How to dockerize Spring Boot + React apps

Luiz Gustavo De O. Costa

.

Follow

6 min read

.

May 10, 2022

--

1

Listen

Share

A long time ago we lived in a land without a container, but we had (we still have) a bunch of servers and dedicated servers running our applications.

containers

In this how to article, I'll explain how to use Docker in your Spring Boot and React application and the last but not the least, use of docker-compose ■.

1. Docker architecture

The architecture diagram below is an image from docker.com, for more information visit it.

In this case, instead of having Redis, NGINX etc, we will have a Spring Boot and a React application.

Docker architecture

1.1. Docker lifecycle

The Docker lifecycle happens in this way: Create, Running, Pause/Unpause, Stop and Kill. I'll cover the two first, the Create and Running.

Docker lifecycle — from docker-saigon

2. Hands on

This is the funniest part, I really love it ■, lets see how the project is and how to create the file for back and front end.

2.1. Project structure

The project is available on my GitHub. This project has 2 projects, the back-end and the front-end parts.

Under the folder 16-bits-zero-hero, we've the projects and the docker-compose file (we talk about this file later).

The folder zero2hero has the Java Spring Boot project, and the folder ui-app as the name says, is the React project.

If you want, you can run the projects independently, but remember, the front-end application needs a source of data ■.

2.2. Spring Boot

For the Spring Boot, Gradle will use the as dependency manager, then let's start.

The application will not have any external dependency, ie, database, queue and so on, since we're using H2 in memory database.

Let's analyze the Dockerfile.

Line 1 → Which image we gonna use, since is a Java project and using the version 11 was decided to use adoptopenjdk 11.

Line 2 → Where is the jar file and the name. This can be get dynamically as well.

Line 3 → Copy the jar using the name **zeroToHero.jar**

Line 4 → Run the jar, using java -jar zeroToHero.jar

Dockerfile for Back-end application

2.3. Build the project

In order to have an image, we need to generate the jar. Since this project is based on Gradle, use the command **./gradlew build**

Build command + artifact generated

After the artifact has been generated let's create the image.

2.4. Create the image

To create the image, you need to have the docker installed. I'm using the Docker Desktop program.

Since it is not my first time to run the docker build command, I already have the images, because this time is 0.0s.

The command to create the image is

```
docker build -t <IMAGE-NAME>:<TAG> .
docker build -t 16-bits/zero2hero:0.0.1 .
```

Docker build output

If you want, browse the image using the command `docker image ls | grep 16-bits/zero2hero`

docker image ls output

2.5. Run the image

Using the terminal, type the command below

```
docker run --name <NAME-TO-BE-SHOW> -p8080:8080 -d <IMAGE_ID>
docker run --name 16-bits-hero -p8080:8080 -d 25430fca19a8
```

docker run command

Is possible to check if the container is running, since we started using the option -d, the log will not show on the terminal.

Docker Desktop — Containers

If you want to access the Swagger or the H2 client, check the project documentation here or use ***james***/***bond***.

Swagger UI — Back-end

2.6. React App

For the React application, was used the Dockerfile below, let's analyze line by line.

Line 1 → Which image will be used, in this case Node 17.

Line 2 → Application container directory, ie, app

Line 3 → Adding the dependencies to path

Line 4 and 5 → Copying the dependency names

Line 6 → Installing the dependencies

Line 7 → Copy from where I am, references here.

Line 8 → Expose the port, references here.

Line 9 → Run the application using npm

Dockerfile for Front-end application

2.7. Create the image

The command to create the image is

```
docker build -t <IMAGE-NAME>:<TAG> .
docker build -t 16-bits-ui:0.0.1 .
```

Docker build output

2.8. Run the image

First, find the image using `docker image ls | grep 16-bits-ui`

docker image output

Using the terminal, type the command below

```
docker run --name <NAME-TO-BE-SHOW> -p8080:8080 -d <IMAGE_ID>
docker run --name 16-bits-ui  -p3000:3000 eda7e37e2b75
```

docker run output

A different way to create the image and the run container to show is not the only way to do it.

The container is up and running

Front-end application running

The two images running ■

Docker Desktop — Containers

Now it's time to use the back-end services, ie, authentication/authorization and the api.

Sing-in using boba/fett
Movie list

3. Docker compose

From the Docker site "*Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. To learn more about all the features of Compose*"

At the root folder, we've the docker-compose.yml file. If you navigate through the section 2, you can notice the work to start on one container, then another etc.

Now, just using one file starts the two services.

The amazing thing here is the cohesion. Reading the file, we can notice 2 services, one back-end and other front-end, the image name and then the container and ports.

This file can be executed using the command line or inside the IntelliJ.

docker compose file

3.1. Run

Go to the root folder and type the command `docker-compose up` . The two images will start.

docker-compose up output
Docker Desktop — Containers
Front-end application running
Movie list

4. Tech stack

- Java 11 — openJDK
- SpringBoot — 2.6.3
- React — 17.0.2
- Docker Desktop — 4.6.1
- IntelliJ — 2021.2.4

5. References

GitHub - luizgustavocosta/16-bits-zero-to-hero: Java project to show how to run front-end (React)…

■ Zero to hero ■ using React and SpringBoot Create a project with front-end and back-end to run easily locally or…

github.com

Docker Internals

This post was the basis for a joint event with the grokking engineering community in Saigon. The event was centered…

docker-saigon.github.io

What is the Docker container lifecycle?

Creator: abhilash

www.educative.io

Dockerfile reference

This page describes the commands you can use in a Dockerfile. When you are done reading this page, refer to…

docs.docker.com

Compose specification

The Compose file is a YAML file defining services, networks, and volumes for a Docker application. The latest and…

docs.docker.com