### **Spring 2021 CS157C: NoSQL Database Systems**

Take-Home Midterm Exam

1. AWS setup:

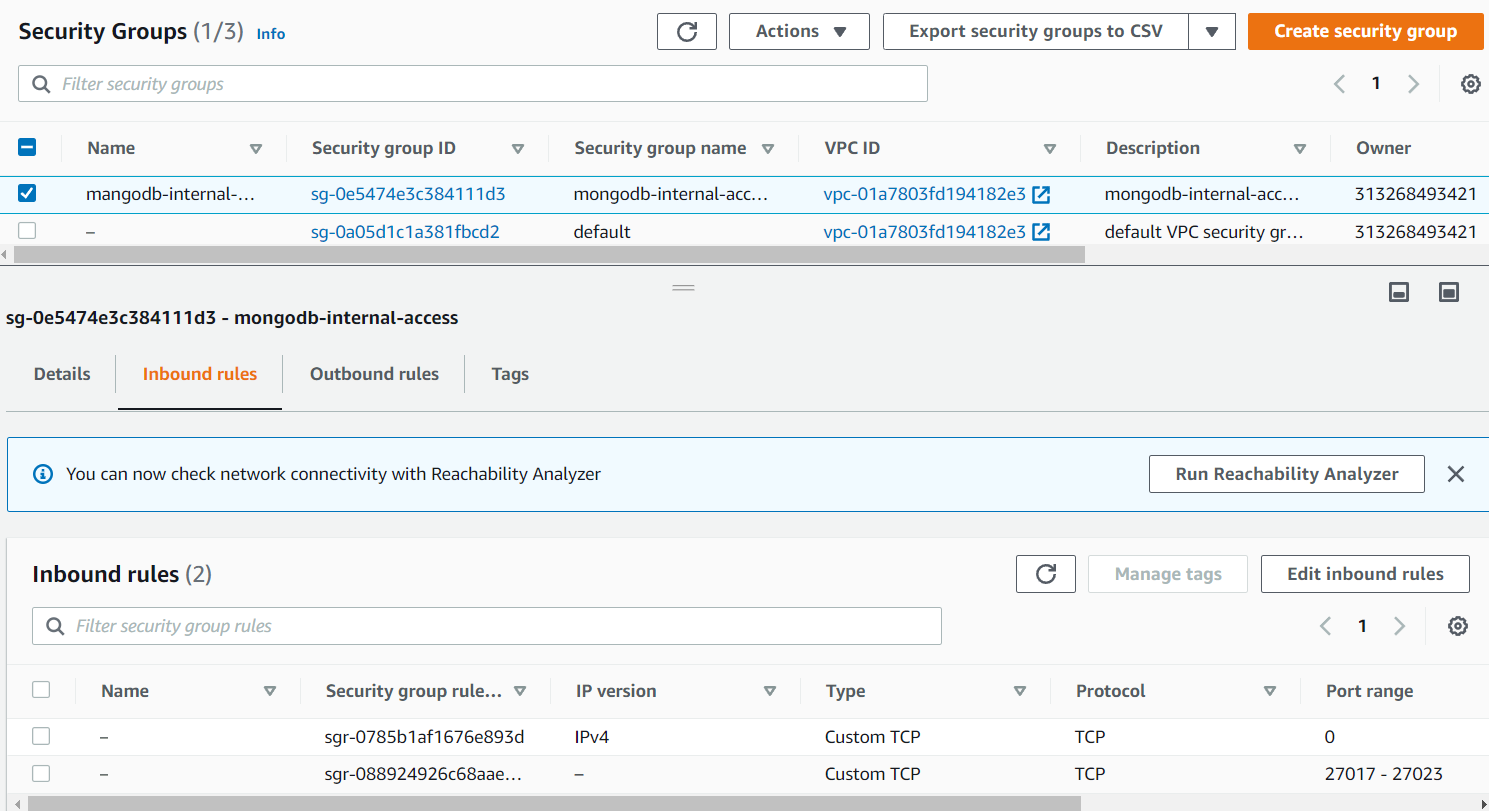
I have made use of 7 nodes for the purpose of the Midterm Exam. One instance for mongos. Three instances for config servers. Three instances for shards.

I planned on creating three replica sets in the shards.

