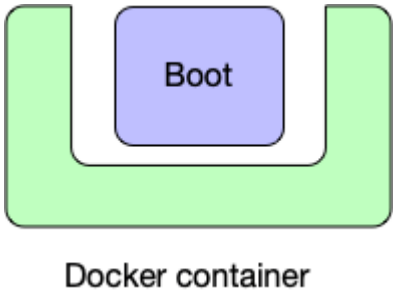
Spring Boot Docker ¿Como configurarlo? . Hoy en queremos que nuestras aplicaciones sean fáciles de mover de un entorno a otro sin tener que pelear con configuraciones distintas o dependencias. Docker es una herramienta increíble para lograr esto porque encapsula nuestra aplicación en un contenedor, que es como una “caja” que podemos ejecutar en cualquier lado sin preocuparnos por las diferencias entre entornos.



En este artículo, te voy a mostrar cómo crear una aplicación simple en Spring Boot y meterla en un contenedor Docker para luego poder desplegarla de manera rápida.

1. Crear una aplicación básica de Spring Boot Docker

Primero, vamos a construir una aplicación sencilla de Spring Boot que exponga un endpoint REST.

Paso 1: Configura el proyecto

Lo más fácil es generar tu proyecto en Spring Initializr. Ahí selecciona lo básico:

- Spring Web: Para poder crear un API REST.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
```

```
    </dependency>
</dependencies>
```

Paso 2: Crea el controlador REST

Ahora, creamos un controlador que exponga un endpoint simple, algo como esto:

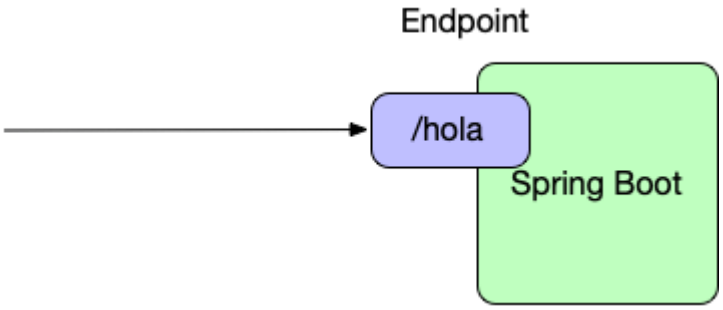
```
package com.arquitecturajava

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HolaController {

    @GetMapping("/hola")
    public String hola() {
        return "¡Hola desde Docker!";
    }
}
```

Cuando accedas a /hola, este endpoint te devolverá un mensaje simple.



Recordemos que también tendremos el código de la clase principal.

```
package com.arquitecturajava;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class HolaApplication {

    public static void main(String[] args) {
        SpringApplication.run(HolaApplication.class, args);
    }

}
```

Paso 3: Prueba tu aplicación localmente

Antes de meterla en Docker, asegurémonos de que la aplicación funciona. Ejecuta el programa main y ponla en marcha. Luego, abre tu navegador y ve a http://localhost:8080/hola. Deberías ver el mensaje “¡Hola desde Docker!”.

2. Spring Boot Docker

Ahora vamos a empaquetar la aplicación en un contenedor Docker.

Paso 1: Crear el archivo Dockerfile

En la raíz de tu proyecto, crea un archivo llamado Dockerfile. Este archivo contiene las instrucciones que Docker necesita para construir tu imagen.

```
# Partimos de una imagen de Java 17 con Alpine (más ligera)
FROM openjdk:17-jdk-alpine

# Establecemos el directorio de trabajo dentro del contenedor
WORKDIR /app
```

```
# Copiamos el JAR generado en el contenedor
COPY target/hola-0.0.1-SNAPSHOT.jar app.jar

# Exponemos el puerto 8080 (el que usa Spring Boot por defecto)
EXPOSE 8080

# Comando para ejecutar la aplicación cuando el contenedor arranque
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Paso 2: Empaquetar la aplicación como JAR

Necesitamos generar el archivo .jar de nuestra aplicación. Ejecuta el siguiente comando:

```
./mvnw clean package
```

Esto creará un archivo hola-0.0.1-SNAPSHOT.jar dentro del directorio target/.

Paso 3: Crear la imagen para Spring Boot Docker

Ahora vamos a usar el Dockerfile para crear una imagen Docker de nuestra app. Ejecuta este comando en la raíz del proyecto:

```
docker build -t springboot-docker .
```

Con esto, Docker creará una imagen llamada springboot-docker apoyandose en el dockerfile.

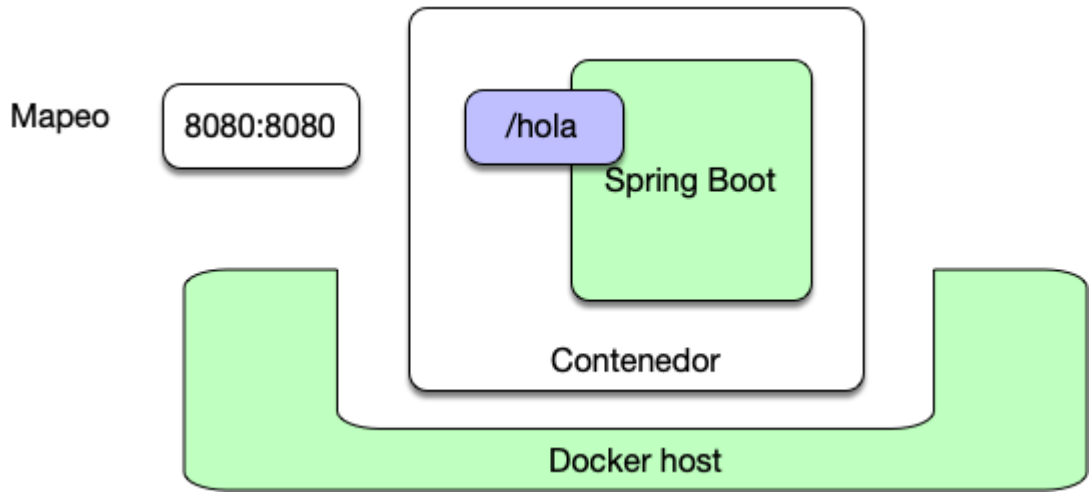
Paso 4: Ejecutar el contenedor

Una vez que tenemos la imagen, podemos lanzar un contenedor con este comando:

```
docker run -p 8080:8080 springboot-docker
```

Este comando levanta el contenedor y mapea el puerto 8080 del contenedor al puerto 8080

de tu máquina. Ahora, si accedes a http://localhost:8080/hola, verás el mismo mensaje “¡Hola desde Docker!”.



Spring Boot Docker y Conclusiones

Usar Docker con Spring Boot hace que el despliegue de aplicaciones sea mucho más sencillo y manejable. La posibilidad de empaquetar todo lo necesario para ejecutar tu app en un contenedor te ahorra muchos dolores de cabeza cuando llevas tu aplicación a producción. ¡Ahora, tú también puedes empezar a dockerizar tus apps de Spring Boot y olvidarte de problemas de configuración en distintos entornos!

Otros artículos relacionados:

- Spring Boot ¿Qué es y cómo funciona?
- ¿Que es GraalVM ?
- Curso Docker y manejo de Contenedores
- ¿Qué es Docker y para qué sirve?
- Java JAR , Java Archive y empaquetamiento