**MongoDB Replication**

**Replication -** The replication is the process to synchronizing data across multiple servers. Replication provides redundancy and increase high availability with multiple copies of data on different database servers. Replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions. With additional copies of the data, you can dedicate one to disaster recovery, reporting, or backup.

There are some reasons to maintains the Data Replication

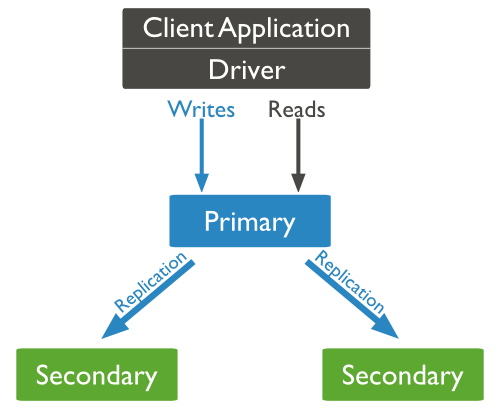
1. To keep data safe.
2. High availability
3. Disaster recovery
4. No downtime for maintenance (like backups, rebuild index, compaction)
5. Replica set is transparent to the application.

**How replication works**

MongoDB achieves replication by using replica set. The **Replica Set** is a group of **mongod** instances that keeps the same data set. In replica set, one node is primary node that receives all write operations. All other instances that is known as secondary nodes. The primary records all changes to its data sets in its operation log, i.e. oplog. The secondaries replicate the primary’s oplog and apply the operations to their data sets such that the secondaries data sets reflect the primary’s data set. If the primary is unavailable, an eligible secondary will hold an election to elect itself the new primary.

1. The replica set is group of two or more nodes.
2. In replica set, one node is primary, others nodes are secondaries
3. All data replicate from primary to secondary.
4. At the time of automatic failover or maintenance, election establishes for primary and a new primary node is elected.
5. After the recovery of failed node, it again join the replica set and works as a secondary node.

The following pictures shows the replication



**Setting Up Replica Set Syntax and commands**

**Steps to setup replica set.** To convert all ready running mongodb to replica set we will follows the given steps

Step 1- Shutdown already mongoDB server.

Step 2- Start mongdb server by specifying –replSet option

mongod --port "PORT" --dbpath "YOUR\_DB\_DATA\_PATH" --replSet "REPLICA\_SET\_INSTANCE\_NAME"

For Example :

mongod --port 27017 --dbpath "D:\set up\mongodb\data" --replSet rs0

1. It will start a mongod instance with name rs0, on port 27017
2. Now start the command prompt and connect to this mongod instance.
3. In Mongo client, issue the command rs.initiate() to initiate a new replica set.
4. To check the replica set configuration, issue the command rs.conf(). To check the status of replica set issue the command rs.status().

**Add Members to Replica Set**

To add members to replica set, start mongod instances on multiple machines. Now start a mongo client and issue a command rs.add().

Syntax :

rs.add(HOSTNAME:PORT);

For Example :

Suppose your instance name is pcSap1 and running mongod port is 27017

rs.add("pcSap1:27017");

You can add mongod instance to replica set only when you are connected to primary node.

To check whether you are connected to primary or not, issue the command db.isMaster() in mongo client.

**How to Setup Replica on windows machine with Example :**

1. Start standalone server as shown below.

"C:\Program Files\MongoDB\Server\3.0\bin\mongod.exe" --config "C:\Program Files\MongoDB\mongod.cfg" --journal --serviceName MongoDB --serviceDisplayName MongoDB –install

Where mongod.cfg contains the following lines

dbpath = E:\MongoData

logpath = E:\MongoData\mongo.log

logappend = true

#port number

port=27017

#replica set name

replSet=rs1

1. Connect to the server with port number 27017

mongo --port 27017

1. Then, create variable rsconf

rsconf={\_id:"rs1",members:[{\_id:0,host:"localhost:27017"}]}

rs.initiate(rsconf);

1. Start secondary server on the port 27018
2. "C:\Program Files\MongoDB\Server\3.0\bin\mongod.exe" --config "C:\Program Files\MongoDB1\mongod1.cfg" --journal --serviceName MongoDB1 --serviceDisplayName MongoDB1 –install

