Develop a multistage dockerfile for container orchestration

Step 1 - Create the following directory structure

- Run *npm init* to initialize a node app
- Run *npm i express* to install express
- Create a Dockerfile by running *touch Dockerfile*

```
└─(10:13:50)—> tree -L 2 -I node_modules/ .
.
├── build
├── dist
├── Dockerfile
├── package.json
├── package-lock.json
└── src
    └── index.js

3 directories, 4 files
```

Step 2 - Create /src/index.js and initialize a basic express server

```
┌─(~/Desktop/devops/lab2/src)──────────────────────────────────────
└─(10:48:56)—> cat index.js
const express = require("express")
const app = express()
const PORT = 3000


app.get("/", (req,res)=>{
        res.send("hello from docker")
})

app.listen(PORT, ()=>console.log("Listening on", PORT));
```

Step 3 - Modify package.json to include the following scripts

```
┌──(~/Desktop/devops/lab2/src)──────────────────────────
└─(10:49:52)──> cd .. && cat package.json
{
  "name": "lab2",
  "version": "1.0.0",
  "description": "",
  "license": "ISC",
  "author": "",
  "type": "commonjs",
  "main": "index.js",
  "scripts": {
    "start":"node dist/index.js",
    "build":"mkdir -p dist && cp -r src/* dist/"
  },
  "dependencies": {
    "express": "^5.1.0"
  }
}
```

Step 4 - Write the Dockerfile

```
┌──(~/Desktop/devops/lab2)──────────────────────────────(raksha@asus:pts/2)─┐
└─(10:50:00)──> cat Dockerfile                                  ──(Fri,Oct31)─┘
#Stage 1 - BUILD

#base builder image
FROM node:20-alpine AS builder

#set workdir
WORKDIR /app

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

#copy source code
COPY . .

#build app
RUN npm run build


#Stage 2 - PRODUCTION

#base image
FROM node:20-alpine

#copy important files
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 port
EXPOSE 3000

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

Step 5 - Build the image by running
sudo docker build . -t node_image

```
┌──(~/Desktop/devops/lab2)
└─(10:22:01)──> sudo docker build . -t node_image
[+] Building 7.0s (15/15) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 603B
 => [internal] load metadata for docker.io/library/node:20-alpine
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 2.70MB
 => CACHED [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af020ffa39f0a770265435
 => CACHED [builder 2/6] WORKDIR /app
 => [builder 3/6] COPY package.json package-lock.json ./
 => [builder 4/6] RUN npm install
 => [builder 5/6] COPY . .
 => [builder 6/6] RUN npm run build
 => [stage-1 2/5] COPY --from=builder /app/package.json ./
 => [stage-1 3/5] COPY --from=builder /app/package-lock.json ./
 => [stage-1 4/5] COPY --from=builder /app/dist ./dist
 => [stage-1 5/5] COPY --from=builder /app/node_modules ./node_modules
 => exporting to image
 => => exporting layers
 => => writing image sha256:28259176952b14ccd6fb78841a2c0d40f6d4b103773afe06c725b15400dbf338
 => => naming to docker.io/library/node_image
```

Step 6 - Create and run a new container of the image by running
sudo docker run -p 3000:3000 node_image

```
┌──(~/Desktop/devops/lab2)
└─(10:29:06)──> sudo docker run -p 3000:3000 node_image
Listening on 3000
```

Step 7 - Verify output