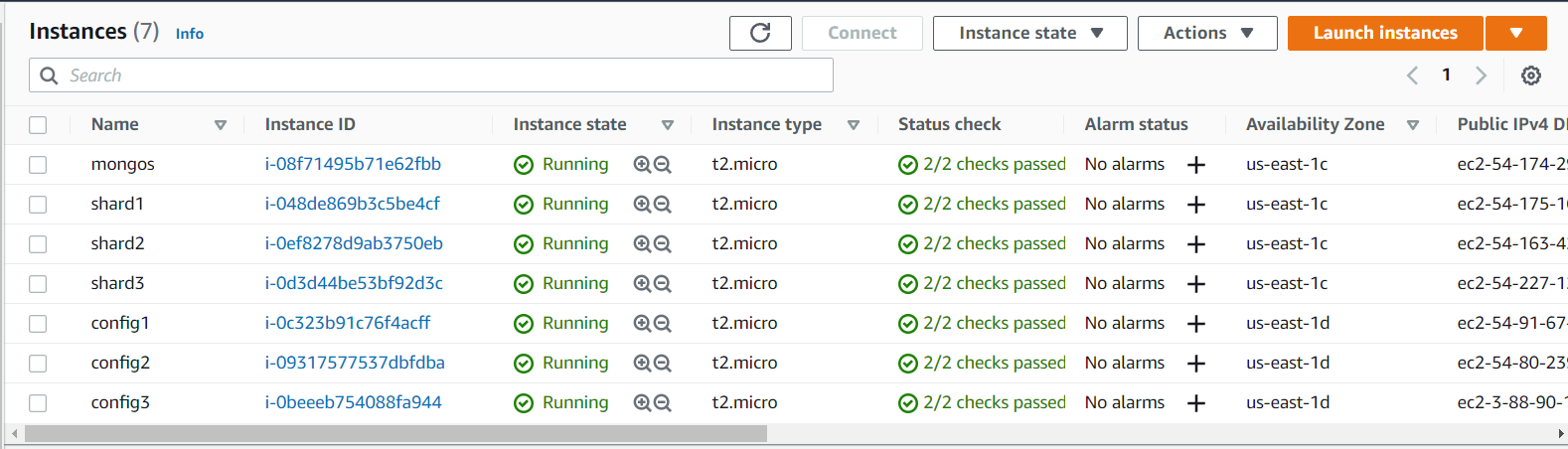
I have planned on serving the mongos on port 27018. Config servers on port 27019 and the replica sets of shards on ports 27020, 27021, and 27022.

I have created a security group with the name of mongodb-internal-access and gave ports as 27017-27023 in the inbound rule.

Screenshot of Security Group:



ScreenShot of EC2 instances:



1. Accessing the instances through ssh:

Commands used for connecting:

ssh -i kira.pem ubuntu@54.174.29.207 -> mongos

ssh -i kira.pem ubuntu@54.91.67.113 -> config1

ssh -i kira.pem ubuntu@54.80.239.239 -> config2

ssh -i kira.pem ubuntu@3.88.90.168 -> config3

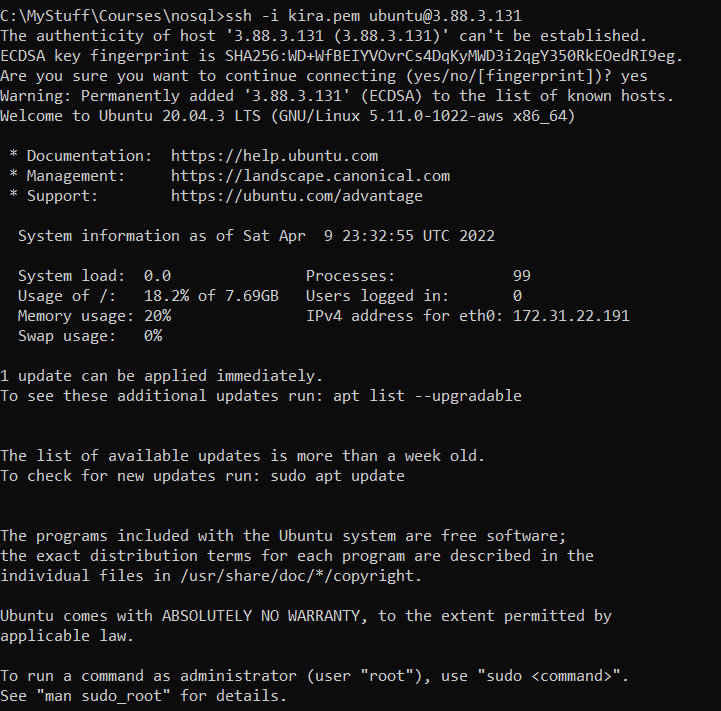
ssh -i kira.pem ubuntu@54.175.165.239 -> shard1