The mongod1.cfg file contains the lines

dbpath = E:\MongoData1

logpath = E:\MongoData1\mongo1.log

logappend = true

#port number

port=27018

#replica set name

replSet=rs1

1. Logon to secondary server

mongo --port 27018

1. Run the following commands on Primary server

rs.add("localhost:27017");

rs.add("localhost:27018");

1. Now go to secondary servers and run below command on both the secondary servers

rs.slaveOk();

**How we can verify replica set**

1. When we connect with primary it shows the following prompt

mongo –port 27017  
rs1:PRIMARY>

1. Then run use mydb;

rs1:PRIMARY>use mydb;

When you insert any records in book collection. For example

It will reflect in secondary

1. Now connect with secondary

mongo –port 27018

It shows

rs1:SECONDARY>

then run the use mydb

and run the following command to view the inserted records by primary.

db.book.find({});

**Sharding**

Sharding is a MongoDB process to store data-set across different machines. It allows you to perform a horizontal scale of data and to partition all data across independent instances. Sharding allows you to add more machines based on data growth to your stack.

There are three components of sharding.

1. **Shard** - Used to store all data. And in a production environment, each shard is replica sets. Provides high-availability and data consistency.
2. **Config Server** - Used to store cluster metadata, and contains a mapping of cluster data set and shards. This data is used by mongos/query server to deliver operations. It's recommended to use more than 3 instances in production.
3. **Mongos/Query Router** - This is just mongo instances running as application interfaces. The application will make requests to the 'mongos' instance, and then 'mongos' will deliver the requests using shard key to the shards replica sets.

In this documents I only provide methods for setting up sharding in production/development environments but not practical in local machine.

**Prerequisites**

2 centOS 7 server as Config Replica Sets

10.0.15.31 configsvr1

10.0.15.32 configsvr2

4 CentOS 7 server as Shard Replica Sets

10.0.15.21 shardsvr1

10.0.15.22 shardsvr2

10.0.15.23 shardsvr3

10.0.15.24 shardsvr4

1 CentOS 7 server as mongos/Query Router

10.0.15.11 mongos

Root privileges

Each server connected to another server

**There are following steps which need to be follows for setting up sharding.**

**Step 1 :**

**Disable SELinux and configuration hosts**

Connects to all nodes through putty or openSSH tool.

root@serverIP

vim /etc/sysconfig/selinux

change SELINUX value to 'disabled'.

change SELINUX value to 'disabled'.

Save and Exit

Next, edit the hosts file on each server.

vim /etc/hosts

Paste the following hosts configuration:

10.0.15.31 configsvr1

10.0.15.32 configsvr2

10.0.15.11 mongos

10.0.15.21 shardsvr1

10.0.15.22 shardsvr2

10.0.15.23 shardsvr3

10.0.15.24 shardsvr4

Save and Exit

Now restart the all nodes with reboot command

**Step 2** :

**Install mongoDB server to all nodes**

Add new MongoDB repository by executing the following command:

cat <<'EOF' >> /etc/yum.repos.d/mongodb.repo

[mongodb-org-3.4]

name=MongoDB Repository

baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.4/x86\_64/

gpgcheck=1

enabled=1

gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc

EOF

Now install MongoDB 3.4 from mongodb repository using the following yum command.

sudo yum -y install mongodb-org

After mongodb is installed, you can use the 'mongo' or 'mongod' command.

mongod –version

**Step 3 :**

**Create Config Server Replica Set**

If there is a mongod service running on the server, stop it using the systemctl command.

systemctl stop mongod

open the mongodb configuration file and add the following configuration

vim /etc/mongod.conf

paste or edit the following lines in mongod.conf

#Change the DB storage path to your own directory. We will use '/data/db1' for the first #server, and '/data/db2' directory for the second config server.

storage:

dbPath: /data/db1

#Change the value of the line 'bindIP' to your internal network addres - 'configsvr1' with #IP address 10.0.15.31, and the second server with 10.0.15.32.

bindIP: 10.0.15.31

#On the replication section, set a replication name.

replication:

replSetName: "replconfig01"

#And under sharding section, define a role of the instances. We will use these two #instances as 'configsvr'.

sharding:

clusterRole: configsvr

After adding save and exit.

