MongoDB Replica Set with Docker

Introduction

In this project, we have created 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 Docker Compose file simplifies the process of running multiple containers. It uses a YAML file to specify the services, networks, and volumes required for your application.

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

A MongoDB Replica Set is a distributed database system that provides high availability and data redundancy. It consists of multiple MongoDB server instances, known as nodes, that work together to maintain the same set of data. Replica Sets are designed to ensure data durability, automatic failover, and fault tolerance.

In the provided Docker Compose file, the Replica Set Configuration is defined for the MongoDB containers (mongo1, mongo2, and mongo3). Here's a breakdown of the configuration:

- Image: Each MongoDB container is created using the `mongo:5` image, which corresponds to the MongoDB version 5.
- Command: The containers are configured to run the `mongod` command with specific options. The `mongod` command is the primary process for MongoDB database servers.
- `--replSet rs0`: This flag specifies the replica set name as `rs0`. By setting a common replica set name, the MongoDB nodes can identify each other and communicate.

Volumes

Volumes in Docker are a mechanism for persisting data generated or used by containers. They allow data to be stored outside the container's filesystem, ensuring that it remains intact even

when containers are stopped, restarted, or removed. Volumes provide a convenient way to manage and share data between containers or between a container and the host system.

In the vLab setup, volumes are utilized to ensure data persistence for each MongoDB replica set node (`mongo1`, `mongo2`, and `mongo3`). Specifically, separate data volumes are created for each node: `dataMongo01`, `dataMongo02`, and `dataMongo03`. These volumes are mounted to the `/data/db` directory within the containers, which is the default data directory for MongoDB.

By using separate data volumes for each replica set node, the data remains isolated and independent across the nodes. This means that each replica set node can have its own data stored in its respective volume. The data volumes associated with the nodes allow for efficient storage and retrieval of data, providing a consistent and reliable data storage solution.

Out of Memory (OOM)

Out of Memory (OOM) handling is an important aspect of container management, and Docker provides mechanisms to set resource limits and manage memory usage effectively. In the vLab setup, resource limits are defined for the `mongo1`, `mongo2`, `mongo3`, `mongo-express`, and `country-info` containers to prevent excessive resource consumption and potential memory-related issues.

Resource limits allow you to specify the maximum amount of memory that a container can utilize. By setting these limits, you ensure that containers do not consume an excessive amount of memory, which could lead to performance degradation or even a complete system crash. The resource limits defined in the vLab's Docker Compose file help maintain the stability and availability of the host system.

Additionally, the vLab configuration includes reservations, which indicate the minimum amount of memory that must be available for a container to run. Reservations ensure that there is enough memory allocated for the containers to operate properly without causing resource contention or competing for system resources. This prevents situations where containers are starved of memory and unable to function correctly.

The combination of resource limits and reservations allows for better resource allocation and efficient memory management. It helps prevent containers from overwhelming the host system's resources, such as running out of memory, which could result in crashes or downtime.

Networking

Networking in Docker is a powerful feature that enables containers to communicate with each other and with external networks. In the Docker Compose file for the vLab setup, a custom Docker network named `myNetwork` is created, and all containers (`mongo1`, `mongo2`, `mongo3`, `mongo-express`, and `country-info`) are connected to this network.

By connecting the containers to the same network, they can communicate with each other using their service names as hostnames. This simplifies the process of establishing connections between containers and allows them to interact seamlessly. For example, the `country-info` container can easily access the MongoDB replica set nodes using the hostnames `mongo1`, `mongo2`, and `mongo3`.

Networking in Docker provides isolation and security by creating a separate network for the containers. This ensures that containers within the network can communicate with each other, but they are isolated from external networks by default. This isolation prevents unwanted access and enhances the security of the vLab environment.

Docker networking supports various networking modes, including bridge networks, overlay networks, and host networks. Bridge networks, which are used in the provided Docker Compose file, are the most commonly used type of network in Docker. They create an internal network that allows containers to communicate with each other while remaining isolated from the host system and external networks.

Overall, Docker's networking capabilities simplify the process of connecting and communicating between containers, providing a flexible and scalable networking solution for containerized applications. It enhances the collaboration and integration between different components within the vLab environment and allows for seamless communication between containers using intuitive service names as hostnames.

Country Info Application

The Country Info application in the vLab serves as a demonstration of how to interact with the MongoDB Replica Set. Its primary function is to take a JSON file containing country information and write this data into the MongoDB Replica Set.

Here are some key points about the Spring Boot application:

Purpose: The Country Info app showcases the integration of a Java-based application with the MongoDB Replica Set. It demonstrates how to establish a connection to the replica set, and perform data operations.

Technology Stack: The Country Info app is developed using the Spring Boot framework, which simplifies the process of building Java applications, and it utilizes Java 17.

Connecting to the Replica Set: The Spring Boot application establishes a connection to the MongoDB Replica Set using the Spring Data MongoDB. It configures the connection details, including the replica set name, hostnames, and port numbers, to establish communication with the replica set nodes.

Data Ingestion: The Spring Boot app reads the provided JSON file containing country information. It parses the JSON data and maps it to the Country Java object using the Jackson library. The application then writes this data into the MongoDB Replica Set.

Overall, the Country Info application serves as a practical example of integrating a Java-based application with a MongoDB Replica Set. It demonstrates how to establish connections, and perform data operations, providing a hands-on demonstration of working with replica sets using Spring Boot and Java 17.

Mongo Express GUI

Mongo Express serves as a user-friendly web interface for MongoDB, allowing easy visualization and interaction with the data stored in the MongoDB Replica Set. In the context of this vLab, its purpose is to provide a beautiful and intuitive way to demonstrate the country information written by the Spring Boot application. By accessing Mongo Express, users can visually verify that the country data has been successfully written to the replica set, ensuring the proper functioning of the replica set. With its user-friendly interface, Mongo Express enables effortless browsing, querying, and manipulation of the data, making it easy to explore and understand the stored country information.

Usage

To run the vLab, follow these steps:

1. Build the Country Info Application

Before starting the vLab, you need to build the Country Info 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.