Program 2: Develop a Multi-Stage Dockerfile for Container Orchestration.

Step 1: Create a project folder mkdir multi-stage-docker-demo cd multi-stage-docker-demo

Step 2: Create subfolder for the app mkdir app

cd app

Step 3: Create your app files

Create package.json

nano package.json

Then paste this inside:

{

"name": "multi-stage-demo",

"version": "1.0.0",

"main": "server.js", "scripts": {

"start": "node server.js"

},

"dependencies": { "express": "^4.18.2"

}

}

Create index.js

nano index.js

const express = require('express'); const app = express();

app.get('/', (req, res) => {

res.send('Hello from Multi-Stage Docker Build!');

});

const PORT = process.env.PORT || 8080;

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

Step 4: Create Dockerfile

Step 5: Build Docker Image docker build -t program2-app .

Step 6: Run the Container

docker run -d -p 3000:3000 program2-app

Check running containers: docker ps

```
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog1$ cd ..
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile$ cd prog2
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2$ ls
Dockerfile  node_modules  package.json  package-lock.json  src
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2$ cat Dockerfile
#stage 1: Build Stage
FROM node:20-alpine AS builder

#set working directory
WORKDIR /app

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

#copy application source code
COPY . .

#build the application (if using a build step, e.g., for React, Next.js)
RUN npm run build

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

#set working directory
WORKDIR /app

#copy only necessary files from build stage
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 application port
EXPOSE 3000

#start the application
CMD ["node", "dist/index.js"]
```

```
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2$ cat package.json
{
  "name": "program-2",
  "version": "1.0.0",
  "description": "A simple Node.js app with multi-stage Dockerfile",
  "main": "dist/index.js",
  "scripts": {
    "start": "node dist/index.js",
    "build": "mkdir -p dist && cp -r src/* dist/"
  },
  "author": "girish",
  "license": "MIT",
  "dependencies": {
    "express": "4.18.2"
  }
}

1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2$ cd src
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2/src$ ls
index.js
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2/src$ cat index.js

const express = require('express');


const app = express();

const PORT = 3000;


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

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

```
});
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2/src$ sudo docker build -t prog2 .
[+] Building 0.1s (1/1) FINISHED                                                                docker:default
 => [internal] load build definition from Dockerfile                                                     0.0s
 => => transferring dockerfile: 2B                                                                       0.0s
ERROR: failed to build: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2/src$ cd ..
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2$ sudo docker build -t prog2 .
[+] Building 3.2s (15/15) FINISHED                                                              docker:default
 => [internal] load build definition from Dockerfile                                                     0.0s
 => => transferring dockerfile: 765B                                                                     0.0s
 => [internal] load metadata for docker.io/library/node:20-alpine                                        2.9s
 => [internal] load .dockerignore                                                                        0.0s
 => => transferring context: 2B                                                                          0.0s
 => [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af020ffa39f0a770265435   0.0s
 => [internal] load build context                                                                        0.1s
 => => transferring context: 43.48kB                                                                     0.1s
 => CACHED [builder 2/6] WORKDIR /app                                                                    0.0s
 => CACHED [builder 3/6] COPY package.json package-lock.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 3/6] COPY --from=builder /app/package.json ./                                        0.0s
 => CACHED [stage-1 4/6] COPY --from=builder /app/package-lock.json ./                                   0.0s
 => CACHED [stage-1 5/6] COPY --from=builder /app/dist ./dist                                            0.0s
 => CACHED [stage-1 6/6] COPY --from=builder /app/node_modules ./node_modules                            0.0s
 => exporting to image                                                                                   0.0s
 => => exporting layers                                                                                  0.0s
 => => writing image sha256:0538137ae630708fe52c75452deacb0f4622e1f9c83bbc245000d7da8cae580d             0.0s
 => => naming to docker.io/library/prog2                                                                 0.0s
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog2$ sudo docker run -it -p 3000:3000 prog2
Server running on port 3000
```