To deploy your Angular and Spring Boot applications as a single Docker image with Nginx as a reverse proxy, you’ll need to:

1. Build both applications.
2. Configure Nginx to serve the Angular frontend and proxy backend requests to the Spring Boot application.
3. Package everything into one Docker image.

Here’s a detailed guide:

**1. Build Both Applications**

Make sure both applications are built and ready to be served.

* **Angular App**: Build the Angular app for production.

bash

Copy code

# Inside the Angular project folder

ng build --prod

This will generate static files in the dist/your-angular-app directory.

* **Spring Boot App**: Package the Spring Boot app as a JAR file.

bash

Copy code

# Inside the Spring Boot project folder

./mvnw clean package

This should generate a target/your-spring-boot-app.jar file.

**2. Set Up Nginx as a Reverse Proxy**

Create an Nginx configuration file to serve the Angular app and reverse-proxy requests to the Spring Boot app.

Create an nginx.conf file:

nginx

Copy code

server {

listen 80;

# Serve Angular app

location / {

root /usr/share/nginx/html;

try\_files $uri $uri/ /index.html;

}

# Proxy requests to the Spring Boot backend

location /api/ {

proxy\_pass http://localhost:8080;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

In this configuration:

* Angular files are served from /usr/share/nginx/html.
* API requests prefixed with /api/ are proxied to http://localhost:8080, which will be the Spring Boot app running on port 8080.

**3. Create a Dockerfile for the Combined Application**

Now, create a Dockerfile at the root of your project to combine both applications and configure Nginx.

Dockerfile

Copy code

# Step 1: Build Angular app

FROM node:16 AS build-angular

WORKDIR /app

COPY path/to/angular-app .

RUN npm install && npm run build --prod

# Step 2: Build Spring Boot app

FROM maven:3.8.5-openjdk-11 AS build-spring

WORKDIR /app

COPY path/to/spring-boot-app .

RUN mvn clean package

# Step 3: Prepare Nginx with built Angular and Spring Boot artifacts

FROM nginx:alpine

# Copy Nginx configuration

COPY nginx.conf /etc/nginx/conf.d/default.conf

# Copy Angular build files to Nginx HTML directory

COPY --from=build-angular /app/dist/your-angular-app /usr/share/nginx/html

# Copy Spring Boot JAR and run it in the background

COPY --from=build-spring /app/target/your-spring-boot-app.jar /app.jar

EXPOSE 80

# Start both Nginx and Spring Boot

CMD sh -c "java -jar /app.jar & nginx -g 'daemon off;'"

**4. Build and Test the Docker Image**

Build the Docker image:

bash

Copy code

docker build -t my-combined-app .

Run it locally to test:

bash

Copy code

docker run -p 80:80 my-combined-app

**5. Push the Image to Amazon ECR**

Once verified, push the Docker image to Amazon ECR.

1. Create an ECR repository (if you haven’t already).
2. Tag the image:

bash

Copy code

docker tag my-combined-app:latest [your-account-id].dkr.ecr.[region].amazonaws.com/my-combined-app:latest

1. Push to ECR:

bash

Copy code

docker push [your-account-id].dkr.ecr.[region].amazonaws.com/my-combined-app:latest

**6. Deploy to AWS ECS**

1. **Define an ECS Task Definition**: Set up a task definition that uses the combined image.
2. **Create an ECS Service** to manage the task and keep it running.

With this setup, your Angular app will be served directly by Nginx, and Nginx will forward API requests to the Spring Boot backend within the same container. This approach simplifies deployment as a single image in AWS ECS.