Next, we must create a new directory for MongoDB data, and then change the owner of that directory to 'mongod' user.

mkdir -p /data/db1

chown -R mongod:mongod /data/db1

After this, start the mongod service with the command below.

mongod --config /etc/mongod.conf

after doing this repeats the same for 10.0.15.32

Configsvr1 and Configsvr2 are ready for the replica set. Connect to the 'configsvr1' server and access the mongo shell.

ssh root@configsvr1

mongo --host configsvr1 --port 27017

Initiate the replica set name with all configsvr member using the query below.

rs.initiate(

{

\_id: "replconfig01",

configsvr: true,

members: [

{ \_id : 0, host : "configsvr1:27017" },

{ \_id : 1, host : "configsvr2:27017" }

]

}

);

If you get a results '{ "ok" : 1 }', it means the configsvr is already configured with replica set. and you will be able to see which node is master and which node is secondary.

rs.isMaster();

rs.status();

The configuration of Config Server Replica Set is done.

**Step 4 :**

**Create the shard replica set**

In this step, we will configure 4 'centos 7' servers as 'Shard' server with 2 'Replica Set'.

* 2 server - 'shardsvr1' and 'shardsvr2' with replica set name: 'shardreplica01'
* 2 server - 'shardsvr3' and 'shardsvr4' with replica set name: 'shardreplica02'

Connect to each server, stop the mongod service (If there is service running), and edit the MongoDB configuration file.

systemctl stop mongod

vim /etc/mongod.conf

Add or edit the following lines in mongodb.conf file

#Change the default storage to your specific directory.

storage:

dbPath: /data/db1

#On the 'bindIP' line, change the value to use your internal network address.

bindIP: 10.0.15.21

#On the replication section, you can use 'shardreplica01' for the first and second #instances. And use 'shardreplica02' for the third and fourth shard servers.

replication:

replSetName: "shardreplica01"

#Next, define the role of the server. We will use all this as shardsvr instances.

sharding:

clusterRole: shardsvr

after adding this save and exit

Now, create a new directory for MongoDB data.

mkdir -p /data/db1

chown -R mongod:mongod /data/db1

Start the mongod service.

mongod --config /etc/mongod.conf

Next, create a new replica set for these 2 shard instances. Connect to the 'shardsvr1' and access the mongo shell.

ssh root@shardsvr1

mongo --host shardsvr1 --port 27017

Initiate the replica set with the name 'shardreplica01', and the members are 'shardsvr1' and 'shardsvr2'.

rs.initiate(

{

\_id : "shardreplica01",

members: [

{ \_id : 0, host : "shardsvr1:27017" },

{ \_id : 1, host : "shardsvr2:27017" }

]

}

);

If there is no error, you will see results as below.

Results from shardsvr3 and shardsvr4 with replica set name 'shardreplica02'.

Redo this step for shardsvr3 and shardsvr4 servers with different replica set name 'shardreplica02'.

Now we've created 2 replica sets - 'shardreplica01' and 'shardreplica02' - as the shard.

**Step 5 -**

**Configure mongos/Query Router**

The 'Query Router' or mongos is just instances that run 'mongos'. You can run mongos with the configuration file, or run with just a command line.

Login to the mongos server and stop the MongoDB service.

ssh root@mongos

systemctl stop mongod

Run mongos with the command line as shown below.

mongos --configdb "replconfig01/configsvr1:27017,configsvr2:27017"

**Step 6 –**

**Add shards to mongos/Query Router**

Open another shell from the previous step, connect to the mongos server again, and access the mongo shell.

ssh root@mongos

mongo --host mongos --port 27017

Add shard server with the sh mongodb query.

For 'shardreplica01' instances:

sh.addShard( "shardreplica01/shardsvr1:27017");

sh.addShard( "shardreplica01/shardsvr2:27017");

For 'shardreplica02' instances:

sh.addShard( "shardreplica02/shardsvr3:27017");

sh.addShard( "shardreplica02/shardsvr4:27017");

Make sure there is no error and check the shard status.

sh.status();

Now We have 2 shard replica set and 1 mongos instance running on our stack.

**Step 7 –**

**Testing**

To test the setup, access the mongos server mongo shell.

ssh root@mongos

mongo --host mongos --port 27017

**Enable Sharding for a Database**

Create a new database and enable sharding for the new database.

use lemp;

sh.enableSharding("lemp");

sh.status();

Now see the status of the database, it's has been partitioned to the replica set 'shardreplica01'.

