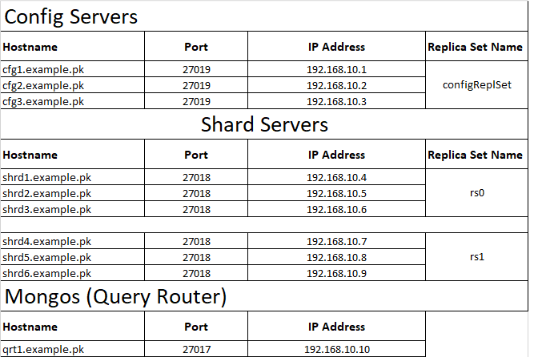
## **How to Setup MongoDB Sharding with Replicas.**

Graphical user interface, diagram

Description automatically generated

We will use the following setup for setting up MongoDB Sharding:

We want 10 servers:



## **Step-1: Install MongoDB on all Servers**

First you need to update and upgrade your systems repository in order to install MongoDB. Type the following command in your terminal and then press Enter.

|  |
| --- |
| sudo apt update && sudo apt upgrade -y |

First, you will need to install the [MongoDB community server](https://cloudinfrastructureservices.co.uk/how-to-setup-mongodb-community-server-on-azure-aws/) package on all 10 servers. Perform the below steps on all servers to install the MongoDB community server:

First, run the following command to install all required dependencies.

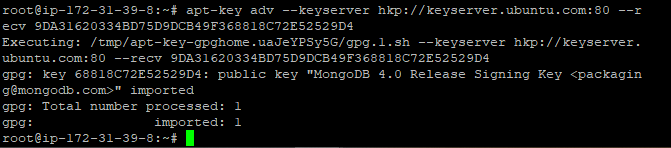
|  |
| --- |
| apt-get install gnupg2 wget tree -y |

Next, import the MongoDB GPG key using the command below:

(The MongoDB release team digitally signs all software packages to certify that a particular MongoDB package is a valid and unaltered MongoDB release. Before installing MongoDB, you should validate the package using either the provided GPG signature or SHA-256 checksum.)

|  |
| --- |
| apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4 |

Sample output:



Next, add the MongoDB repository to APT source file:

|  |
| --- |
| echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0 multiverse" | tee /etc/apt/sources.list.d/mongodb-org.list |

Next, update the repository and install the MongoDB server package:

|  |
| --- |
| apt-get update -y  apt-get install mongodb-org -y |

Once the MongoDB server package is installed, verify the MongoDB version using the following command:

|  |
| --- |
| mongo --version |

## **Step-2: Configure Config Servers (run the commands in 3 configuration servers)**

In this section, we will configure the config server to be a replica set. First, create a directory structure using the following command:

|  |
| --- |
| mkdir -p /mongodb-config/data/configdb  mkdir -p /mongodb-config/data/logs  touch /mongodb-config/data/logs/configsvr.log |

You can now verify your directory structure using the following command:

|  |
| --- |
| tree /mongodb-config |

Sample output:

Graphical user interface, text

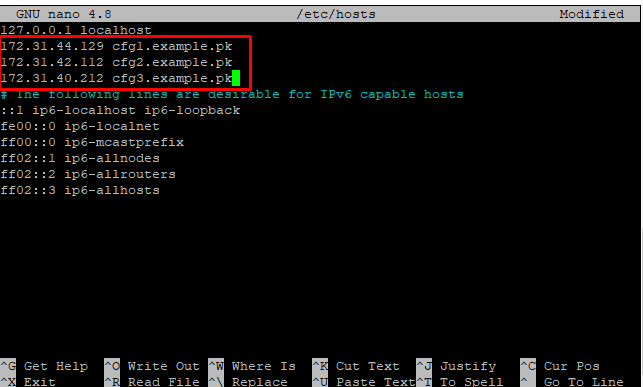
Description automatically generated

On Linux and other Unix-like systems, hosts is stored in the /etc/ directory. On **each of your three servers**, edit the file with your preferred text editor. Here, we’ll use nano:

|  |
| --- |
| sudo nano /etc/hosts |

|  |
| --- |
| **172.31.11.162** cfg1.example.pk  **172.31.6.229** cfg2.example.pk  **172.31.1.137** cfg3.example.pk |

Sample output:



Next save and exit.

