SPRING BOOT

DOCKER COMPOSE

Rohan Thapa
thaparohan2019@gmail.com

What is Docker Compose?

Docker Compose is a tool for defining and running **multi-container Docker applications**.

It allows you to configure your application's services using a **docker-compose.yml** file and start them all with a single command (**docker-compose up**).

It simplifies managing containers for complex applications that need **multiple services**, like **databases, web servers, or caching systems**.

Why use Docker Compose?

- **Multi-container apps:** Easily manage applications that rely on multiple services (e.g., **web app, database, caching layer**).

- **Isolation:** Each service runs in its own container, ensuring better modularity and isolation.

- **One configuration file:** Centralizes the application configuration in one **YAML file**.

- **Reusability:** You can reuse the configuration in different environments **(development, testing, production).**

In Docker Compose, you can:

- **Define services:** These are your Docker containers.
- **Set up networking:** Each service can communicate with other services on a shared network.
- **Use volumes:** For persistent data storage shared between services or available after a container is destroyed.
- **Scale services:** You can easily scale services horizontally by specifying how many instances of a service you want to run.

Key Concepts in Docker Compose

- **Service:** Defines a container. You can specify details such as the image to use, volumes, ports, and environment variables.

- **Network:** Docker Compose automatically creates networks for services to communicate.

- **Volume:** Persistent data storage shared between your **host** and **container** or between containers.

Docker Compose File Structure

docker-compose.yml

```yaml
version: '3.8'    # Specifies the version of the Docker Compose file format

services:
  app:
    image: springbootapp    # Build from Dockerfile or pull from Docker Hub
    build: ./app            # Build the Docker image from the local directory
    container_name: springboot-container
    ports:
      - "8080:8080"         # Map host port 8080 to container port 8080
    depends_on:
      - db                  # Ensures the 'db' service is started first
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://db:3306/mydb
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: password

  db:
    image: mysql:8.0        # Use official MySQL image
    container_name: mysql-container
    environment:
      MYSQL_ROOT_PASSWORD: password  # Root password for MySQL
      MYSQL_DATABASE: mydb           # Create a database 'mydb'
    ports:
      - "3306:3306"         # Map MySQL port to host
    volumes:
      - db-data:/var/lib/mysql   # Persist MySQL data between restarts

volumes:
  db-data:                # Named volume to persist database data
```

Spring Boot + MySQL with Docker Compose

Create the Spring Boot Application

- First, create a simple Spring Boot application with JPA and MySQL support.

Create Dockerfile for Spring Boot App

In the root of your Spring Boot project, create a Dockerfile:

```
FROM openjdk:22

WORKDIR /app

COPY /target/db-connection.jar /app/app.jar

EXPOSE 8080

ENTRYPOINT ["java" , "-jar" , "app.jar"]
```

Contd..

Create the docker-compose.yaml file

- Place the **docker-compose.yaml** in the root directory of your project, at the same level as the **Dockerfile**.

```yaml
version: "3.8"

services:
  db:
    image: mysql
    container_name: mysql_db_container
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: rohan123
      MYSQL_DATABASE: mysqlbookdb
    volumes:
      - db-data:/var/lib/mysql
    restart: always
    healthcheck:
      test: ["CMD-SHELL" , "mysqladmin ping -h localhost -uroot -prohan123"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 30s

  backend:
    build: .
    container_name: springbootimage
    environment:
      MYSQL_HOST : db
      MYSQL_PORT : 3306
      MYSQL_USERNAME : root
      MYSQL_PASSWORD: rohan123
    ports:
      - "8080:8080"
    restart: always
    depends_on:
      db:
        condition: service_healthy

volumes:
  db-data:
```

Contd..