**Enable Sharding for Collections**

next, add new collections to the database with sharding support. We will add new collection named 'stack' with shard collection 'name', and then see database and collections status.

sh.shardCollection("lemp.stack", {"name":1});

sh.status();

New collections 'stack' with shard collection 'name' has been added.

**Add documents to the collections 'stack'.**

Now insert the documents to the collections. When we add documents to the collection on sharded cluster, we must include the 'shard key'.

In the example below, we are using shard key 'name', as we added when enabling sharding for collections.

db.stack.save({

"name": "LEMP Stack",

"apps": ["Linux", "Nginx", "MySQL", "PHP"],

});

If you want to test the database, you can connect to the replica set 'shardreplica01' PRIMARY server and open the mongo shell. I'm logging in to the 'shardsvr2' PRIMARY server.

ssh root@shardsvr2

mongo --host shardsvr2 --port 27017

Check database available on the replica set.

show dbs;

use lemp;

db.stack.find();

You will see that the database, collections, and documents are available in the replica set.

MongoDB Sharded Cluster on CentOS 7 has been successfully installed and deployed.

----------------------------------------------------------------------------------------------------

**Sharding setup in single linux machine :**

# MongoDB

# script to start a sharded environment on localhost

# clean everything up

echo "killing mongod and mongos"

killall mongod

killall mongos

echo "removing data files"

rm -rf /data/config

rm -rf /data/shard\*

# start a replica set and tell it that it will be shard0

echo "starting servers for shard 0"

mkdir -p /data/shard0/rs0 /data/shard0/rs1 /data/shard0/rs2

mongod --replSet s0 --logpath "s0-r0.log" --dbpath /data/shard0/rs0 --port 37017 --fork --shardsvr --smallfiles

mongod --replSet s0 --logpath "s0-r1.log" --dbpath /data/shard0/rs1 --port 37018 --fork --shardsvr --smallfiles

mongod --replSet s0 --logpath "s0-r2.log" --dbpath /data/shard0/rs2 --port 37019 --fork --shardsvr --smallfiles

sleep 5

# connect to one server and initiate the set

echo "Configuring s0 replica set"

mongo --port 37017 << 'EOF'

config = { \_id: "s0", members:[

{ \_id : 0, host : "localhost:37017" },

{ \_id : 1, host : "localhost:37018" },

{ \_id : 2, host : "localhost:37019" }]};

rs.initiate(config)

EOF

# start a replicate set and tell it that it will be a shard1

echo "starting servers for shard 1"

mkdir -p /data/shard1/rs0 /data/shard1/rs1 /data/shard1/rs2

mongod --replSet s1 --logpath "s1-r0.log" --dbpath /data/shard1/rs0 --port 47017 --fork --shardsvr --smallfiles

mongod --replSet s1 --logpath "s1-r1.log" --dbpath /data/shard1/rs1 --port 47018 --fork --shardsvr --smallfiles

mongod --replSet s1 --logpath "s1-r2.log" --dbpath /data/shard1/rs2 --port 47019 --fork --shardsvr --smallfiles

sleep 5

echo "Configuring s1 replica set"

mongo --port 47017 << 'EOF'

config = { \_id: "s1", members:[

{ \_id : 0, host : "localhost:47017" },

{ \_id : 1, host : "localhost:47018" },

{ \_id : 2, host : "localhost:47019" }]};

rs.initiate(config)

EOF

# start a replicate set and tell it that it will be a shard2

echo "starting servers for shard 2"

mkdir -p /data/shard2/rs0 /data/shard2/rs1 /data/shard2/rs2

mongod --replSet s2 --logpath "s2-r0.log" --dbpath /data/shard2/rs0 --port 57017 --fork --shardsvr --smallfiles

mongod --replSet s2 --logpath "s2-r1.log" --dbpath /data/shard2/rs1 --port 57018 --fork --shardsvr --smallfiles

mongod --replSet s2 --logpath "s2-r2.log" --dbpath /data/shard2/rs2 --port 57019 --fork --shardsvr --smallfiles

sleep 5

echo "Configuring s2 replica set"

mongo --port 57017 << 'EOF'

config = { \_id: "s2", members:[

{ \_id : 0, host : "localhost:57017" },

{ \_id : 1, host : "localhost:57018" },

{ \_id : 2, host : "localhost:57019" }]};

rs.initiate(config)

