Program 2 : Develop a Multi Stage Dockerfile for Container Orchestration

Step 1 : Create the Project Directory.

1. Create the main folder for your project.
   mkdir Program-2
2. 2. Move into the new directory.
   cd Program-2

```
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program1$ cd ..
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~$ mkdir program2
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~$ cd program2
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program2$ mkdir src
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program2$ nano src/index.js
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program2$ nano package.json
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program2$ nano Dockerfile
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program2$
```

Step 2: Create the Source Code

1. Create the src folder.
   mkdir src
2. Create the index.js file inside src.
   index.js

```
 GNU nano 7.2                                                src/index.js
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 3 : Create a package.json
>nano package.json
> npm init -y

```
 GNU nano 7.2                                                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": "Deepika K",
"license": "MIT",
"dependencies": {
"express": "4.18.2"
}
}
```

Step 4 : Create a Dockerfile
> nano Dockerfile

```
  GNU nano 7.2                                                Dockerfile
# Stage 1: Build Stage
# We use a full Node.js image and name this stage 'builder'
FROM node:18-alpine AS builder
# Set the working directory inside the container
WORKDIR /app
# Copy package files first to leverage Docker's layer cache
COPY package.json package-lock.json ./
# Install all dependencies (including devDependencies, if any)
RUN npm install
# Copy the rest of the application source code
COPY . .
# Run the build script defined in package.json
# This will create the /app/dist folder inside this stage
RUN npm run build
# ---
# Stage 2: Production Stage
# We start from a fresh, lightweight alpine image
FROM node:18-alpine
# Set the working directory
WORKDIR /app
# Copy ONLY the necessary files from the 'builder' 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 port the application runs on
EXPOSE 3000

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Und
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Red
```

Step 5 : Install node modules with the following commands
> npm install

Step 6 : Execute the Docker build command
> docker build -t program2 .

```
up to date, audited 70 packages in 2s

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

found 0 vulnerabilities
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program2$ docker build -t program2 .
[+] Building 17.7s (15/15) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                          0.2s
 => => transferring dockerfile: 1.08kB                                                                        0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                             4.8s
 => [internal] load .dockerignore                                                                             0.3s
 => => transferring context: 2B                                                                               0.0s
 => [builder 1/6] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8c  0.0s
 => [internal] load build context                                                                             0.5s
 => => transferring context: 2.31MB                                                                           0.2s
 => 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
 => [builder 5/6] COPY . .                                                                                    4.1s
 => [builder 6/6] RUN npm run build                                                                           4.4s
 => 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
 => [stage-1 5/6] COPY --from=builder /app/dist ./dist                                                        0.4s
 => [stage-1 6/6] COPY --from=builder /app/node_modules ./node_modules                                        0.9s
 => exporting to image                                                                                        0.7s
 => => exporting layers                                                                                       0.5s
 => => writing image sha256:91e149c316e49eaafde78d3c4bbca36ff1a80a47dfcca2ae1ca8ae1784dc5922                  0.0s
 => => naming to docker.io/library/program2                                                                   0.0s
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program2$
```

Step 7 : Run the Docker run command

> docker run -it -p 3000:3000 program2

```
1rv24mc101_shreya@shreya-Lenovo-IdeaPad-S145-15IWL:~/program2$ docker run -it -p 3000:3000 program2
Server running on port 3000
```

Step 8 : Verifying the details on the browser with the text -> “Hello from multi stage Docker”