Build and Run with Docker Compose

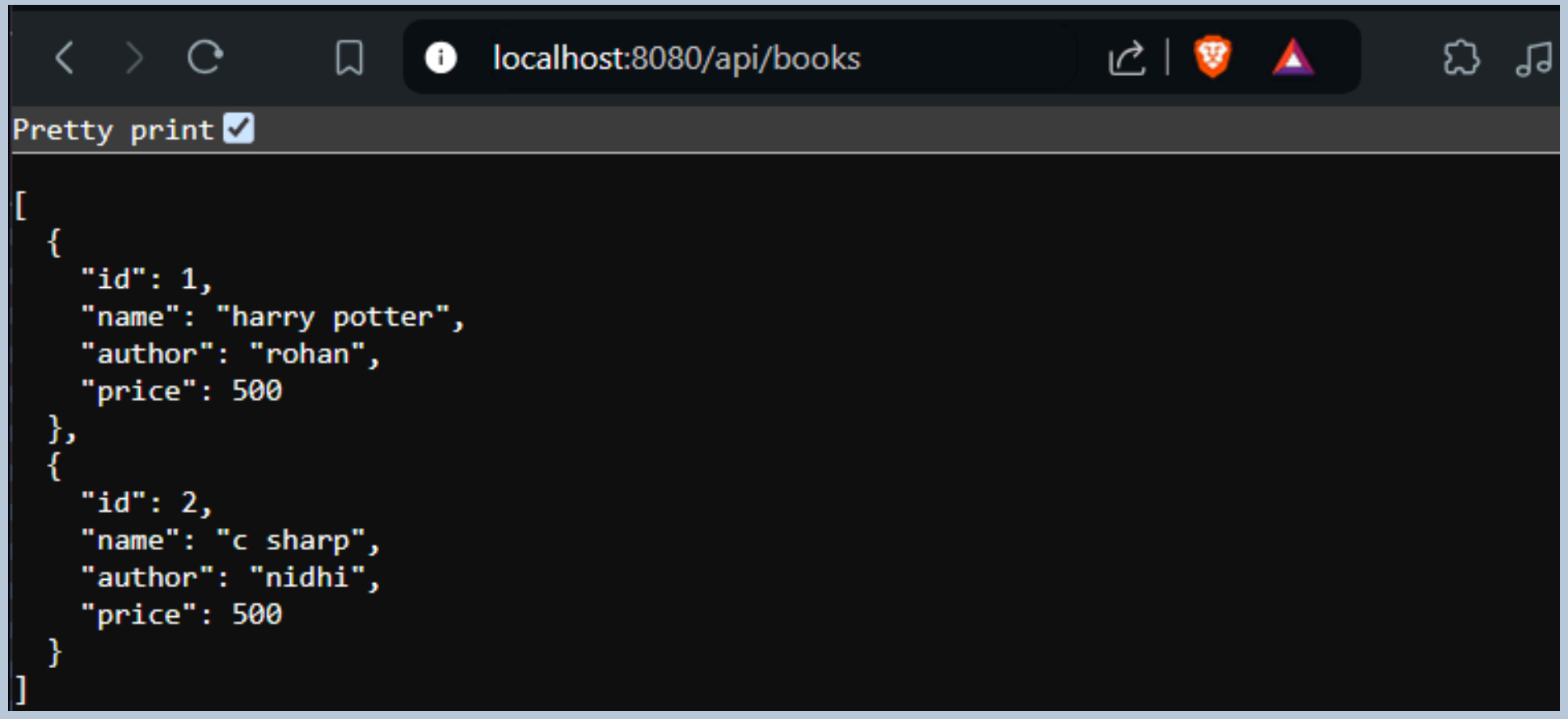
Run the following commands from the root of your project:

```
SQL> docker compose -f .\docker-compose.yaml up
```

```
PS C:\Users\RohanThapa\Desktop\learnjava\Database-Connection-with-MySQL> docker compose -f .\docker-compose.yaml up
time="2024-09-16T16:02:52+05:45" level=warning msg="C:\\Users\\RohanThapa\\Desktop\\learnjava\\Database-Connection-with-MySQL\\docker-compos
[+] Running 2/2
 ✔Container mysql_db_container  Recreated                                                                    0.2s
 ✔Container springbootimage     Recreated                                                                    0.5s
Attaching to mysql_db_container, springbootimage
mysql_db_container  | 2024-09-16 10:17:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 9.0.1-1.el9 started.
mysql_db_container  | 2024-09-16 10:17:54+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql_db_container  | 2024-09-16 10:17:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 9.0.1-1.el9 started.
mysql_db_container  | 2024-09-16 10:17:55+00:00 [Note] [Entrypoint]: Initializing database files
mysql_db_container  | 2024-09-16T10:17:55.144992Z 0 [System] [MY-015017] [Server] MySQL Server Initialization - start.
mysql_db_container  | 2024-09-16T10:17:55.148887Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 9.0.1) initializing of server in
mysql_db_container  | 2024-09-16T10:17:55.204068Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql_db_container  | 2024-09-16T10:17:56.524715Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql_db_container  | 2024-09-16T10:18:00.747428Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please
mysql_db_container  | 2024-09-16T10:18:06.157110Z 0 [System] [MY-015018] [Server] MySQL Server Initialization - end.
mysql_db_container  | 2024-09-16 10:18:06+00:00 [Note] [Entrypoint]: Database files initialized
mysql_db_container  | 2024-09-16 10:18:06+00:00 [Note] [Entrypoint]: Starting temporary server
mysql_db_container  | 2024-09-16T10:18:06.284245Z 0 [System] [MY-015015] [Server] MySQL Server - start.
mysql_db_container  | 2024-09-16T10:18:06.670784Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 9.0.1) starting as process 133
mysql_db_container  | 2024-09-16T10:18:06.701639Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql_db_container  | 2024-09-16T10:18:07.927604Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql_db_container  | 2024-09-16T10:18:08.630227Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql_db_container  | 2024-09-16T10:18:08.630284Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted co
mysql_db_container  | 2024-09-16T10:18:08.642034Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run
 a different directory.
mysql_db_container  | 2024-09-16T10:18:08.693763Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: /var/run/mysqld/my
mysql_db_container  | 2024-09-16T10:18:08.693900Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '9.0.1'
er - GPL.
mysql_db_container  | 2024-09-16 10:18:08+00:00 [Note] [Entrypoint]: Temporary server started.
mysql_db_container  | '/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
mysql_db_container  | Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
mysql_db_container  | Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
mysql_db_container  | Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skipping it.
springbootimage     |
springbootimage     |   .   ____          _            __ _ _
springbootimage     |  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
springbootimage     | ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
springbootimage     |  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
springbootimage     |   '  |____| .__|_| |_|_| |_\__, | / / / /
springbootimage     |  =========|_|==============|___/=/_/_/_/
springbootimage     |
springbootimage     |  :: Spring Boot ::                (v3.3.2)
springbootimage     |
springbootimage     | 2024-09-16T10:18:13.698Z  INFO 1 --- [Database-Connection-with-MySQL] [           main] M.DatabaseConnectionWithMySqlA
NAPSHOT using Java 22 with PID 1 (/app/app.jar started by root in /app)
springbootimage     | 2024-09-16T10:18:13.708Z  INFO 1 --- [Database-Connection-with-MySQL] [           main] M.DatabaseConnectionWithMySqlA
```