EOF

# now start 3 config servers

echo "Starting config servers"

mkdir -p /data/config/config-a /data/config/config-b /data/config/config-c

mongod --logpath "cfg-a.log" --dbpath /data/config/config-a --port 57040 --fork --configsvr --smallfiles

mongod --logpath "cfg-b.log" --dbpath /data/config/config-b --port 57041 --fork --configsvr --smallfiles

mongod --logpath "cfg-c.log" --dbpath /data/config/config-c --port 57042 --fork --configsvr --smallfiles

# now start the mongos on a standard port

mongos --logpath "mongos-1.log" --configdb localhost:57040,localhost:57041,localhost:57042 --fork

echo "Waiting 60 seconds for the replica sets to fully come online"

sleep 60

echo "Connnecting to mongos and enabling sharding"

# add shards and enable sharding on the test db

mongo <<'EOF'

db.adminCommand( { addshard : "s0/"+"localhost:37017" } );

db.adminCommand( { addshard : "s1/"+"localhost:47017" } );

db.adminCommand( { addshard : "s2/"+"localhost:57017" } );

db.adminCommand({enableSharding: "school"})

db.adminCommand({shardCollection: "school.students", key: {student\_id:1}});

EOF

**Sharding setup on Single Window machine :**

REM --------------------------------------------------------------------------------------

REM Batch script to deploy MongoDB shard cluster.

REM For more detail on MongoDB shard cluster, see : http://docs.mongodb.org/manual/tutorial/deploy-shard-cluster/

REM --------------------------------------------------------------------------------------

@echo off

echo Start

set mongo\_home\_path=C:\Program Files\MongoDB\Server\3.0\bin

set root\_path=E:\temp\mongodb

set data\_path=%root\_path%\data\_sharded

REM --------------------------------------------------------------------------------------

echo Start replicatSet #1 and #2

REM Specifies the replica set name through the --replSet

REM By default, MongoDB overwrites the log file when the process restarts. To instead append to the log file, set the --logappend option.

REM Sends all diagnostic logging information to a log file with --logpath

REM Chose the directory where the mongod instance stores its data with --dbpath

REM Specifies the TCP port on which the MongoDB instance listens for client connections with --port

REM Specifies a maximum size in megabytes for the replication operation log with --oplogSize

REM Sets MongoDB to use a smaller default file size with --smallfiles

REM Configures this mongod instance as a shard in a partitioned cluster with --shardsvr

echo Start 3 mongod for replicatSet #1

start /b "s1-a" "%mongo\_home\_path%\mongod" --replSet s1 --logappend --logpath "%data\_path%\s1\a\s1-a.log" --dbpath "%data\_path%\s1\a" --port 37017 --oplogSize 200 --smallfiles --shardsvr

start /b "s1-b" "%mongo\_home\_path%\mongod" --replSet s1 --logappend --logpath "%data\_path%\s1\b\s1-b.log" --dbpath "%data\_path%\s1\b" --port 37018 --oplogSize 200 --smallfiles --shardsvr

start /b "s1-c" "%mongo\_home\_path%\mongod" --replSet s1 --logappend --logpath "%data\_path%\s1\c\s1-c.log" --dbpath "%data\_path%\s1\c" --port 37019 --oplogSize 200 --smallfiles --shardsvr

echo Start 3 mongod for replicatSet #2

start /b "s2-a" "%mongo\_home\_path%\mongod" --replSet s2 --logappend --logpath "%data\_path%\s2\a\s2-a.log" --dbpath "%data\_path%\s2\a" --port 47017 --oplogSize 200 --smallfiles --shardsvr

start /b "s2-b" "%mongo\_home\_path%\mongod" --replSet s2 --logappend --logpath "%data\_path%\s2\b\s2-b.log" --dbpath "%data\_path%\s2\b" --port 47018 --oplogSize 200 --smallfiles --shardsvr

start /b "s2-c" "%mongo\_home\_path%\mongod" --replSet s2 --logappend --logpath "%data\_path%\s2\c\s2-c.log" --dbpath "%data\_path%\s2\c" --port 47019 --oplogSize 200 --smallfiles --shardsvr

REM --------------------------------------------------------------------------------------

REM --------------------------------------------------------------------------------------

echo Start 3 config servers

REM By default, MongoDB overwrites the log file when the process restarts. To instead append to the log file, set the --logappend option.

