**MongoDB – Session – Day 4 & 5**

### Setting Up a 3-Node MongoDB Replica Set and Shard Cluster on Ubuntu 18.04 with MongoDB 4.x

Here's a step-by-step guide to set up a 3-node MongoDB replica set and shard cluster on Ubuntu 18.04.

### Prerequisites

1. **Three Ubuntu 18.04 machines**: Ensure each machine has MongoDB 4.x installed.
2. **Network Configuration**: All machines should be able to communicate with each other.

### Step 1: Install MongoDB on Each Node – Skip to Step 2 if already done

Install MongoDB on each node following these steps:

# Import the public key used by the package management system

wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -

# Create a list file for MongoDB

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list

# Reload local package database

sudo apt-get update

# Install MongoDB packages

sudo apt-get install -y mongodb-org

### Step 2: Configure MongoDB Replica Set

1. **Edit the MongoDB configuration file on each node**:

sudo nano /etc/mongod.conf

1. **Update the following lines**:

# network interfaces

network:

port: 27017

bindIp: 0.0.0.0 # This ensures MongoDB listens on all interfaces

# replication settings

replication:

replSetName: "rs0" # Name of the replica set

1. **Start MongoDB service on each node**:

sudo systemctl start mongod

sudo systemctl enable mongod

1. **Initiate the replica set on the primary node**: - first time

Mongo/sh

Within the MongoDB shell, run:

rs.initiate({

\_id: "rs0",

members: [

{ \_id: 0, host: "node1:27017" },

{ \_id: 1, host: "node2:27017" },

{ \_id: 2, host: "node3:27017" }

]

})

### Step 3: Verify the Replica Set

Check the status of the replica set on the primary node:

rs.status()

### Step 4: Configure Shard Cluster

1. **Edit the configuration file on each node to act as a shard**:

sudo nano /etc/mongod.conf

Update the configuration to include:

sharding:

clusterRole: "shardsvr"

1. **Restart MongoDB service on each shard node**:

sudo systemctl restart mongod

1. **Setup the Config Servers**:

Choose three nodes to act as config servers. Configure and start MongoDB with config server role on these nodes.

Edit /etc/mongod.conf on each config server:

sharding:

clusterRole: "configsvr"

Restart the MongoDB service on each config server:

sudo systemctl restart mongod

1. **Setup the Mongos Router**:

Install mongos on a separate machine or one of the existing machines:

sudo apt-get install -y mongodb-org-mongos

Edit or create the configuration file /etc/mongos.conf:

# network interfaces

net:

port: 27017

bindIp: 0.0.0.0

# sharding configuration servers

sharding:

configDB: rs0/node1:27017,node2:27017,node3:27017

Start the mongos service:

sudo systemctl start mongos

sudo systemctl enable mongos

1. **Add Shards to the Cluster**:

Connect to the mongos instance:

mongo --host <mongos\_host>:27017

Add the shards:

sh.addShard("rs0/node1:27017,node2:27017,node3:27017")

### Step 5: Verify the Shard Cluster

Check the shard status:

sh.status()

### Example: Insert Data into the Sharded Cluster

1. **Enable Sharding for a Database**:

sh.enableSharding("testDB")

1. **Shard a Collection**:

Choose a shard key for the collection and shard it:

sh.shardCollection("testDB.testCollection", { shardKey: 1 })

1. **Insert Data**:

use testDB

for (let i = 0; i < 10000; i++) {

db.testCollection.insert({ shardKey: i, data: "example" + i })

}

1. **Verify Data Distribution**:

db.testCollection.getShardDistribution()

This completes the setup of a 3-node MongoDB replica set and shard cluster on Ubuntu 18.04 with MongoDB 4.x.

Upgrading a 3-node MongoDB 4.x replica set on Ubuntu 18.04 to a sharded cluster involves configuring the current replica set to act as shards, setting up config servers, and adding a mongos router. Here's a step-by-step guide:

### Prerequisites

* Three Ubuntu 18.04 machines with MongoDB 4.x installed.
* A working 3-node MongoDB replica set.

### Step 1: Backup Your Data

Before making any major changes, ensure you have a backup of your data. You can use mongodump to create a backup.

mongodump --out /path/to/backup/directory

### Step 2: Configure the Replica Set Nodes as Shards

1. **Edit the MongoDB configuration file on each replica set node**:

sudo nano /etc/mongod.conf

1. **Update the configuration to include shard settings**:

sharding:

clusterRole: "shardsvr"

1. **Restart MongoDB service on each node**:

sudo systemctl restart mongod

### Step 3: Setup Config Servers

Choose three nodes to act as config servers. These can be new nodes or the existing replica set nodes. For simplicity, let's assume you use new nodes.