## **Updating Each Server’s Firewall Configurations with UFW**

However, these firewalls will also block the MongoDB instances on each server from communicating with one another, preventing you from initiating the replica set. To correct this, you’ll need to add new firewall rules to allow each server access to the port on the other two servers on which MongoDB is listening for connections.

On **config1**, run the following ufw command to allow **config2** access to port 27019 on **config3**:

Bellow commands run in the Cfg1-server:

|  |
| --- |
| sudo ufw allow from **cfg2\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **cfg3\_server\_private\_ip** to any port 27019 |

Bellow commands run in the Cfg2-server:

|  |
| --- |
| sudo ufw allow from **cfg1\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **cfg3\_server\_private\_ip** to any port 27019 |

Bellow commands run in the Cfg3-server:

|  |
| --- |
| sudo ufw allow from **cfg1\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **cfg2\_server\_private\_ip** to any port 27019 |

Next, create a new configuration file for three Config Servers using your favorite editor:

|  |
| --- |
| nano /etc/mongodConfig.conf |

Define your storage path, logging path, port, server IP, cluster role, and replica set as shown below (copy and paste below lines):

|  |
| --- |
| storage:  dbPath: /mongodb-config/data/configdb  journal:  enabled: true  systemLog:  destination: file  logAppend: true  path: /mongodb-config/data/logs/configsvr.log  net:  port: 27019  bindIp: **(config\_server\_private\_ip)**  sharding:  clusterRole: configsvr  replication:  replSetName: configReplSet |

Sample output:Text

Description automatically generated

Next click ctrl+x,y enter

Save and close the file once you are finished. Next, start the three config servers using the following command:

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

Next, check the three config servers listening port with the following command:

|  |
| --- |
| ss -antpl | grep 27019 |

Sample output:

Text

Description automatically generated

## **Starting the Config Replica Set and Adding Members**

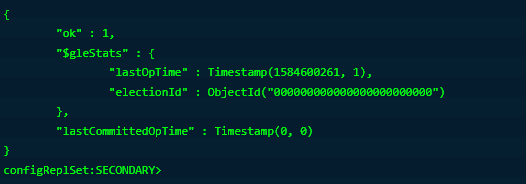
Next, connect to the three config servers using the following command:

|  |
| --- |
| mongo **(config\_server\_private\_ip)**:27019 |

Bellow commands run in the Cfg1-server:

|  |
| --- |
| rs.initiate({ \_id: "configReplSet", configsvr: true, members: [{ \_id : 0, host : "**cfg1\_server\_private\_ip**:27019"},{ \_id : 1, host : "**cfg2\_server\_private\_ip**:27019"},{ \_id : 2, host : "**cfg3\_server\_private\_ip**:27019"}]}) |

Sample output:



Notice that the MongoDB shell prompt has also changed to configReplSet:SECONDARY&gt; or configReplSet:PRIMARY>.

# To make sure that each config server has been added to the replica set, type below on mongo shell (cfg1-server):

|  |
| --- |
| rs.config() |

If the replica set has been configured properly, you’ll see output similar to the following:

Graphical user interface, text

Description automatically generated

For maximum replica set configuration information, type below in three servers(config-1, config-2,config-3):

|  |
| --- |
| rs.status() |

## **Step-3: Configure Shard-1 replica set and Shard-2 replica set.**

On Linux and other Unix-like systems, hosts is stored in the /etc/ directory. On **each of your three shard-“rs0” servers**, edit the file with your preferred text editor. Here, we’ll use nano:

|  |
| --- |
| sudo nano /etc/hosts |

|  |
| --- |
| **172.31.45.170** shrd1.example.pk  **172.31.46.22** shrd2.example.pk  **172.31.46.214** shrd3.example.pk |

Observe bellow image:

Text

Description automatically generated

Control x ,y enters.

rules to allow each server access to the port on the other two servers on which MongoDB is listening for connections.

