

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

```
one more file
{
  "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 created

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

```
one more file
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

```
[root@irv24mC46_jamuna] ~]# irv24mC46_jamuna@irv24mC46_jamuna:~/Documents/Deveops/program2$ sudo docker build -t program2 .
[sudo] password for irv24mC46_jamuna:
[+] Building 3.5s (14/14) FINISHED
   => [internal] load build definition from Dockerfile
   => [internal] loading dockerfile: 36B
   => [internal] load build context
   => [internal] load .dockerignore
   => [internal] load context: 28
[+] Builder 1/6 FROM docker.io/library/node:20-alpine
[+] Internal 1/1 Load build context
   => [internal] transfer context: 41.47kB
[+] Builder 2/6 RUN npm install --frozen-lockfile
[CACHED] builder 2/6 RUN npm install --frozen-lockfile
[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 layers
[+] Exporting layers
[+] Writing image sha256:f01bdd6e615ab2b5bee5577715778fd399a0e1f85ae3f89b20b4d1684e3f9de0
[+] Naming to docker.io/library/program2
[+] Program 1/1 Image: docker.io/library/program2:latest
[+] Program 1/1 Command: /bin/sh -c node app.js
[+] Program 1/1 Create: 3 minutes ago
[+] Program 1/1 Status: Up 3 minutes
[+] Program 1/1 Ports: 0.0.0.0:5000 ->5000/tcp, [:]:5000 ->5000/tcp
[+] Program 1/1 Networks: containerflask1
[+] Program 2/1 Image: docker.io/library/program2:latest
[+] Program 2/1 Command: /bin/sh -c apache2 -k start
[+] Program 2/1 Create: 3 minutes ago
[+] Program 2/1 Status: Up 23 minutes
[+] Program 2/1 Ports: 0.0.0.0:80 ->80/tcp, [:]:80 ->80/tcp
[+] Program 2/1 Networks: apache2
[+] Program 3/1 Image: docker.io/library/program2:latest
[+] Program 3/1 Command: /bin/sh -c mysqld_safe &
[+] Program 3/1 Create: 3 minutes ago
[+] Program 3/1 Status: Up 23 minutes
[+] Program 3/1 Ports: 0.0.0.0:3306 ->3306/tcp, [:]:3306 ->3306/tcp
[+] Program 3/1 Networks: mysql
[+] Program 4/1 Image: docker.io/library/program2:latest
[+] Program 4/1 Command: /bin/sh -c rm -rf /var/www/html/*
[+] Program 4/1 Create: 3 minutes ago
[+] Program 4/1 Status: Up 37 minutes
[+] Program 4/1 Ports: 0.0.0.0:80 ->80/tcp
[+] Program 4/1 Networks: container2
[+] Program 5/1 Image: docker.io/library/program2:latest
[+] Program 5/1 Command: /bin/sh -c rm -rf /var/www/html/*
[+] Program 5/1 Create: 3 minutes ago
[+] Program 5/1 Status: Up 37 minutes
[+] Program 5/1 Ports: 0.0.0.0:80 ->80/tcp
[+] Program 5/1 Networks: container1
[+] Program 6/1 Image: docker.io/library/program2:latest
[+] Program 6/1 Command: /bin/sh -c rm -rf /var/www/html/*
[+] Program 6/1 Create: 3 minutes ago
[+] Program 6/1 Status: Up 37 minutes
[+] Program 6/1 Ports: 0.0.0.0:80 ->80/tcp
[+] Program 6/1 Networks: container1
```

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