1. **Install MongoDB on each config server node** if not already installed:

# Import the public key used by the package management system

wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -

# Create a list file for MongoDB

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list

# Reload local package database

sudo apt-get update

# Install MongoDB packages

sudo apt-get install -y mongodb-org

1. **Edit the configuration file on each config server node**:

sudo nano /etc/mongod.conf

Update the configuration to include:

sharding:

clusterRole: "configsvr"

rs.initiate() – to be done …to start replicate set

1. **Restart MongoDB service on each config server node**:

sudo systemctl restart mongod

### Step 4: Setup the Mongos Router

Install mongos on a separate machine or one of the existing machines:

sudo apt-get install -y mongodb-org-mongos

Edit or create the configuration file /etc/mongos.conf:

# network interfaces

net:

port: 27017

bindIp: 0.0.0.0

# sharding configuration servers

sharding:

configDB: rs0/config1:27019,config2:27019,config3:27019

Start the mongos service:

sudo systemctl start mongos

sudo systemctl enable mongos

### Step 5: Add Shards to the Cluster

Connect to the mongos instance:

mongo --host <mongos\_host>:27017

Add the shards (assuming your replica set is named rs0):

sh.addShard("rs0/node1:27017,node2:27017,node3:27017")

### Step 6: Verify the Shard Cluster

Check the shard status:

sh.status()

### Step 7: Enable Sharding for a Database

1. **Enable Sharding for a Database**:

sh.enableSharding("yourDatabase")

1. **Shard a Collection**:

Choose a shard key for the collection and shard it:

sh.shardCollection("yourDatabase.yourCollection", { shardKey: 1 })

1. **Insert Data**:

for (let i = 0; i < 10000; i++) {

db.yourCollection.insert({ shardKey: i, data: "example" + i })

}

1. **Verify Data Distribution**:

db.yourCollection.getShardDistribution()

**Setting up a MongoDB replica set and sharded cluster using *Three* Ubuntu 18.04 servers involves several steps.**

### Prerequisites

1. Three Ubuntu 18.04 servers with MongoDB 4.x installed.
2. Network connectivity between the servers.
3. Root or sudo access to all servers.

### Step-by-Step Guide

#### 1. Install MongoDB on All Servers

Ensure MongoDB is installed on all three servers. If not, install it using the following commands on each server:

sudo apt update

sudo apt install -y mongodb

#### 2. Configure Replica Set

##### On each server, edit the MongoDB configuration file (/etc/mongodb.conf or /etc/mongod.conf):

net:

bindIp: 0.0.0.0 # Listen on all interfaces

port: 27017

replication:

replSetName: "rs0" # Replica set name

##### Start MongoDB on each server:

sudo systemctl start mongod

sudo systemctl enable mongod

##### Initialize the Replica Set

1. On the primary node (e.g., server1), connect to MongoDB shell:

mongo

1. Initiate the replica set:

rs.initiate({

\_id: "rs0",

members: [

{ \_id: 0, host: "server1:27017" },

{ \_id: 1, host: "server2:27017" },

{ \_id: 2, host: "server3:27017" }

]

})

1. Check the replica set status:

rs.status()

#### 3. Configure Sharding

##### Configure a Config Server Replica Set

1. Choose one server to host the config servers, and start three mongod instances with config server roles.

Create directories for the config server data on server1:

mkdir -p /data/configdb1 /data/configdb2 /data/configdb3

1. Start the config server instances:

mongod --configsvr --replSet configReplSet --port 27019 --dbpath /data/configdb1 --bind\_ip\_all --fork --logpath /var/log/mongodb/configdb1.log

mongod --configsvr --replSet configReplSet --port 27020 --dbpath /data/configdb2 --bind\_ip\_all --fork --logpath /var/log/mongodb/configdb2.log

mongod --configsvr --replSet configReplSet --port 27021 --dbpath /data/configdb3 --bind\_ip\_all --fork --logpath /var/log/mongodb/configdb3.log

1. Connect to one of the config server instances and initiate the replica set:

mongo --port 27019

rs.initiate({

\_id: "configReplSet",

configsvr: true,

members: [

{ \_id: 0, host: "server1:27019" },

{ \_id: 1, host: "server1:27020" },

{ \_id: 2, host: "server1:27021" }

]

})

#### 4. Configure Shards

1. On each of the three servers, start a mongod instance for the shard:

mkdir -p /data/sharddb

mongod --shardsvr --replSet rs0 --port 27018 --dbpath /data/sharddb --bind\_ip\_all --fork --logpath /var/log/mongodb/sharddb.log

1. Add the shards to the cluster:

mongo --port 27017

sh.addShard("rs0/server1:27018")

sh.addShard("rs0/server2:27018")

sh.addShard("rs0/server3:27018")

#### 5. Configure Mongos Router

1. Choose one server to host the mongos instance. Start mongos and point it to the config server replica set:

mongos --configdb configReplSet/server1:27019,server1:27020,server1:27021 --bind\_ip\_all --fork --logpath /var/log/mongodb/mongos.log

#### 6. Enable Sharding for a Database

1. Connect to the mongos instance:

mongo --port 27017

1. Enable sharding for a specific database:

sh.enableSharding("myDatabase")

1. Shard a specific collection:

sh.shardCollection("myDatabase.myCollection", { shardKey: 1 })

### Summary

You have set up a MongoDB replica set and configured it as a sharded cluster using three Ubuntu 18.04 servers. This setup involves configuring network interfaces, initiating the replica set, setting up config servers, adding shards, and configuring a mongos router to handle client requests. This architecture ensures high availability and scalability for your MongoDB deployment.

<https://www.mongodb.com/blog/post/demystifying-sharding-mongodb>

<https://www.digitalocean.com/community/tutorials/how-to-use-sharding-in-mongodb>

<https://www.analyticsvidhya.com/blog/2022/12/mongodb-replication-and-sharding-a-complete-introduction/>