On **shard1**, run the following ufw command to allow **shard2** access to port 27019 on **shard3**:

Bellow commands run in the Shard1-server:

|  |
| --- |
| sudo ufw allow from **shard2\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **Shard3\_server\_private\_ip** to any port 27019 |

Bellow commands run in the shard2-server:

|  |
| --- |
| sudo ufw allow from **shard1\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **shard3\_server\_private\_ip** to any port 27019 |

Bellow commands run in the Shard3-server:

|  |
| --- |
| sudo ufw allow from **shard1\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **shard2\_server\_private\_ip** to any port 27019 |

On Linux and other Unix-like systems, hosts is stored in the /etc/ directory. On **each of your three shard-“rs1” servers**, edit the file with your preferred text editor. Here, we’ll use nano:

|  |
| --- |
| sudo nano /etc/hosts |

|  |
| --- |
| **172.31.10.154** shrd4.example.pk  **172.31.1.70** shrd5.example.pk  **172.31.12.66** shrd6.example.pk |

rules to allow each server access to the port on the other two servers on which MongoDB is listening for connections.

On **shard4**, run the following ufw command to allow **shard5** access to port 27019 on **shard6**:

Bellow commands run in the **Shard4**-server:

|  |
| --- |
| sudo ufw allow from **shard5\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **Shard6\_server\_private\_ip** to any port 27019 |

Bellow commands run in the shard5-server:

|  |
| --- |
| sudo ufw allow from **shard4\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **shard6\_server\_private\_ip** to any port 27019 |

Bellow commands run in the Shard6-server:

|  |
| --- |
| sudo ufw allow from **shard4\_server\_private\_ip** to any port 27019 |

|  |
| --- |
| sudo ufw allow from **shard5\_server\_private\_ip** to any port 27019 |

Next, log in to Shard servers and create a directory structure with the following command (execute in two replica sets(6) servers Shard-“rs0” and Shard-“rs1”):

|  |
| --- |
| mkdir -p /mongodb-config/data/sharddb/  mkdir -p /mongodb-config/data/logs  touch /mongodb-config/data/logs/shard.log |

Next, create a new configuration file for Shards with the following command (execute in two replica sets (6)servers Shard-”rs0”and Shard-“rs1”):

|  |
| --- |
| nano /etc/mongodShard.conf |

execute (Shard-1,shard-2,shard-3)

|  |
| --- |
| storage:  dbPath: /mongodb-config/data/sharddb  journal:  enabled: true  systemLog:  destination: file  logAppend: true  path: /mongodb-config/data/logs/shard.log  net:  port: 27018  bindIp: **(shard\_server\_private\_ip)**  sharding:  clusterRole: shardsvr  replication:  replSetName: **"rs0"** |

execute (Shard-4,shard-5,shard-6)

|  |
| --- |
| storage:  dbPath: /mongodb-config/data/sharddb  journal:  enabled: true  systemLog:  destination: file  logAppend: true  path: /mongodb-config/data/logs/shard.log  net:  port: 27018  bindIp: **(shard\_server\_private\_ip)**  sharding:  clusterRole: shardsvr  replication:  replSetName: **"rs1"** |

Save and close the file then start the Shard servers with the following command (execute in two replica sets (6) servers Shard-“rs0” and Shard-“rs1”):

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

At this point, the Shard servers is started and listening on port 27018. You can check it with the following command (execute in two replica sets servers Shard-“rs0” and Shard-“rs1”):

|  |
| --- |
| ss -antpl | grep 27018 |

Sample output:

Text

Description automatically generated

## **Starting the shard Replica Sets and Adding Members**

Next, connect to the six shard servers using the following command(execute in two replica sets (6) servers Shard-“rs0” and Shard-“rs1”)::

|  |
| --- |
| mongo **(shard\_server\_private\_ip)**:27018 |

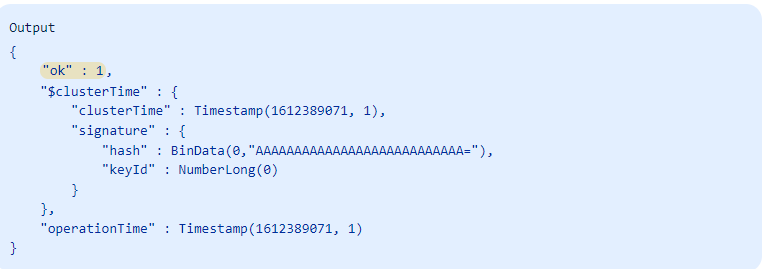
Bellow commands run in the **shard1-**server(initiate shard replica set (**rs0**)):

