How to Dockerize Spring Boot Microservices: A Step-by-Step Guide

In the modern software development landscape, microservices architecture has gained significant traction due to its scalability, flexibility, and maintainability.

Docker, a leading containerization platform, complements this architecture by providing isolated environments for each service, facilitating continuous integration and continuous deployment (CI/CD), and ensuring consistent behavior across different environments.

In this article, we will walk through the process of dockerizing microservices, highlighting key concepts, best practices, and practical steps.

What are Microservices?

Microservices architecture involves breaking down a monolithic application into smaller, independent services that can be developed, deployed, and scaled independently. Each microservice is responsible for a specific functionality and communicates with other services over APIs.

Why Docker for Microservices?

Docker containers offer numerous benefits for microservices:

- **Isolation:** Each microservice runs in its own container, ensuring process and resource isolation.
- **Consistency:** Docker containers guarantee that the application behaves the same in development, testing, and production environments.
- **Portability:** Docker images can run on any platform that supports Docker.
- **Scalability:** Containers can be easily scaled up or down based on demand.
- **Simplified CI/CD:** Docker integrates well with CI/CD pipelines, enabling automated builds, tests, and deployments

Prerequisites

- Basic understanding of Docker and Docker Compose.
- A Spring Boot microservices project set up locally.
- Docker installed on your machine. You can download it from the official Docker website.

Steps to Dockerize Spring Boot Microservices

1. **Create a Dockerfile for Each Microservice**

For each Spring Boot microservice, you need to create a `Dockerfile` in the root directory of the project. Here's an example Dockerfile for a Spring Boot application:

For coupon service:

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17-alpine

# Copy the JAR file to the container
ADD target/couponservice-0.0.1-SNAPSHOT.jar couponservice-0.0.1-SNAPSHOT.jar

# Run the application
ENTRYPOINT [ "java","-jar","couponservice-0.0.1-SNAPSHOT.jar" ]
```

For product service:

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17-alpine

# Copy the JAR file to the container
ADD target/productservice-0.0.1-SNAPSHOT.jar productservice-0.0.1-SNAPSHOT.jar

# Run the application
ENTRYPOINT [ "java","-jar","productservice-0.0.1-SNAPSHOT.jar" ]
```

2. Build both microservices through maven using the following commands:

```
mvn clean package -DskipTests
```

3. Build Docker Images

Navigate to the directory containing the Dockerfile and build the Docker image:

```
docker build -f Dockerfile -t coupon_app .
docker build -f Dockerfile -t product_app .
```

4. Set up the MySQL container for database connection.

```
docker run -d -p 6666:3306 --name=docker-mysql --
env="MYSQL_ROOT_PASSWORD=test1234" --env="MYSQL_DATABASE=mydb" mysql
```

5. Run the Docker images using the following commands:

1. Run the MySQL image and execute the queries with single command.

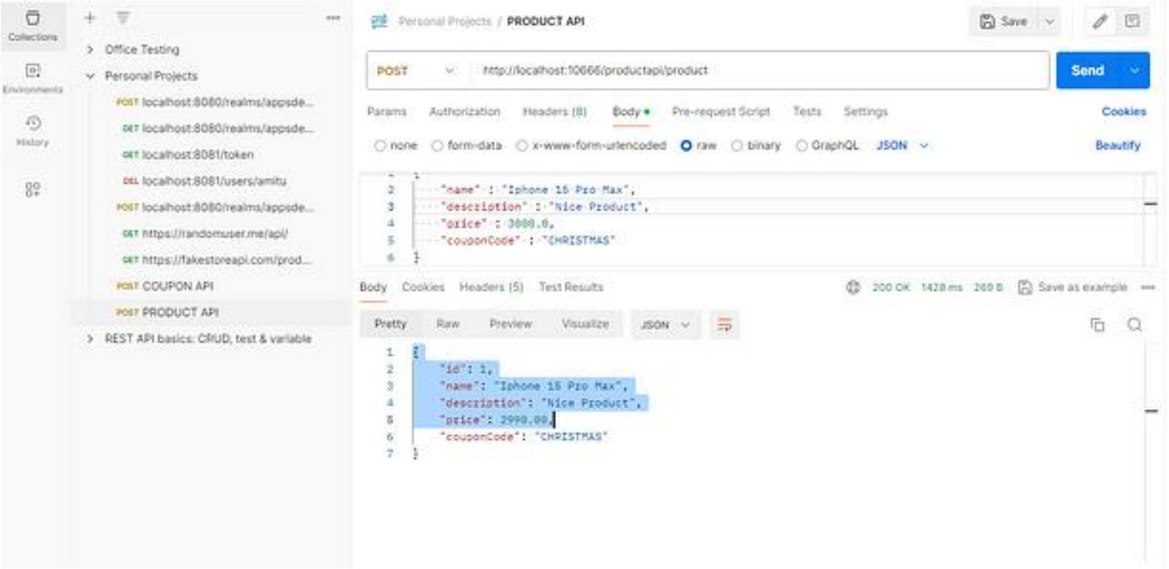
```
docker exec -i docker-mysql mysql -uroot -ptest1234 mydb <tables.sql
```