ssh -i kira.pem ubuntu@54.163.42.15 -> shard2

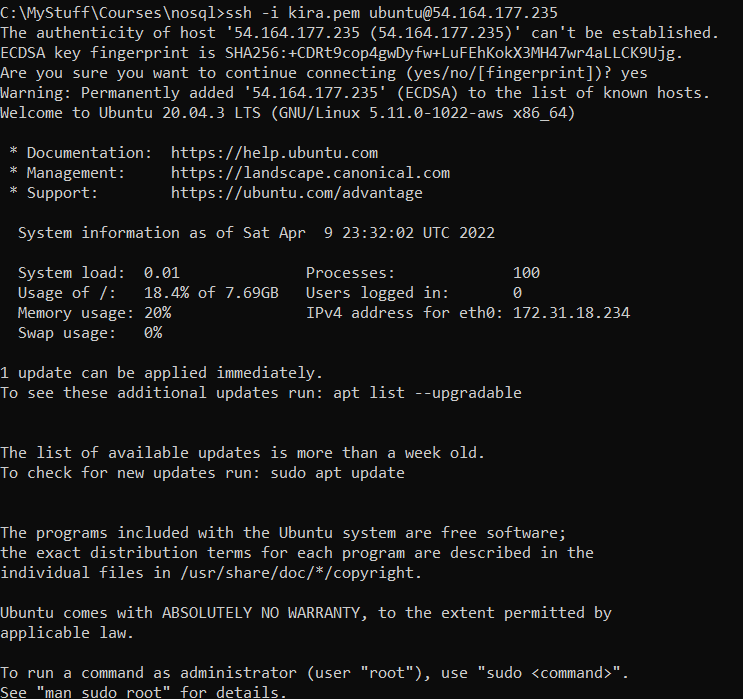
ssh -i kira.pem ubuntu@54.227.125.207 -> shard3

Proofs:

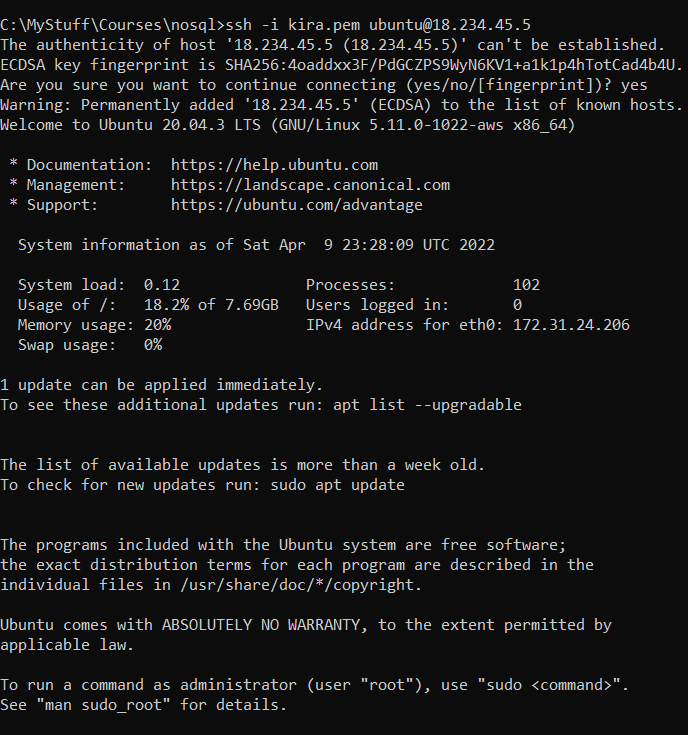
Config 1:



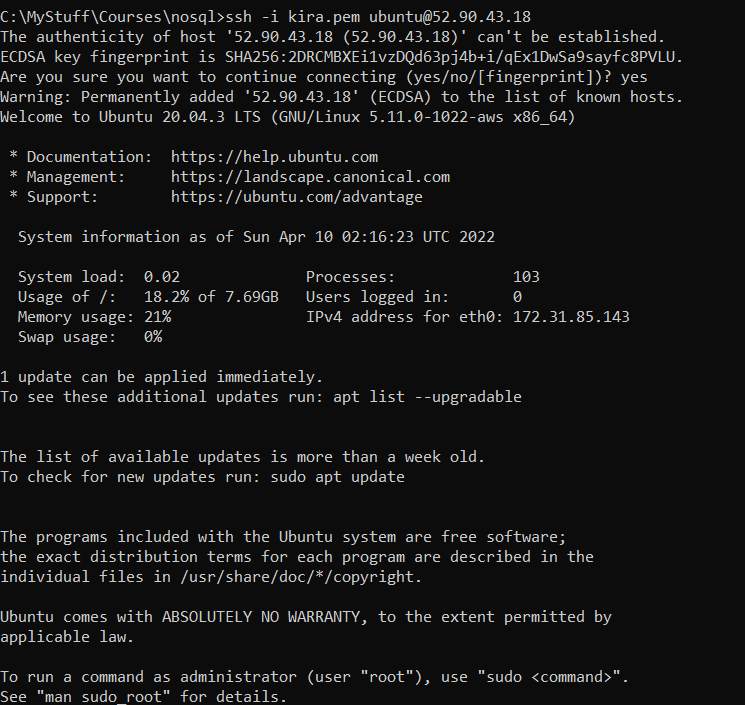
Config 2



Config 3



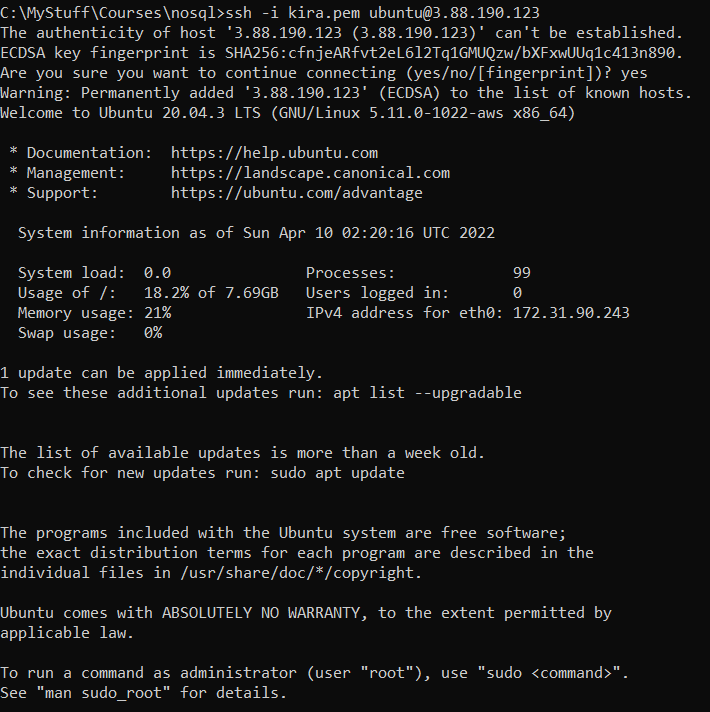
Mongos



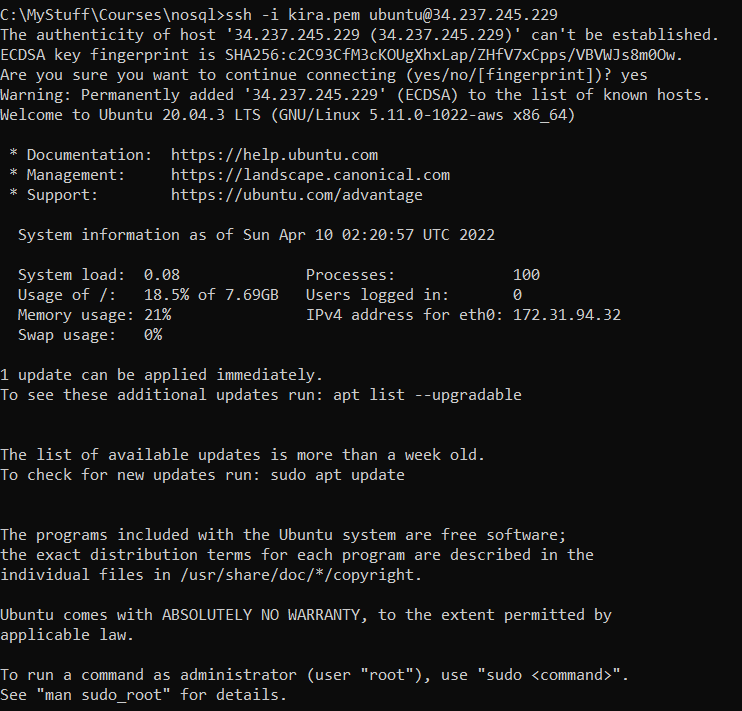
Shard1



Shard2



Shard 3



3. Install MongoDB in each node (i.e. instance)

I have created a bash script to install MongoDB. On each instance imported the bash script and executed it. Alternatively, I could have used ec2 images to install MongoDB.