Contd..

You should see both the **Spring Boot app** and **MySQL database starting up**. Spring Boot will automatically connect to the MySQL container using the **connection string** in the **environment variables**.



Docker Compose Best Practices

Health Checks: Add health checks for services to ensure that containers are **only started** when their dependencies (**like databases)** are ready.

Example:

```
healthcheck:
  test: ["CMD-SHELL" , "mysqladmin ping -h localhost -uroot -prohan123"]
  interval: 10s
  timeout: 5s
  retries: 5
  start_period: 30s
```

Docker Compose Best Practices

Volumes for Persistence: Always use named volumes for databases and other services that require data persistence across container restarts.

Multi-stage Builds: For complex applications, consider using **multi-stage builds** in your **Dockerfile** to optimize the image size.

Docker Compose Commands

- **docker-compose up:** Starts all services defined in the docker-compose.yml.
- **docker-compose down:** Stops and removes containers, networks, and volumes defined in the file.
- **docker-compose build:** Builds the service images.
- **docker-compose stop:** Stops containers without removing them.
- **docker-compose restart:** Restarts all services.
- **docker-compose logs:** Displays logs for all services.
- **docker-compose exec:** Run commands inside a running container (e.g., docker-compose exec app bash).

Health Checks and Asynchronous Behavior

You can enhance the **docker-compose.yml** with more advanced features like **health checks**, which **wait for MySQL** to be ready, and **asynchronous** Spring Boot scheduling **(as discussed before)**:

Mentioned in the previous example code

When to Use Docker Compose vs. Kubernetes?

Use Docker Compose when:

- You are in the **development stage**.
- You have a **small-scale project** that does not need to run across **multiple nodes** or requires **complex orchestration**.
- You want a **lightweight, quick solution** for deploying applications on a single machine or in small environments.

Use Kubernetes when:

- You need to **manage applications at scale** across multiple nodes.
- You require features like **auto-scaling**, **advanced networking, load balancing,** and rolling updates.
- You are deploying to production in a cloud environment or on a large cluster.

Conclusion

Docker Compose simplifies the **orchestration of multi-container applications**, providing an easy way to manage and configure services.

In our example, we used **Spring Boot** and **MySQL** with **Docker Compose**, allowing for a simple and repeatable environment setup.

Thank You

Rohan Thapa
thaparohan2019@gmail.com