

DevOps Lab

Program 2- Creating a Multi-Stage Dockerfile

Project Structure:

```
program_2
  Dockerfile
  package.json
  package-lock.json
  node_modules/
  src/
  index.js
```

Step 1: Create Project Folder

```
mkdir program_2
cd program_2
```

Step 2: Initialize Node.js Project

```
npm init -y
```

Step 3: Install Express Framework `npm install express`

Step 4: Write the DockerFile code and the [index.js](#) code

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

The screenshot shows a VS Code editor with a project named 'second'. The Explorer sidebar on the left shows the file structure: `node_modules`, `src`, `index.js`, `Dockerfile`, `package-lock.json`, and `package.json`. The Dockerfile editor shows the following content:

```
1 #Stage 1: Building Stage|
2 FROM node:20-alpine AS builder
3 WORKDIR /app
4 COPY package.json ./
5 RUN npm install
6
7 COPY src ./src
8 RUN npm run build
9
10 #Stage 2: Production
11 FROM node:20-alpine
```

The terminal at the bottom shows the output of the `docker build` command, listing the repository, tag, image ID, creation time, and size for the 'second' image. The output is as follows:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
second	latest	7df4a917cf06	4 minutes ago	137MB
first	latest	75f9b4dd0079	14 minutes ago	142MB
<none>	<none>	3be63f941bb5	17 minutes ago	134MB
<none>	<none>	3948174b44b7	About an hour ago	137MB
server	2.0	e0c552f11552	2 days ago	17.8MB
none	1.0	5aa124d45007	6 days ago	147MB
wordpress	latest	7332768c717f	4 weeks ago	734MB

The terminal also shows the command `docker run -d -p 3000:3000 second` and the command `docker images` which lists the same images as the table above.

Step 6: Verify the output on localhost:3000