Contents of bash script

wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add - \

&& apt-get install gnupg \

&& wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add - \

&& echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list \

&& apt-get update \

&& apt-get install -y mongodb-org \

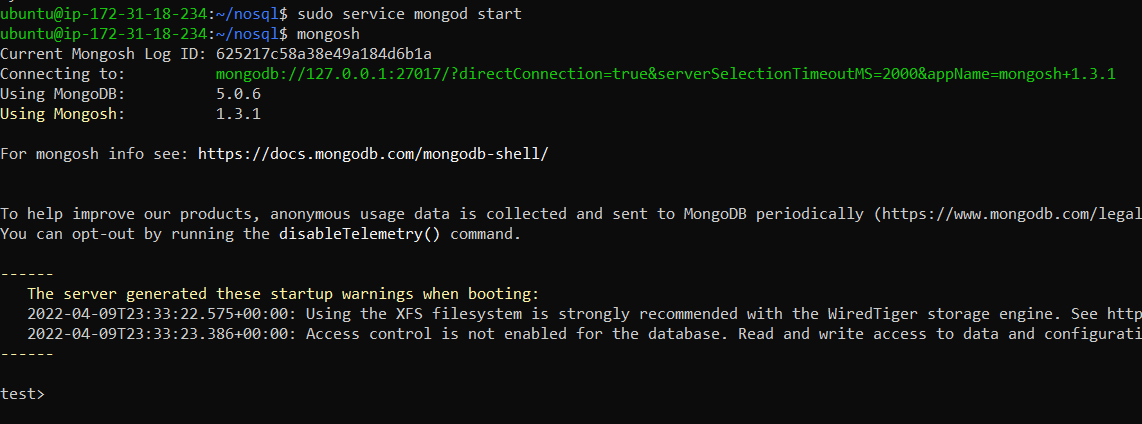
&& ps --no-headers -o comm 1

Proofs:

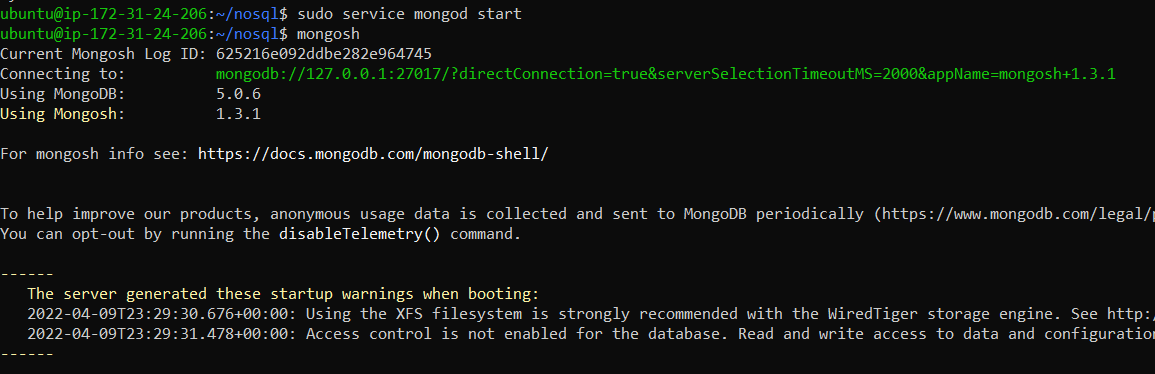
Config 1



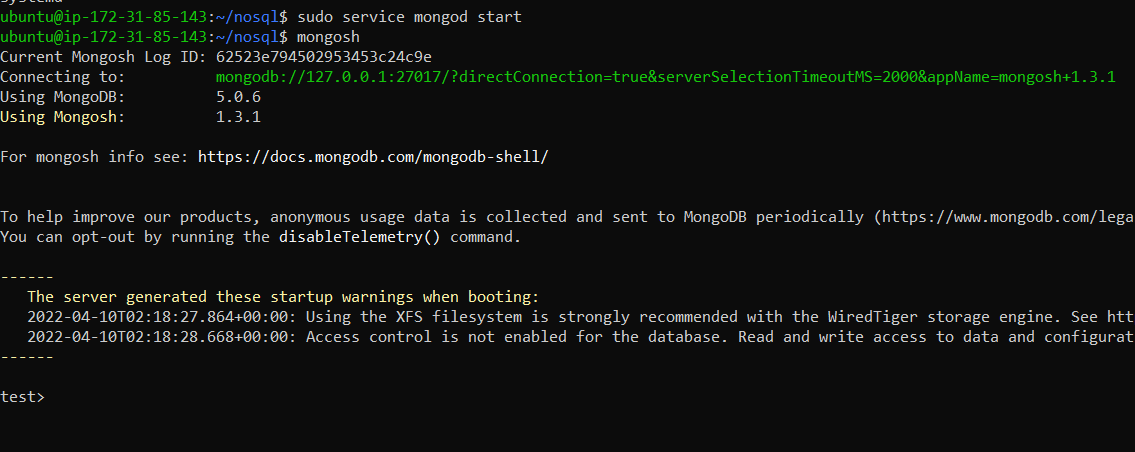
Config 2



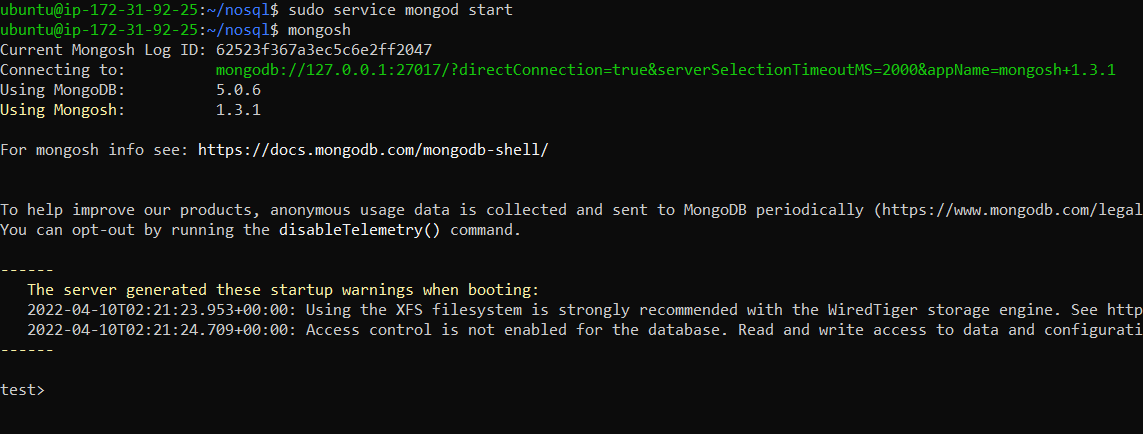
Config 3



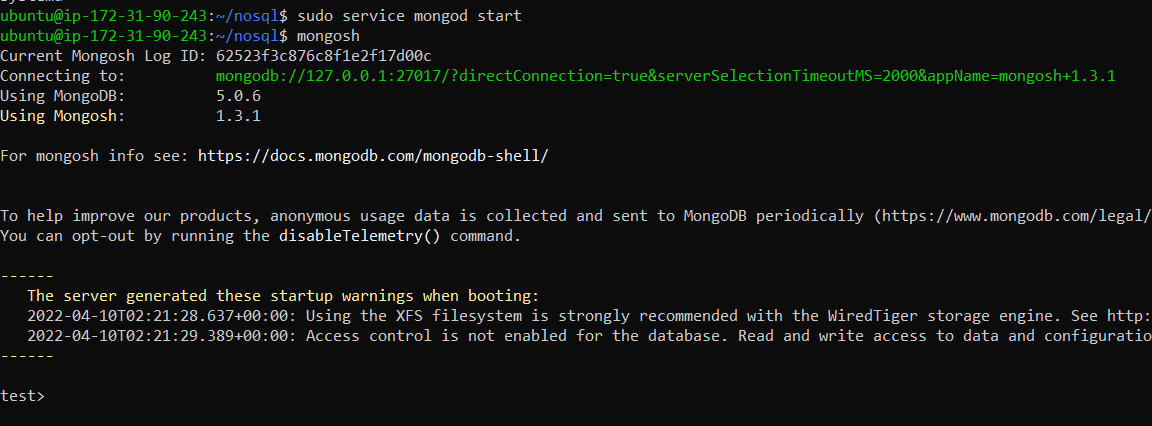
Mongos

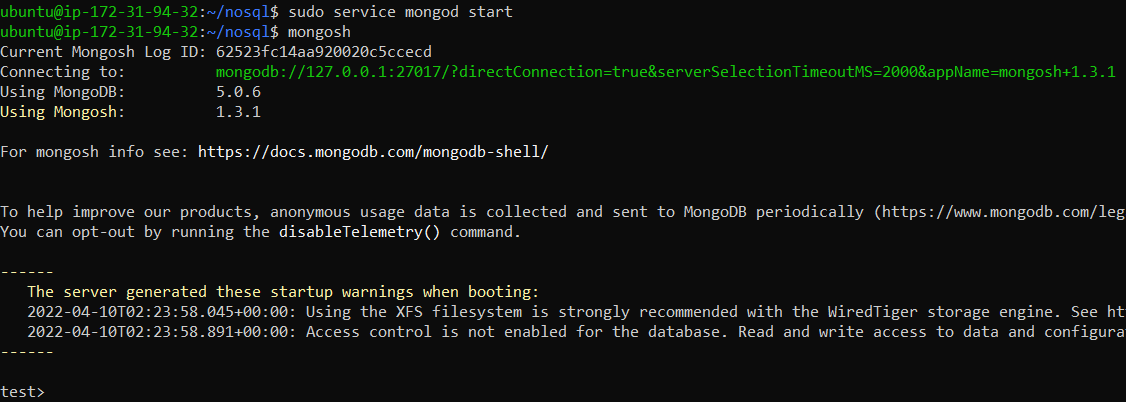


Shard1



Shard2:



Shard 3:

4. Creating directories to store the database in each node

I have created data/db on config instances to store the database and data/db0, data/db1 and data/db2 on shards instances.

