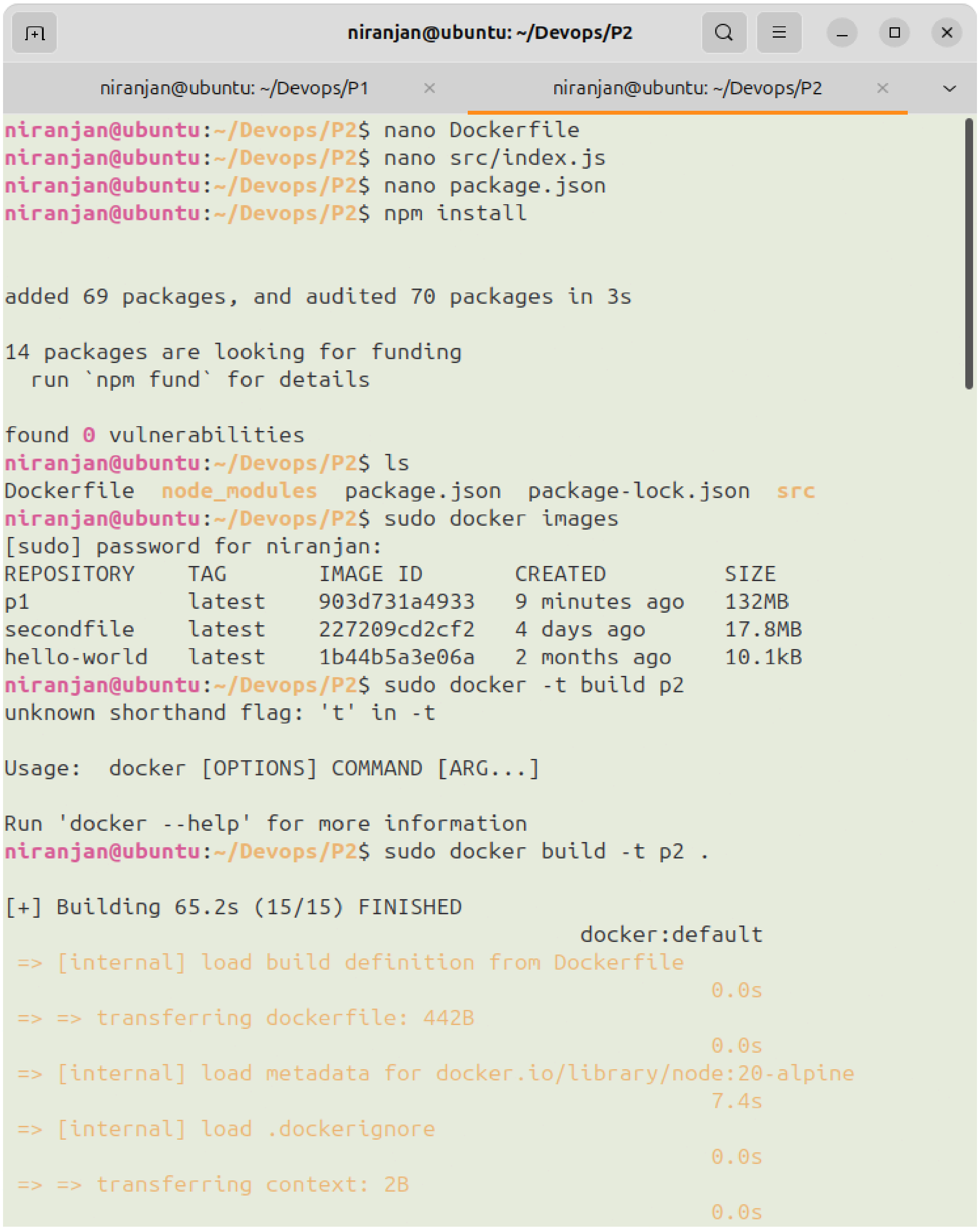
→ / src / index.js

```
const express= require('express');
const app = express();
const PORT = 5000;

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

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

Step-3: Create a custom package.json file or you can also run **npm init -y** it'll generate package.json for you and you can modify the content according to you.

→package.json