|  |
| --- |
| rs.initiate({ \_id : "rs0", members:[{ \_id : 0, host : "**shrd1\_private\_ip**:27018" },{ \_id : 1, host : "**shrd2\_server\_private\_ip**:27018" },{ \_id : 2, host : "**shrd3\_server\_private\_ip**:27018" }]}) |

initiate the config server using the bellow command to see output:

|  |
| --- |
| rs.initiate() |

Sample output:



Bellow commands run in the **shard4-**server(initiate shard replica set (**rs1**)):

|  |
| --- |
| rs.initiate({ \_id : "rs0", members:[{ \_id : 0, host : "**shrd4\_private\_ip**:27018" },{ \_id : 1, host : "**shrd5\_server\_private\_ip**:27018" },{ \_id : 2, host : "**shrd6\_server\_private\_ip**:27018" }]}) |

initiate the config server using the bellow command to see output:

|  |
| --- |
| rs.initiate() |

Next, check the status of the server with the following command(execute in two replica sets (6) servers Shard-“rs0” and Shard-“rs1”):

|  |
| --- |
| rs.status() |

**Step-4**

## **Configure Query Router**

Next, log in to the Query Router server and create a directory structure using the following command:

|  |
| --- |
| mkdir -p /mongodb-config/data/logs  touch /mongodb-config/data/logs/mongorouter.log |

Next, create a new configuration file for Query Router with the following command:

|  |
| --- |
| nano /etc/mongoRouter.conf |

Define your log file, IP address, port, and configDB as shown below:

|  |
| --- |
| systemLog:  destination: file  logAppend: true  path: /mongodb-config/data/logs/queryrouter.log  net:  port: 27017  bindIp: (Router\_server\_private\_ip)  sharding:  configDB:  configReplSet/**cfg1\_private\_IP**:27019,**cfg2\_server\_private\_ip**:27019,**cfg3\_server\_private\_ip**:27019 |

Sample output:

Text

Description automatically generated

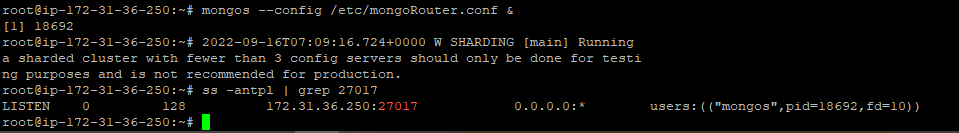
Save and close the file then start the Query Router with the following command:

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

At this point, your MongoDB Query Router is started and listening on port 27017. You can check it with the following command:

|  |
| --- |
| ss -antpl | grep 27017 |

Sample output:



Now, connect to the Query Router using the following command:

|  |
| --- |
| mongo **(router\_server\_private\_ip)**:27017 |

Once you are connected, you should get the following output:

Text

Description automatically generated

## **Step-5**

## **Add the Shard to the Cluster**

Next, you will need to add your Shard server to the cluster. First, log in to the Query Router server. Then, connect to the mongo shell with the following command**(query router):**

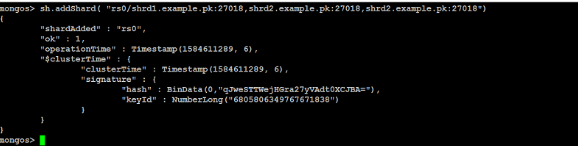
(In case you already login router mongo shell not use this command)

|  |
| --- |
| mongo **(router\_server\_private\_ip)**:27017 |

Once you are connected query router server, add your **Shard-“rs0”** servers with the following command:

|  |
| --- |
| sh.addShard( "rs0/**shrd1\_server\_private\_ip**:27018,**shrd2\_server\_private\_ip**:27018,**shrd3\_server\_private\_ip**:27018") |

