Steps to create a sharded cluster (without Replication) on multiple systems

Note: Configure the hosts and hostname file of all the systems (4) to represent the host names (system names) and their IP addresses to enable communication between the systems (similar to replication).

Use following commands

- ⇒ **sudo chmod -R 777 /etc/hostname**
- ⇒ **nano /etc/hostname**
- ⇒ **nano /etc/hosts**

Infrastructure requirements

Minimum number of servers (systems) 4, which are described below

1. Shard servers – 2 systems
2. Config server -1 system (It is a server which contains the meta-data of the shards and acts as a router for all the operations performed on shards. A replica set must be created within this machine)
3. Client Server (mongos) - 1 system (To perform CRUD operations on shards)

Maintain the following details of systems

1. Host name
2. Data directory
3. Port number
4. IP Address

Example

System	Host name (System name)	Data directory	Port number	IP Address
1	shard1 (shard server 1)	/srv/mongodb/shard1	29010	
2	shard2 (shard server 2)	/srv/mongodb/shard2	29020	
3	cserver (config server)	/srv/mongodb/cserver	29030	
4	client (client server)	/data/db (default)	29017	

Note:

The table above is just provided as reference, the names of the systems, data directories and port number can be given as per user convenience except for client server (it has to be default directory "/data/db" and default port number "27017").

The steps provided below uses the above information for creating a sharded cluster.

Step 1 (In each system)

Set permissions to the directory in which data directories for the shard is being created

sudo chmod -R 777 /srv

NOTE: srv is the folder in which data directories will be created.

Step 2

To start Shard-1 (system: First)

2.1. Create appropriate directory **mkdir -p /srv/mongodb/shard1**

2.2. Deploy Shard1

mongod –shardsvr --port 29010 --dbpath /srv/mongodb/shard1

2.3. Connect to a mongo shell (new terminal) **mongo --port 29010**

Note: Create a new collection and insert the documents into the database using script

Step 3

To start Shard-2 (system: Second)

3.1. Create appropriate directory **mkdir -p /srv/mongodb/shard2**

3.2. Deploy Shard2

mongod –shardsvr --port 29020 --dbpath /srv/mongodb/shard2

3.3. Connect to a mongo shell (new terminal) **mongo --port 29020**

Note: For testing purpose, do not insert any data into shard 2.

Step 4

To start config server (system: Third)

4.1. Deploy Config servers: Replication set name: **configReplSet**

Config Server 1: first terminal (system name: **configp**)

mongod --configsvr --replSet configReplSet --port 28023 --dbpath /srv/mongodb/config1

Config Server 2: second terminal (system name: **configs1**)

mongod --configsvr --replSet configReplSet --port 28024 --dbpath /srv/mongodb/config2

Config Server 3: third terminal (system name: **configs2**)

mongod --configsvr --replSet configReplSet --port 28025 --dbpath /srv/mongodb/config3

Note: --configsvr represents that the replication set being deployed acts as config servers (routers).

4.2. Connect a mongo shell to one of the config servers and run rs.initiate() to initiate the replica set. **mongo --port 28023**

Inside the mongo instance initiate the config server replica set using the code given below:

```
rs.initiate ({_id: "configReplSet", configsvr: true,
members: [{_id: 0, host: "configp:28023"},
          {_id: 1, host: "configs1:28024"},
          {_id: 2, host: "configs2:28025"}]})
```

Step 5

To start Mongos (system: Fourth)

5.1. Create appropriate directory

For client server create directory only if it doesn't exist (/data/db)

Note:

The client server connects on default mongoDB port 27017 which stores the data in the path /data/db in the root directory.

Hence the path /data/db has to be created by executing the code below

```
student@student:~$ sudo su
root@student:/home/student# mkdir -p /data/db
root@student:/home/student# sudo chmod -R 777 /data
root@student:/home/student# sudo chmod -R 777 /data/db
root@student:/home/student# exit
```

5.2. Deploy the Client server **mongos --configdb cserver:29030**

Note:

In the above example cserver represents the name of the system on which the config server is hosted (system: 3)

5.3. Connect a mongo shell

mongo student:27017/admin

admin represents that all operations can be performed on the data sets.

5.4. Open mongod.conf file (use nano editor) in other terminal in mongos machine and write following

security: authorization: enabled
keyFile: /var/run/mongodb/secKey.key

Step 6

On Mongos Shell (system:Fourth)

6.1. Add shard1 **sh.addShard("shard1:29010")**
6.2. Add shard2 **sh.addShard("shard2:29020")**
6.3. Enable sharding for a specific database and shard a collection (inside the mongos instance)

Note: Before a collection can be sharded, first enable sharding for the collection's database. Enabling sharding for a database does not redistribute data but makes it possible to shard the collections in that database.

The following operation enables sharding on the pesu database

sh.enableSharding("pesu")
Note: pesu is the name of the database present in shard 1.

6.4. Determine shard key

Note:

1. The shard key determines how MongoDB distributes the documents between shards.
2. Once a collection is sharded with the specified shard key the shard key key cannot be changed.
3. The shard key determines the distribution of the collection's documents among the cluster's shards.
   The shard key is either an indexed field or an indexed compound field that exists in every document in the collection.

For example "number" field of the collection mca is taken as shard key and before sharding a nonempty collection, an index is created on the shard key

use pesu
db.mca.createIndex({ number : 1 })
In the above example "number" represents one of the attributes in the collection which is taken as shard key (any attribute can be taken as shard key).

6.5. Execute Sharding on the collection

sh.shardCollection("pesu.mca", {number:1})

6.6. To confirm balancing activity

use pesu

db.stats() db.printShardingStatus()

Verification of the shards can be viewed by using the command **db.mca.find().count()**, number of documents in both the shard. Documents can be seen and accessed.

Script to insert documents Create a new collection and insert the documents into the database as follows using script

use pesu //database is pesu

people = ["isha", "lekha", "krishna", "manish", "pushpa", "achutha", "varshini"];

var script = db.mca.initializeUnorderedBulkOp(); //mca is the collection

```
for (var i=0; i<1000000; i++)
{
        user_id = i;
        name = people [Math.floor(Math.random()*people.length)];
        number = Math.floor(Math.random()*100);
        script.insert( { "user_id": user_id, "name": name, "number": number });
}
script.execute()
```

In the above example "pesu" is the database and "mca" is the collection.

Note: script.execute() may take several minutes depending upon the system/s.