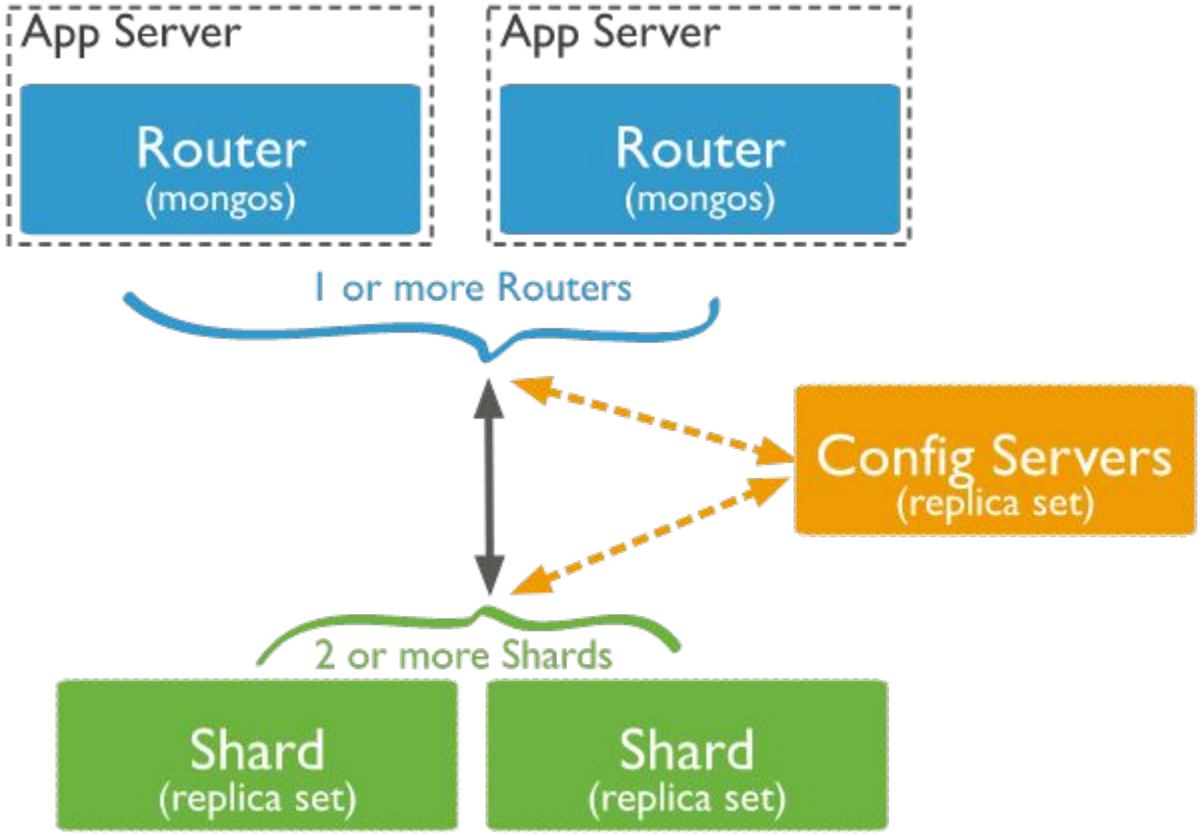
DBS101 Database Systems Fundamentals

Lesson 16

Learning Outcomes

1. Implement database sharding
2. Understand access controls in mongodb
3. Apply access controls to leverage user permissions.
4. Understand database logging.

Sharding in Mongodb



Create directories to store mongodb files

Linux systems:

```
mkdir -pv  mongodb/data/configdb/
```

```
mkdir -pv  mongodb/data/logs
```

```
touch mongodb/data/logs/configsvr.log
```

```
ls -alRv mongodb/
```

Create directories to store mongodb files

Mac systems(Installed mongdb using brew):

```
mkdir -pv opt/homebrew/etc/mongodb

mkdir -pv opt/homebrew/mongodb/data

mkdir -pv opt/homebrew/mongodb/data/configdb

mkdir -pv opt/homebrew/mongodb/data/logs

touch opt/homebrew/mongodb/data/logs/configsvr.log
```

Create directories to store mongodb files

What are we doing?

