

Program2 : Multistage Dockerfile for a Node.js App

We have to create files and folder as:

—>src

- - index.js

—>Dockerfile

—>package.json

—>package-lock.json

—>node_modules

We have to create a folder called program2

```
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~$ cd program2
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$
```

Step1:create a Dockerfile without any extension and we have to add the content in sudo Dockerfile

```
GNU nano 7.2
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
# The command to start the application
CMD [ "node", "dist/index.js" ]
```

Step2:

Create a index file as [index.js](#) in src

```
GNU nano 7.2
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 is running on port ${PORT}`);
});
```

Step3 : now we have run the following commands

npm init -y

npm install express

npm install

```
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ nano src/index.js
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ npm init -y
Wrote to /home/sindhra/program2/package.json:
```

```
{
  "name": "program2",
  "version": "1.0.0",
  "description": "A simple Node.js app with the multi stage docker",
  "main": "dist/index.js",
  "scripts": {
    "start": "node dist/index.js",
    "build": "mkdir -p dist && cp -r src/* dist/"
  },
  "author": "Sindhura",
  "license": "MIT",
  "dependencies": {
    "express": "^4.21.2"
  },
  "devDependencies": {},
  "keywords": []
}
```

```
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ □
```

```

    "author": "Sindhura",
    "license": "MIT",
    "dependencies": {
      "express": "^4.21.2"
    },
    "devDependencies": {},
    "keywords": []
  }
}

```

```

1RV24MC104_SINDHURA@sindhura-IdeaPad-Slim-5-16IAH8:~/program2$ npm intall express
Unknown command: "intall"

```

```

Did you mean this?
  npm install # Install a package

```

```

To see a list of supported npm commands, run:
  npm help

```

```

1RV24MC104_SINDHURA@sindhura-IdeaPad-Slim-5-16IAH8:~/program2$ npm install express

```

```

up to date, audited 70 packages in 740ms

```

```

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

```

```

found 0 vulnerabilities

```

```

1RV24MC104_SINDHURA@sindhura-IdeaPad-Slim-5-16IAH8:~/program2$ 

```

Step4

Now we have to run Docker build command
 docker build -t program2 .

```

found 0 vulnerabilities
1RV24MC104_SINDHURA@sindhura-IdeaPad-Slim-5-16IAH8:~/program2$ docker build -t program2 .
[+] Building 10.3s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 993B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 97.52kB
=> [builder 1/6] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
=> 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 3/6] COPY --from=builder /app/package.json ./
=> [stage-1 4/6] COPY --from=builder /app/package-lock.json ./
=> [stage-1 5/6] COPY --from=builder /app/dist ./dist
=> [stage-1 6/6] COPY --from=builder /app/node_modules ./node_modules
=> exporting to image
=> => exporting layers
=> => writing image sha256:bfa1b615fa3941769242482fafb52af8a19cb4723b8449998c1944d1161fba1b
=> => naming to docker.io/library/program2
1RV24MC104_SINDHURA@sindhura-IdeaPad-Slim-5-16IAH8:~/program2$ 

```

Step5 we have to specify the port number

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

```
Run 'docker run --help' for more information
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ docker rm a6cb9c31d033
Error response from daemon: cannot remove container "a6cb9c31d033": container is running: stop the container before removing or force remove
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ docker stop a6cb9c31d033
^C1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ docker stop a6cb9c31d033
a6cb9c31d033
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ docker rm a6cb9c31d033
a6cb9c31d033
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ docker run -p 3000:3000 -d my-node-app
a13d9eb67873e4b4cf4c7f8d777b83b29d9cfdfe07c9389e80fcb288b7f54f7
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
a13d9eb67873   my-node-app    "docker-entrypoint.s..." About a minute ago Up About a minute   0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp   vibrant_rosalind
1RV24MC104_SINDHURA@sindhra-IdeaPad-Slim-5-16IAH8:~/program2$
```

Step6: we can check our output in localhost:3000

