Step by Step Guide to Run MongoDB on Your Local Machine Using Docker

By Hungry Coders

Docker simplifies the process of running applications like MongoDB by creating isolated environments that eliminate compatibility issues. This guide provides a comprehensive step-by-step walkthrough to set up and use MongoDB on your local machine using Docker. It covers essential concepts like Docker Compose, volumes, and container networking.

Why Use Docker for MongoDB?

- **Isolation**: Docker ensures MongoDB runs in a containerized environment, avoiding conflicts with other services or dependencies on your system.
- **Portability**: The same setup can be used across different machines, ensuring consistent behavior.
- **Ease of Use**: Docker Compose simplifies configuration and management of services.
- **Data Persistence**: With volumes, MongoDB data remains intact even if the container stops or restarts.

Step 1: Install Docker

Ensure Docker is installed on your machine. Visit https://www.docker.com/products/docker-desktop for installation instructions. After installation, verify it using `docker --version`.

Step 2: Understanding Docker Compose and Volumes

Docker Compose: It allows you to define and manage multi-container applications using a simple YAML file (`docker-compose.yml`).

Volumes: Volumes are used to persist data generated by a container. Without volumes, data would be lost once the container is stopped or removed. In this guide, the volume maps `mongo-data` on your host to `/data/db` in the container, ensuring MongoDB data is saved persistently.

```
volumes:
  - mongo-data:/data/db
```

Step 3: Docker Compose Configuration

Create a `docker-compose.yml` file with the following configuration to set up MongoDB:

```
services:
  mongodb:
    image: mongo:latest
    container_name: mongodb
    networks:
      - healthcare-network
    ports:
      - "27017:27017"
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=rootpassword
    volumes:
      - mongo-data:/data/db


volumes:
  mongo-data:


networks:
  healthcare-network:
```

Step 4: Start MongoDB

Run the following command in the directory where your `docker-compose.yml` file is located:

`docker-compose up -d`

This will pull the MongoDB image, create the container, and start the MongoDB service.

Step 5: Connect to MongoDB

Use the following commands to connect to MongoDB, authenticate, and interact with collections:

```
docker exec -it mongodb mongosh

use admin

db.auth("root", "rootpassword")

use healthcare

show collections

db.collection_name.find()
```

Step 6: Use MongoDB in Your Application

Configure your application to connect to MongoDB using the following in `application.yml` (for Spring Boot):

```
spring:

  data:

    mongodb:

      uri: mongodb://root:rootpassword@localhost:27017/healthcare
```

Conclusion

By using Docker, you simplify the setup and management of MongoDB while ensuring data persistence and scalability. Docker Compose further enhances the developer experience by streamlining multi-container configurations. This guide equips you with the knowledge to run MongoDB effectively on your local machine.