DockerFile — Multiple Stage build

How To Dockerize an Angular Application with multistage build

- A multistage build allows you to use multiple images to build a final product.
- In a multi-stage build, you have a single Dockerfile but can define multiple images inside it to help build the final image.
- This approach to keeping Docker images small is using multistage builds

Let's try to understand the multi-stage docker file that you can find in the below GitHub page.

Clone the sample angular code

```
git clone https://github.com/akshu20791/medium-angular-docker
```

You will see a DockerFile with the below content.

```
# Step 1: Build the app in image 'builder'
FROM node:12.8-alpine AS builder

# Set the working directory
WORKDIR /usr/src/app

# Add the source code to app
COPY . .

# Generate the build of the application
RUN yarn && yarn build
```

```
# Step 2: Use build output from 'builder'
FROM nginx:stable-alpine

# Use Labels for organize Docker Images
LABEL version="1.0"

# Add the nginx.conf file
COPY nginx.conf /etc/nginx/nginx.conf

# Set the working directory
WORKDIR /usr/share/nginx/html

# Copy the build output to replace the default nginx contents
COPY --from=builder /usr/src/app/dist/my-angular-app/ .
```

This Dockerfile has two stages.

Stage 1:

- It will initialize a first build stage and this stage is named as `build.`
- Angular build in a new layer on top of the base node image. After this instruction is executed, the build output is stored under `usr/local/app/dist/my-angular-app`

Stage 2:

- Initializes a secondary build stage

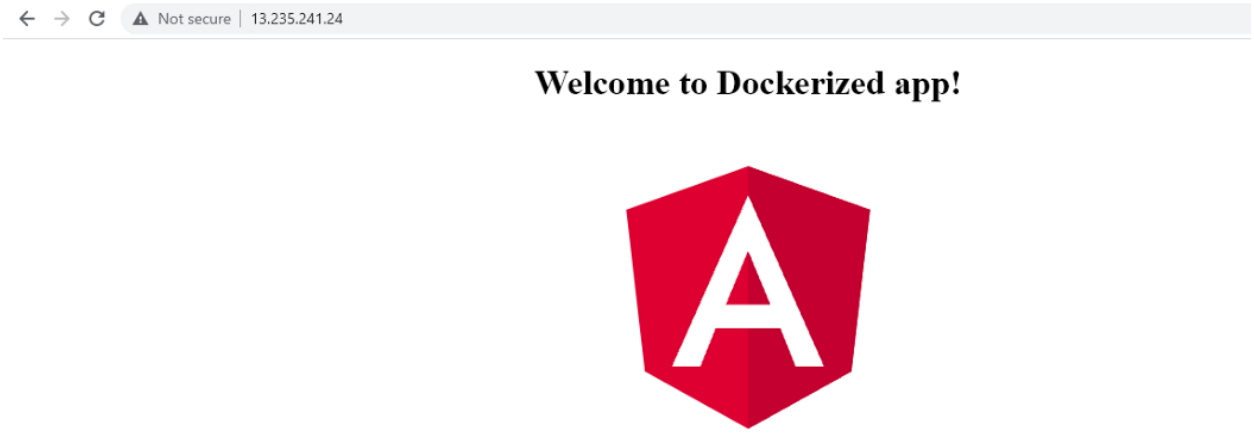
- Copies the build output generated in stage 1 (`--from=build`) to replace the default Nginx contents in this path - /usr/share/nginx/html.

Now, let's build a docker image and run a container

```
$ docker build -t angular:v1.0.0 .
$ docker run -d  -p 80:80 --name angular-app angular:v1.0.0
```

```
ubuntu@ip-172-31-47-64:~/docker/medium-angular-docker$ docker build -t angular:v1.0.0 .
Sending build context to Docker daemon  1.657MB
Step 1/9 : FROM node:12.8-alpine AS builder
 ---> eaeb7e99eabd
Step 2/9 : WORKDIR /usr/src/app
 ---> Using cache
 ---> bee2f93fcf63
Step 3/9 : COPY . .
 ---> Using cache
 ---> 19a46bcd35b5
Step 4/9 : RUN yarn && yarn build
 ---> Using cache
 ---> 13871d6d5ba1
Step 5/9 : FROM nginx:stable-alpine
 ---> 4146b18ae794
Step 6/9 : LABEL version="1.0"
 ---> Using cache
 ---> 22ce7b2fbbe2
Step 7/9 : COPY nginx.conf /etc/nginx/nginx.conf
 ---> Using cache
 ---> 2149c683125d
Step 8/9 : WORKDIR /usr/share/nginx/html
 ---> Using cache
 ---> aafd5e4a6777
Step 9/9 : COPY --from=builder /usr/src/app/dist/my-angular-app/ .
 ---> Using cache
 ---> bf274f9157aa
Successfully built bf274f9157aa
Successfully tagged angular:v1.0.0
ubuntu@ip-172-31-47-64:~/docker/medium-angular-docker$
ubuntu@ip-172-31-47-64:~/docker/medium-angular-docker$ docker run -d -p 80:80 --name angular-app angular:v1.0.0
cd299abc42a82a0aa22598a7bfeaa3458831e0c46897d5a2452a0264547395ec
ubuntu@ip-172-31-47-64:~/docker/medium-angular-docker$
```

Test the application


Not secure | 13.235.241.24
Welcome to Dockerized app!