```
{
  "name": "p2",
  "version": "1.0.0",
  "description": "A Node js app with multi-staged Dockerfile",
  "main": "dist/index.js",
  "scripts": {
    "start": "node dist/index.js",
    "build": "mkdir -p dist && cp -r src/* dist/"
},
  "keywords": [],
  "author": "Niranjan",
  "license": "No-Licence-yet",
  "dependencies":{
  "express":"^4.18.2"
  }
}
```



niranjan@ubuntu: ~/Devops/P2

niranjan@ubuntu: ~/Devops/P1 | niranjan@ubuntu: ~/Devops/P2

```
niranjan@ubuntu:~/Devops/P2$ nano Dockerfile
niranjan@ubuntu:~/Devops/P2$ nano src/index.js
niranjan@ubuntu:~/Devops/P2$ nano package.json
niranjan@ubuntu:~/Devops/P2$ npm install


added 69 packages, and audited 70 packages in 3s

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

found 0 vulnerabilities
niranjan@ubuntu:~/Devops/P2$ ls
Dockerfile  node_modules  package.json  package-lock.json  src
niranjan@ubuntu:~/Devops/P2$ sudo docker images
[sudo] password for niranjan:
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
p1            latest    903d731a4933   9 minutes ago    132MB
secondfile    latest    227209cd2cf2   4 days ago       17.8MB
hello-world   latest    1b44b5a3e06a   2 months ago     10.1kB
niranjan@ubuntu:~/Devops/P2$ sudo docker -t build p2
unknown shorthand flag: 't' in -t

Usage:  docker [OPTIONS] COMMAND [ARG...]

Run 'docker --help' for more information
niranjan@ubuntu:~/Devops/P2$ sudo docker build -t p2 .

[+] Building 65.2s (15/15) FINISHED
                                            docker:default
 => [internal] load build definition from Dockerfile
                                                      0.0s
 => => transferring dockerfile: 442B
                                                      0.0s
 => [internal] load metadata for docker.io/library/node:20-alpine
                                                      7.4s
 => [internal] load .dockerignore
                                                      0.0s
 => => transferring context: 2B
                                                      0.0s
```

Step-4 : install Node modules using below command

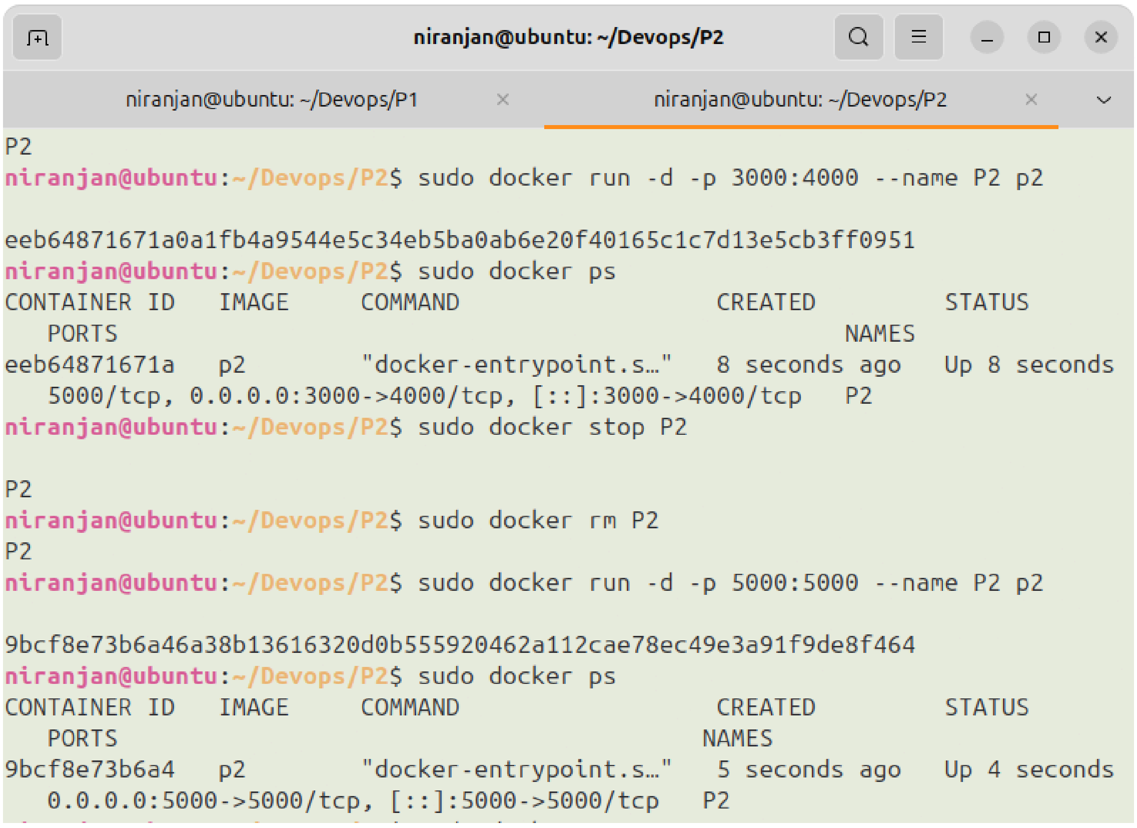
npm install

Step-5 : Build a Docker image using

Sudo docker build -t program2 .

Step-6 : Run the image to build a container

sudo docker run -t -p 3000:3000 program2

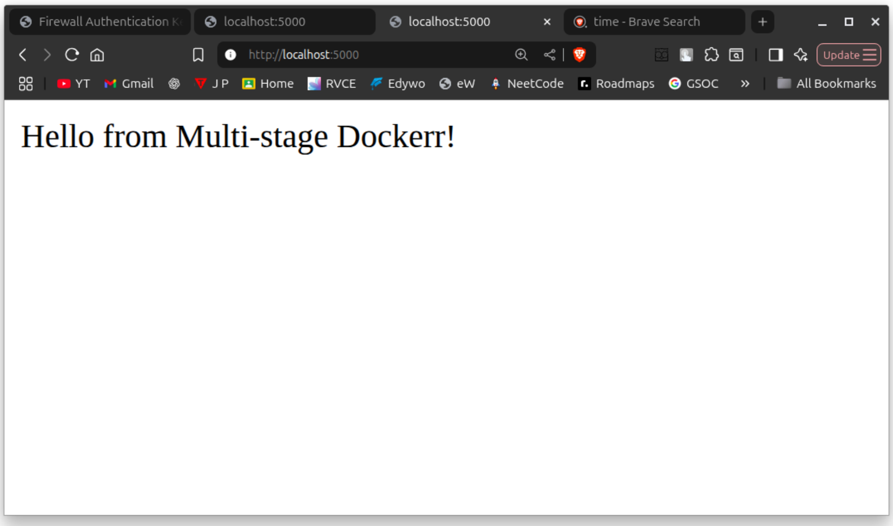

niranjan@ubuntu: ~/Devops/P2
niranjan@ubuntu: ~/Devops/P1
niranjan@ubuntu: ~/Devops/P2