2. Run both microservices using the following commands.

```
docker run -t --name=coupon-app --link docker-mysql:mysql -p 10555:9091
coupon_app
docker run -t --link docker-mysql:mysql --link coupon-app:coupon_app -p
10666:9090 product_app
```

6. Testing the applications using postman.

You can test the applications using postman by hitting the endpoints



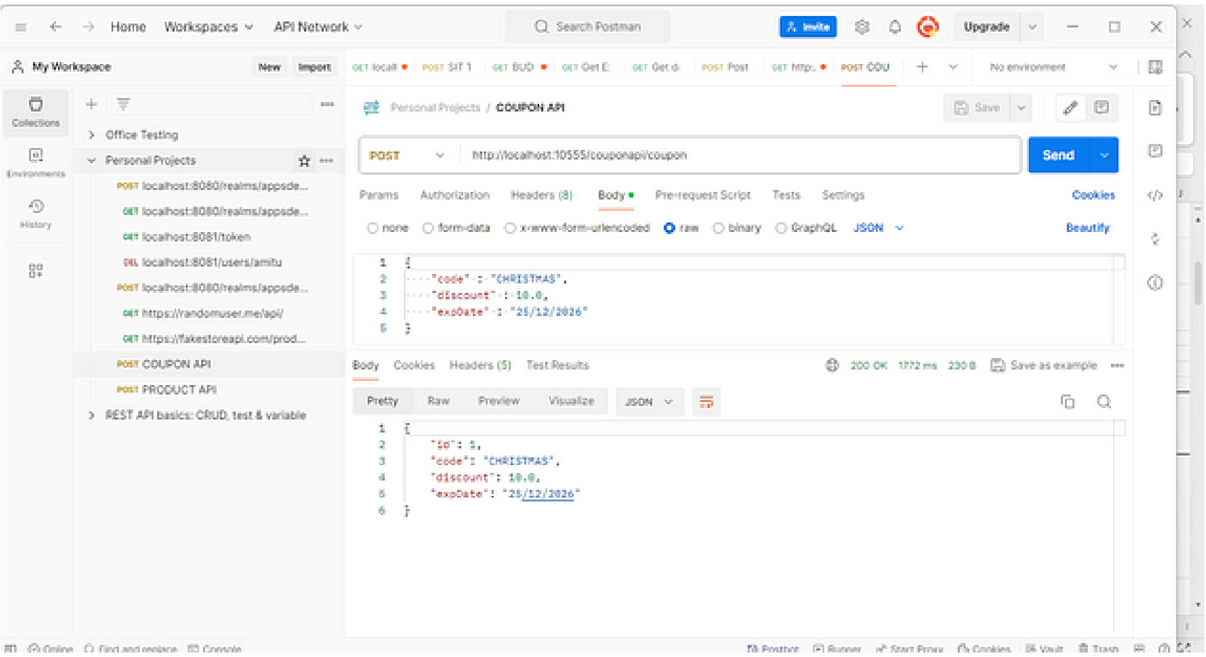
POST http://localhost:10666/productapi/product

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "name" : "Iphone 15 Pro Max",
    "description" : "Nice Product",
    "price" : 3000.0,
    "couponCode" : "CHRISTMAS"
}
```

200 OK 1428 ms 260 B

```
{
    "id": 1,
    "name": "Iphone 15 Pro Max",
    "description": "Nice Product",
    "price": 2990.00,
    "couponCode": "CHRISTMAS"
}
```



POST http://localhost:10555/couponapi/coupon

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "code" : "CHRISTMAS",
    "discount" : 10.0,
    "expDate" : "25/12/2026"
}
```

200 OK 1772 ms 230 B

```
{
    "id": 1,
    "code": "CHRISTMAS",
    "discount": 10.0,
    "expDate": "25/12/2026"
}
```

Source Code : https://github.com/amitu2016/devops_demo

Feel free to reach out or mail me at **abilashs.work@gmail.com**, for help with Interview preparation or resume reviews.

Do follow **Abilash Shankarpalli** for more such posts ✅

Happy Learning!