

Program 2:

Develop a multi-stage Dockerfile for container orchestration

31/10/2025

Project Directory:

|-build

|-dist

|-Dockerfile

|-package.json

|-package-lock.json

|-src

|--index.js

|-node_modules

Program code and steps for execution:

Step 1: Open the mylab directory and create the prog2 directory navigate into that then do the initialize a new Node.js project also do the install of express

Step 2: After the installation write the Dockerfile

```
FROM node:20-alpine AS builder

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .
RUN npm run build

FROM node:20-alpine

WORKDIR /app

COPY --from=builder /app/package*.json ./
COPY --from=builder /app/dist ./dist
COPY --from=builder /app/node_modules ./node_modules

EXPOSE 5000

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

Step 3: After the initialization and installation the package.json file will be automatically added so make some changes to the package.json in order to look like this

```
{
  "name": "program2",
  "version": "1.0.0",
  "main": "dist/index.js",
  "scripts": {
    "start": "node dist/index.js",
    "build": "mkdir -p dist && cp -r src/* dist/"
  },
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

Step 4: Create another folder inside the prog2 as src where the index.js file will be create

Step 5: After creating the index.js add the below code inside the index.js

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

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

app.listen(PORT,()=>{
  console.log('server running');
});
```

Step 6: Come out of the src folder

Step 7: Now build the docker for prog2

Step 8: Now run the docker of prog2 in interactive mode

```
Mon Nov 3 22:05
1rv24nc046_jamuna@jamuna-HP-Pavillon-Laptop-14-dv2xxx:~/Documents/Deveops/program2$ sudo docker build -t program2 .
[sudo] password for 1rv24nc046_jamuna:
[+] Building 3.5s (14/14) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 36B
=> [internal] load metadata for docker.io/library/node:20-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85871aae2af928ffa39f8a779265435
=> [internal] load build context
=> => transferring context: 41.47kB
=> CACHED [builder 2/6] WORKDIR /app
=> CACHED [builder 3/6] COPY package*.json ./
=> CACHED [builder 4/6] RUN npm install
=> CACHED [builder 5/6] COPY . .
=> CACHED [builder 6/6] RUN npm run build
=> CACHED [stage-1 3/5] COPY --from=builder /app/package*.json ./
=> CACHED [stage-1 4/5] COPY --from=builder /app/dist ./dist
=> CACHED [stage-1 5/5] COPY --from=builder /app/node_modules ./node_modules
=> exporting to image
=> => exporting layers
=> => writing image sha256:f01bdc6e15ab2b5bea557715778fd389a8e1f85ae3f89b28b4d1684e3f8de8
=> => naming to docker.io/library/program2
1rv24nc046_jamuna@jamuna-HP-Pavillon-Laptop-14-dv2xxx:~/Documents/Deveops/program2$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED    STATUS    PORTS                               NAMES
1bd688b53b42  program1   "python app.py"         3 minutes ago    Up 3 minutes    0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp    containerflask1
7cf3defbaf7d  nimmis/apache-php5  "/my_init"              23 minutes ago    Up 23 minutes    0.0.0.0:80->80/tcp, [::]:80->80/tcp            apache2
a4928843f996  mysql/mysql-server:5.6  "fentrypoint.sh mysq..." 23 minutes ago    Up 23 minutes (healthy)    0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp    mysql
47e578356b49  firstfile:latest        "bash"                   37 minutes ago    Up 37 minutes                                     container2
1d8ff5d66369  firstfile:latest        "bash"                   37 minutes ago    Up 37 minutes                                     container1
1rv24nc046_jamuna@jamuna-HP-Pavillon-Laptop-14-dv2xxx:~/Documents/Deveops/program2$ sudo docker run -it -p 3000:3000 program2
server running
```

Step 9: Now in the browser search for <http://localhost:3000>