```
P2
niranjan@ubuntu:~/Devops/P2$ sudo docker run -d -p 3000:4000 --name P2 p2

eeb64871671a0a1fb4a9544e5c34eb5ba0ab6e20f40165c1c7d13e5cb3ff0951
niranjan@ubuntu:~/Devops/P2$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
   PORTS                                                 NAMES
eeb64871671a   p2        "docker-entrypoint.s…"   8 seconds ago   Up 8 seconds
   5000/tcp, 0.0.0.0:3000->4000/tcp, [::]:3000->4000/tcp   P2
niranjan@ubuntu:~/Devops/P2$ sudo docker stop P2

P2
niranjan@ubuntu:~/Devops/P2$ sudo docker rm P2
P2
niranjan@ubuntu:~/Devops/P2$ sudo docker run -d -p 5000:5000 --name P2 p2

9bcf8e73b6a46a38b13616320d0b555920462a112cae78ec49e3a91f9de8f464
niranjan@ubuntu:~/Devops/P2$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
   PORTS                                      NAMES
9bcf8e73b6a4   p2        "docker-entrypoint.s…"   5 seconds ago   Up 4 seconds
   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   P2
```

Step-7 : check the status of the container using

Sudo docker ps

If it’s UP , now you verify the output of the container on the host machine at http://localhost:3000 on any browser



Hello from Multi-stage Dockerr!

Step-8 : Stop the container , remove the container and delete the image

```
sudo docker container ls -a     or    docker ps -a
sudo docker container stop <container-id>
sudo docker container rm <container–id>
sudo docker image rm <image-id>    or  docker rmi <image-id>
```