How to Dockerize a Java Application

Dockerizing your Java application can streamline development, testing, and deployment processes, ensuring consistent environments and easy scalability. This guide will walk you through the necessary steps to containerize a Java application using Docker.

Prerequisites

Before we begin, ensure that you have the following:

- Docker installed on your system.
- A Java application ready for containerization.
- Maven to build your project.

Step 1: Install Maven on Ubuntu

Apache Maven is a popular build automation tool used primarily for Java projects. It simplifies dependency management and project configuration through a POM (Project Object Model) file.

Update the package repository index to ensure you have the latest package listings:

sudo apt update

Install Maven from the official Ubuntu repository:

sudo apt install maven -y

Verify the installation by checking the Maven version:

mvn -version

This should display the installed Maven version, confirming the installation was successful.

Step 2: Installing OpenJDK

OpenJDK is the open-source implementation of the Java Platform, which is required for Maven to function correctly.

Update the package repository index once again:

sudo apt update

Install the latest version of OpenJDK with the following command:

sudo apt install default-jdk -y

Verify the installation by checking the Java version:

```
java -version
```

This command confirms that Java is correctly installed on your system.

Step 3: Building the Java Application JAR

Once Java and Maven are set up, the next step is to build your Java application into a JAR file. We'll use a sample Spring Boot project for demonstration purposes.

```
# Clone the sample Java application:
git clone https://github.com/hakanbayraktar/java-spring-petclinic

# Navigate to the project directory:
cd java-spring-petclinic

# Build the project using Maven:
mvn clean install -Dmaven.test.skip=true
```

This command compiles the code, skips the tests, and packages it into a JAR file located in the target directory.

Step 4: Creating a Docker Image for Your Java Application

To containerize the application, we'll create a Docker image. Start by creating a Dockerfile in your project's root directory.

Dockerfile

```
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY /target/*.jar ./java.jar
EXPOSE 8080

CMD ["java", "-jar", "java.jar"]
```

This Dockerfile does the following:

- Uses a pre-built JRE base image to run the Java application.
- Sets the working directory to /app.
- Copies the JAR file from your project's target directory to the container.
- Exposes port 8080 for the application.
- Runs the application using the java -jar command.

Step 5: Building and Running the Docker Container

Build the Docker image with a custom tag:

```
docker build -t java-app:1.0 .
```

This command creates a Docker image with the name java-app and version 1.0.

Run the Docker container and map the container port to your server machine's port:

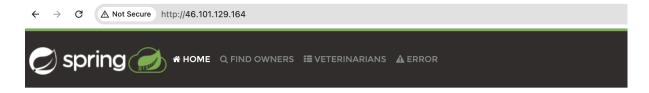
```
docker run -d -p 80:8080 java-app:1.0
```

The -d flag runs the container in detached mode, and -p 80:8080 maps the container's port 8080 to your server's port 80.

Your Java application is now running inside a Docker container and accessible via http://<your-server-ip>.

root@docker:~/spring-petclinic# curl ifconfig.co
46.101.129.164

http://46.101.129.164



Welcome





Conclusion

Containerizing a Java application with Docker simplifies deployment across different environments. This guide has covered the essential steps: setting up Maven and Java, building your application, and creating a Docker image to run your application in a container. With this setup, you can easily scale and manage your Java applications more efficiently.