Creating directories to support the new database instance

Create directories to store mongodb files

```
sudo nano /etc/mongodConfig.conf
```

- Configuration file for the new database instance.
- Set up the instance to save all logs in the configsvr created above.

Create directories to store mongodb files

Add the following lines of code to your configuration file

```
storage:
  dbPath: path to your configdb directory
  journal:
enabled: true
systemLog:
  destination: file
  logAppend: true
  path: path to your configsvr.log file
net:
  port: 27019
  bindIp: 127.0.0.1
sharding:
  clusterRole: configsvr
replication:
  replSetName: ConfigReplSet
```

Start new mongodb instance

```
mongod --config /etc/mongodConfig.conf&
```

Check logs to see if server is running:

```
tail -100 mongodb/data/logs/configsvr.log
```

Connect to shell:

```
mongo 127.0.0.1:27019
```

Start new mongodb instance

Inside the shell use the following commands to initiate the server instance with the default options:

```
rs.initiate()
```

```
rs.status()
```

Start new mongodb instance

Inside the shell use the following commands to initiate the server instance with the default options:

```
rs.initiate()
```

```
rs.status()
```

Query Router

Query routing is a process of directing user queries to appropriate servers by constraining the search space through query refinement and source selection.

The goal of a query routing middleware is to reduce both false positives (useless answers delivered that fail to fulfill user's needs) and false negatives (useful answers that the system fails to deliver to the user) in answering a query.

Query Router

Effective query routing not only minimizes the query response time and the overall processing cost, but also eliminates a lot of unnecessary communication overhead over the global networks and over the individual information sources.

Query Router

The **mongos** instances provide the interface between the client applications and the sharded cluster.

The mongos instances route queries and write operations to the shards.

Query Router

To use mongos to connect shards:

1. Create log file for mongos

Linux Commands:

```
mkdir -pv  mongodb/data/logs
```

```
touch mongodb/data/logs/mongorouter.log
ls -alRv mongodb/
```

Linux Mac Commands(Installation of mongo through brew):

```
touch /opt/homebrew/etc/mongodb/data/logs/mongorouter.log
```

Query Router

To use mongos to connect shards:

2. Create config file for mongos

```
sudo nano /etc/mongoRouter.conf
```

Query Router

To use mongos to connect shards:
2. Create config file for mongos

- Add the following lines of code

```
systemLog:
  destination: file
  logAppend: true
  path: /home/barry/mongodb/data/logs/queryrouter.log
net:
  port: 27018
  bindIp: 127.0.0.1
sharding:
  configDB: ConfigReplSet/127.0.0.1:27019
```

Query Router

To use mongos to connect shards:

3. Start query router with mongos command:

```
mongos --config /etc/mongoRouter.conf&
```

4. Check if mongos instance is reachable:

```
mongo 127.0.0.1:27018
```

Configuring Sharded Servers

1. Create a directory to store data for the shard server

Linux commands:

```
mkdir -pv  mongodb/data/sharddb/
mkdir -pv  mongodb/data/logs
touch mongodb/data/logs/shard.log
ls -alRv mongodb/
```

Mac commands(Installation of mongo through brew)

```
mkdir -pv  /opt/homebrew/etc/mongodb/data/sharddb/
touch /opt/homebrew/etc/mongodb/data/logs/shard.log
```

Configuring Sharded Servers

2. Create a config file

```
sudo nano /etc/mongodShard.conf
```

Add the following lines of code:

```
storage:
  dbPath: path to sharddb diretory
  journal:
    enabled: true
systemLog:
  destination: file
  logAppend: true
  path: path to shard.log file
net:
  port: 27015
  bindIp: 127.0.0.1
sharding:
  clusterRole: shardsvr
replication:
replSetName: ShardReplSet
```

