MongoDB Replica Set with Docker

Table of Contents

Introduction

In this guide, we will create a virtual lab (vLab) using Docker to set up a MongoDB Replica Set. A MongoDB Replica Set is a group of MongoDB servers that maintain the same data set, providing high availability and data redundancy. Docker, a containerization platform, allows us to easily create and manage the MongoDB Replica Set environment.

The vLab setup consists of three MongoDB replica set nodes, a Mongo Express GUI, and a Spring Boot application. The Mongo Express GUI provides a web interface for visualizing and interacting with the data stored in the MongoDB Replica Set. The Spring Boot application communicates with the MongoDB Replica Set to write specific information, which is then presented by Mongo Express.

Docker Compose File

The provided Docker Compose file defines a MongoDB Replica Set with three nodes (`mongo1`, `mongo2`, and `mongo3`). It also includes a Mongo Express container (`mongo-express`) for visualizing the data and a Spring Boot application container (`country-info`) to interact with the replica set.

Replica Set Configuration

Each MongoDB container is started with the `mongo:5` image and configured to run the `mongod` command with the `--replSet rs0` flag, specifying the replica set name as `rs0`. The `--bind_ip localhost,mongoX` option binds the container's IP address to both localhost and its hostname.

Volumes

Volumes in Docker provide a way to persist data even when containers are restarted or removed. In the vLab, each replica set node (`mongo1`, `mongo2`, and `mongo3`) is configured with a separate data volume (`dataMongo01`, `dataMongo02`, and `dataMongo03`) mounted to `/data/db` within the containers. This ensures that the data is stored outside the container's filesystem and remains intact across container lifecycles.

Out of Memory (OOM) Handling

Docker allows you to set resource limits for containers to prevent excessive resource usage. In the vLab, the `mongo1`, `mongo2`, `mongo3`, `mongo-express`, and `country-info` containers have resource limits defined under the `deploy` section. The `limits` specify the maximum amount of memory that each container can use, while the `reservations` indicate the minimum amount of memory that must be available for the container to run. These limits and reservations help prevent the containers from overwhelming the host system's resources, such as running out of memory.

Networking

Docker provides networking capabilities to connect containers and allow them to communicate with each other. In the provided Docker Compose file, all containers (`mongo1`, `mongo2`, `mongo3`, `mongo-express`, and `country-info`) are connected to a custom Docker network named `myNetwork`. This network allows the containers to communicate with each other using their service names as hostnames. For example, the `country-info` container can access the MongoDB replica set nodes using the hostnames `mongo1`, `mongo2`, and `mongo3`.

Usage

To run the vLab, follow these steps:

1. Build the Spring Boot App

Before starting the vLab, you need to build the Spring Boot application JAR file. Make sure you have Maven and Docker installed on your system, and then execute the following command in the root directory of the project:

```
mvn clean install -Dmaven.test.skip
```

This command will compile the source code, and create an executable JAR file in the `target` directory.

Then execute the following command in the root directory of the project in order to build the docker image:

```
./vlab-orchestrator.sh build
```

This command will build a Docker image named `country-info`.

2. Start the Containers

Once the Spring Boot app docker image is built, you can start the vLab containers. Use the following command:

```
./vlab-orchestrator.sh start
```

This command will start the MongoDB replica set containers (`mongo1`, `mongo2`, and `mongo3`), the Mongo Express GUI container (`mongo-express`), and the Spring Boot application container (`country-info`). It will also initialize the replica set if it's not already initialized.

Wait for the MongoDB instances to be ready.

3. Access the vLab

After the containers are started, you can access the vLab components:

Open a web browser and navigate to `http://localhost:8081`. You will be able to visualize and interact with the data stored in the MongoDB Replica Set.

4. Stop the Containers

To stop the vLab containers, use the following command:

```
./vlab-orchestrator.sh stop
```

This command will stop and remove the containers.

That's it! You can now build the Spring Boot app, start the containers, access the vLab components, and stop the containers when you're done.

Conclusion

By using the provided Docker Compose file, you can easily create a MongoDB Replica Set using Docker. The replica set provides data redundancy and high availability, while the additional containers allow you to manage and interact with the data in a convenient manner. Volumes ensure data persistence, resource limits prevent excessive resource usage, and networking enables communication.