You will see the similar output indicating that shard replica set rs0 added successfully.



next add your **Shard-“rs1”** servers with the following command:

|  |
| --- |
| sh.addShard( "rs1/**shrd4\_server\_private\_ip**:27018,**shrd5\_server\_private\_ip**:27018,**shrd6\_server\_private\_ip**:27018") |

**Step-6**

## **Enable Sharding for a Database**

Next, you will need to create a new database and enable sharding for the new database. On the Query Router server, connect to the mongo shell and run the following command to create a new database named **peoples**:

|  |
| --- |
| use peoples |

Next, enable the sharding on the peoples database with the following command:

|  |
| --- |
| sh.enableSharding("peoples") |

Next, check the sharding status using the following command:

|  |
| --- |
| sh.status() |

Next, you will need to add a new collection to the database with sharding support. Let’s add a new collection named **collection** to the **peoples** database:

|  |
| --- |
| sh.shardCollection("peoples.collection", {"name":1}) |

Next, insert the documents to the collections with the following command:

|  |
| --- |
| db.collection.save({  "name": "Application List",  "apps": ["Apache", "MariaDB", "Redis", "PHP"],  }) |

**Step-7**

## **Verify Sharding**

Next, you will need to verify if the sharding is working as intended. First, log in to the Shard server and connect to the mongo shell with the following command:

|  |
| --- |
| mongo 216.98.8.110:27018 |

Once you are connected, run the following command to check the database available on the replica set:

|  |
| --- |
| show dbs |

Next, switch database to peoples with the following command:

|  |
| --- |
| use peoples |

Next, check your collections and documents in the replica set using the following command:

|  |
| --- |
| db.collection.find() |

**Step-7**

**Install mongo dB Compass and connect Database**

Lunch windows instance and login.

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

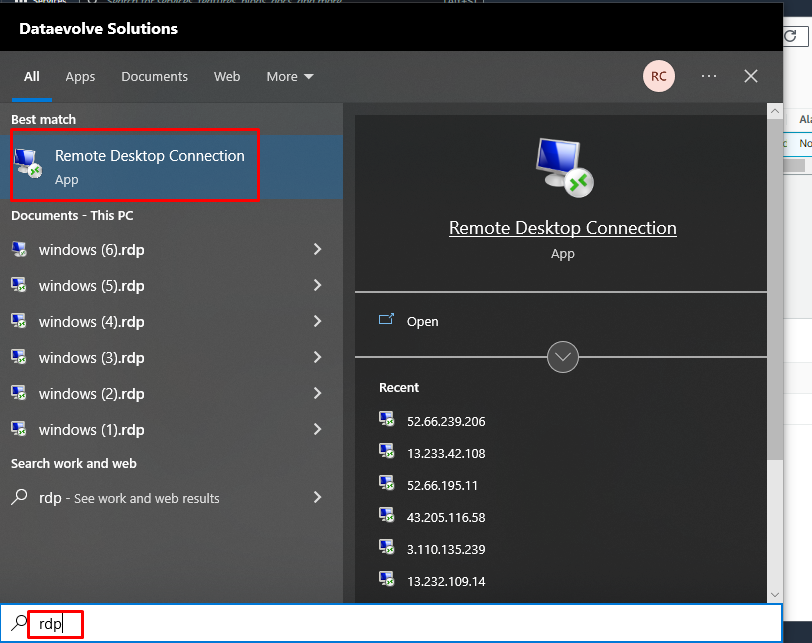
Description automatically generated

Graphical user interface, text, application

Description automatically generated

Copy the above password and username

Open RDP.



Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Next install mongodb compass in windows instance.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Give the router server private ip in the place of local host.

Graphical user interface, application

Description automatically generated

After login

A screenshot of a computer

Description automatically generated

Reference websites:

<https://cloudinfrastructureservices.co.uk/mongodb-sharding/>

<https://www.digitalocean.com/community/tutorials/how-to-configure-a-mongodb-replica-set-on-ubuntu-20-04>

<https://www.techsupportpk.com/2020/03/fault-tolerant-mongodb-sharded-cluster-using-shared-storage-ubuntu-debian.html>