Configuring Sharded Servers

3. Start shard server

```
mongod --config /etc/mongodShard.conf&
```

```
 -  Check logs to see if the server is running:
```

```
tail -100 mongodb/data/logs/shard.log
```

Configuring Sharded Servers

3. Initialize server

Login to server:

```
mongo 127.0.0.1:27015
```

Initiate the server:

```
rs.initiate()
```

```
rs.status()
```

Configuring Sharded Servers

4. Adding the shard to the cluster

Login to mongos(Query Processor):

```
mongo 127.0.0.1:27018
```

Add shard to cluster:- provide the replSetName with the IP address and the port of the shard instance.

```
sh.addShard( "ShardReplSet/127.0.0.1:27015")
```

Configuring Sharding on a Database

1. Create database “student”:

```
use student
```

2. Enable sharding student

```
sh.enableSharding("student")
```

3. Check sharding status:

```
sh.status()
```

Configuring Sharding on a Database

4. Create a collection to be sharded:

```
db.createCollection("student_info")
```

5. Create index and a record. Create index with student_id in descending order.

```
db.student_info.createIndex({studentid: -1})
```

Configuring Sharding on a Database

4. Enable sharding for the collection created above.

Note: Sharding cannot be enabled without the shard key being a hash value

To check if student_id is a hashed value:

```
db.student_info.ensureIndex({studentid : "hashed"})
```

Configuring Sharding on a Database

4. Enable sharding for the collection created above.

```
sh.shardCollection("student.student_info", {studentid : "hashed"})
```

To verify that the sharding is working properly:

```
db.student_info.getShardDistribution()
```

Also, check in sharded server if the database student has been created.

Access Controls in MongoDB

Mongodb has Role-Based Access Control (RBAC) to govern access to a MongoDB system.

A user is granted one or more roles that determine the user's access to database resources and operations. Outside of role assignments, the user has no access to the system.

To know more about this visit the link: RBAC Mongodb

Access Controls in MongoDB

Creating users in MongoDB

1. Switch to admin db:

```
use admin
```

2. Add the user with either the userAdmin role or userAdminAnyDatabase role, and only that role, by issuing a command similar to the following, where <username> is the username and <password> is the password:

Access Controls in MongoDB

Creating users in MongoDB

```
db.addUser( { user: "userdemo",
              pwd: "12345",
              roles: [ "userAdminAnyDatabase" ] } )
```

Access Controls in MongoDB

Adding a user to a database

To add a user to a database you must authenticate to that database as a user with the userAdmin or userAdminAnyDatabase role.

Access Controls in MongoDB

Adding a user to a database

```
use analytics
db.addUser( { user: "userdemo",
              pwd: "12345",
              roles: [ "read", "dbAdmin" ]
            } )
```

Access Controls in MongoDB

Get all users in the server:

```
db.getUsers()
```

To change user in shell:

```
db.auth(“userdemo”, passwordPrompt())
```

To change access shell as a user:

```
mongosh --port 27017  --authenticationDatabase \
    "userdemo" -u "12345" -p
```

Database logging in MongoDB

MongoDB maintains a running log of events, including entries such as incoming connections, commands run, and issues encountered.

Generally, log messages are useful for diagnosing issues, monitoring your deployment, and tuning performance.

Database logging in MongoDB

To retrieve the latest logs:

```
db.adminCommand( { getLog : "global" } )
```

Note: getLog only shows the most recent 1024 logged mongod events, and is not a replacement for the MongoDB log file.

RECAP

References

Wickramasinghe, S. (n.d.). *MongoDB Sharding: Concepts, Examples & Tutorials*. BMC Blogs. https://www.bmc.com/blogs/mongodb-sharding-explained/