REM Sends all diagnostic logging information to a log file with --logpath

REM Chose the directory where the mongod instance stores its data with --dbpath

REM Specifies the TCP port on which the MongoDB instance listens for client connections with --port

REM Declares that this mongod instance serves as the config database of a sharded cluster with --configsvr

start /b "cfg-a" "%mongo\_home\_path%\mongod" --logappend --logpath "%data\_path%\cfg-a\cfg-a.log" --dbpath "%data\_path%\cfg-a" --port 57017 --configsvr

start /b "cfg-b" "%mongo\_home\_path%\mongod" --logappend --logpath "%data\_path%\cfg-b\cfg-b.log" --dbpath "%data\_path%\cfg-b" --port 57018 --configsvr

start /b "cfg-c" "%mongo\_home\_path%\mongod" --logappend --logpath "%data\_path%\cfg-c\cfg-c.log" --dbpath "%data\_path%\cfg-c" --port 57019 --configsvr

REM --------------------------------------------------------------------------------------

timeout /t 1

REM --------------------------------------------------------------------------------------

echo Configure shard #1

REM Specifies the TCP port on which the MongoDB instance listens for client connections with --port

REM Specifies a JavaScript file to run and then exit with <file.js> at the end of the commande line

start /b "configure S1" "%mongo\_home\_path%\mongo" --port 37017 configS1.js

REM --------------------------------------------------------------------------------------

timeout /t 1

REM --------------------------------------------------------------------------------------

echo Configure shard #2

REM Specifies the TCP port on which the MongoDB instance listens for client connections with --port

REM Specifies a JavaScript file to run and then exit with <file.js> at the end of the commande line

start /b "configure S2" "%mongo\_home\_path%\mongo" --port 47017 configS2.js

REM --------------------------------------------------------------------------------------

timeout /t 1

REM --------------------------------------------------------------------------------------

echo Start mongos

REM Specifies the TCP port on which the MongoDB instance listens for client connections with --port

REM By default, MongoDB overwrites the log file when the process restarts. To instead append to the log file, set the --logappend option.

REM Sends all diagnostic logging information to a log file with --logpath

REM Specifies 3 configuration servers with --configdb

start /b "mongos" "%mongo\_home\_path%\mongos" --port 61017 --logappend --logpath "%data\_path%\mongos.log" --configdb localhost:57017,localhost:57018,localhost:57019

REM --------------------------------------------------------------------------------------

REM must wait before defining sharding configuration ...

timeout /t 20

REM --------------------------------------------------------------------------------------

echo define sharding configuration

REM Specifies the TCP port on which the MongoDB instance listens for client connections with --port

REM Provides the user with a shell prompt after the file finishes executing with --shell

REM Specifies a JavaScript file to run and then exit with <file.js> at the end of the commande line

start /b "configure shard" "%mongo\_home\_path%\mongo" --shell --port 61017 "%root\_path%\configShard.js"

REM --------------------------------------------------------------------------------------

echo This is the End.

Where some js file which should be included are as :

**configS1.js**

configS1 = {

"\_id" : "s1",

"members" : [

{

"\_id" : 1,

"host" : "localhost:37017"

},

{

"\_id" : 2,

"host" : "localhost:37018"

},

{

"\_id" : 3,

"host" : "localhost:37019"

}

]

};

rs.initiate(configS1);

**configS2.js**

configS2 = {

"\_id" : "s2",

"members" : [

{

"\_id" : 1,

"host" : "localhost:47017"

},

{

"\_id" : 2,

"host" : "localhost:47018"

},

{

"\_id" : 3,

"host" : "localhost:47019"

}

]

};

rs.initiate(configS2);

**configShard.js**

db.adminCommand({"addshard":"s1/localhost:37017"});

db.adminCommand({"addshard":"s2/localhost:47017"});

db.adminCommand({"enablesharding":"test"});

db.adminCommand({"shardcollection":"test.users","key":{"\_id":1}});

sh.status();

OK