**Kickstart Your Spring Boot Application Development**

Se crea una aplicación springboot utilizando el generador de inicio rápido para proyectos de Spring *Spring Initializr*.

Posteriormente, se ejecuta el comando toma ./mvnw package para empaquetar el código compilado en un archivo .JAR

Luego, se ejecuta el jar con el siguiente código: java -jar target/spring-boot-docker-0.0.1-SNAPSHOT.jar del jar anterior que se acabo de compilar, y es de esta forma que se puede acceder al mensaje inicial "Hello world"

**Contenedorización de la aplicación Spring Boot**

Docker permite agrupar la aplicación Spring Boot completa, el tiempo de ejecución, la configuración y las dependencias a nivel del sistema operativo. Esto incluye todo lo necesario para enviar una aplicación web multiplataforma y multiarquitectura

Se crea un archivo Dockerfile vacío en su proyecto Spring Boot. "touch Dockerfile", y le define la imagen base Docker eclipse-temurin:17-jdk-focal disponible en Docker Hub "FROM eclipse-temurin:17-jdk-focal".

Se crea un directorio para albergar el código de aplicación de nuestra imagen, WORKDIR /app

La siguiente instrucción *COPY* copia los envoltorios de Maven y el archivo pom de la máquina host a la imagen del contenedor, "COPY .mvn/ ./mvn" y "COPY mvnw pom.xml ./"

El complemento Spring Boot Maven incluye un objetivo de ejecución que se puede usar para compilar y ejecutar su aplicación rápidamente. La última línea le dice a Docker que compile y ejecute sus paquetes de aplicaciones. "CMD ["./mvnw", "spring-boot:run"]"

**Construyendo su imagen de Docker**

Primero, se ingresa el siguiente comando para comenzar el proceso: docker build --platform linux/amd64 -t spring-helloworld.

Luego, se ejecuta el contenedor Spring Boot Docker

Docker ejecuta procesos en contenedores aislados. Un contenedor es un proceso que se ejecuta en un host, que es local o remoto. Cuando un operador ejecuta docker run, el proceso del contenedor que se ejecuta se aísla con su propio sistema de archivos, red y árbol de procesos separado del host.

El siguiente comando primero crea una capa de contenedor gravable sobre la imagen especificada y luego la inicia: docker run -p 8080:8080 -t spring-helloworld

Mediante el docker desktop vaya a su dashboard y abra su aplicación en su navegador.

En registros puede ver los comportamientos de la aplicación de Spring Boot levantada desde el docker.

Así mismo tiene una ventana de estadísticas la cual permite observar el consumo de diferentes recursos.

**Creación de aplicaciones Spring Boot de varios contenedores con Docker Compose**

Para esta demostración, se aprovechará el popular repositorio *awesome-compose con el comando:* git clone https://github.com/docker/awesome-compose

El archivo de redacción define una aplicación con dos servicios: backend y db. Mientras implementa la aplicación, Docker Compose asigna el puerto 8080 del contenedor de servicios de backend al puerto 8080 del host, según su archivo. Asegúrese de que el puerto 8080 del host no esté en uso.

Para iniciar la aplicación se ejecuta el comando: *docker compose up -d.* Ahora, para listar todos los contenedores ejecutados se usa el comando: *docker compose ps*.

Ademas puede verificar en el localhost:8080

**Detener y retirar sus contenedores**

Para detener o retirar sus contenedores se usa el comando: *docker compose down*