Commands used:

On each of config instances:

sudo mkdir -p /data/db sudo chmod 777 /data/db

On each of the shard instances:

sudo mkdir -p /data/db0

sudo chmod 777 /data/db0

sudo mkdir -p /data/db1

sudo chmod 777 /data/db1

sudo mkdir -p /data/db2

sudo chmod 777 /data/db2

5. IP addresses:

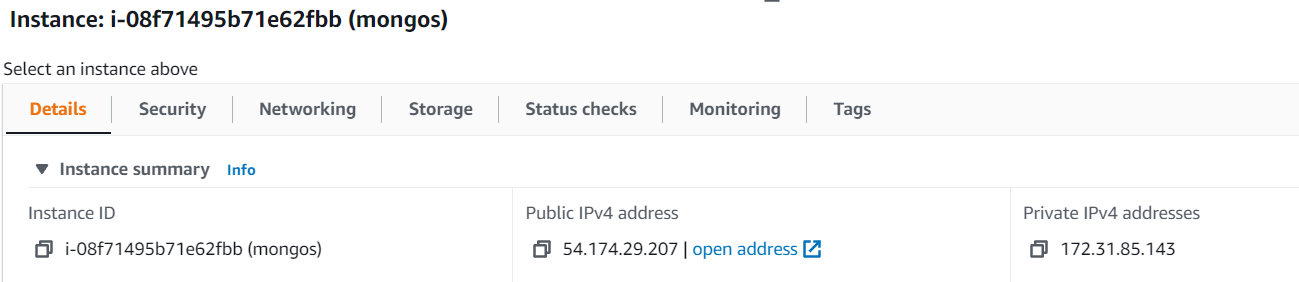
Mongos:

Public:

54.174.29.207

Private:

172.31.85.143



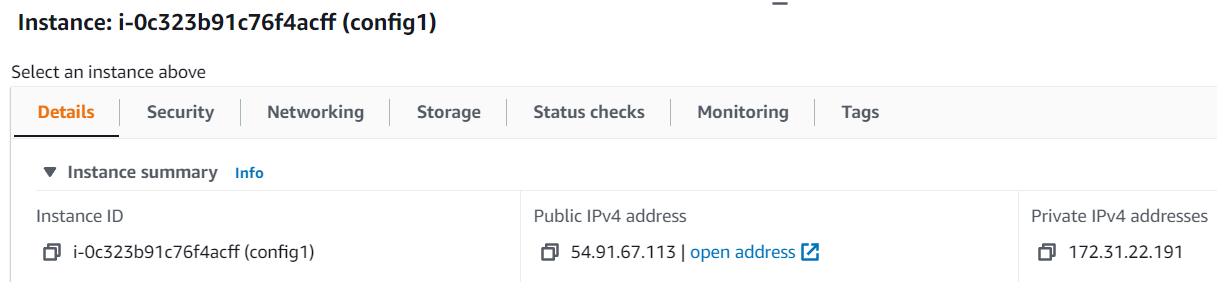
Config1:

Public

54.91.67.113

Private

172.31.24.206



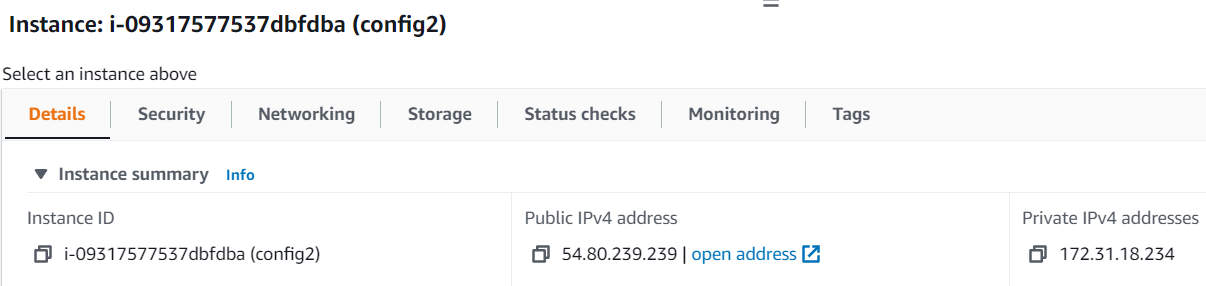
Config2:

Public

54.80.239.239

Private

172.31.18.234



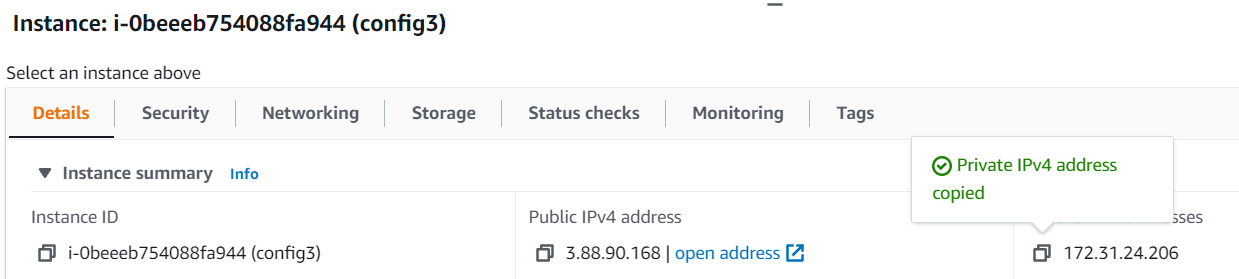
Config3:

Public

3.88.90.168

Private

172.31.24.206



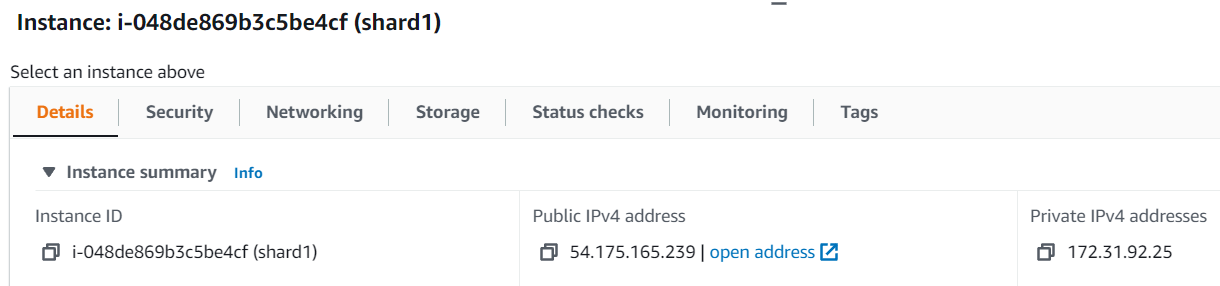
Shard1:

Public

54.175.165.239

Private

172.31.92.25



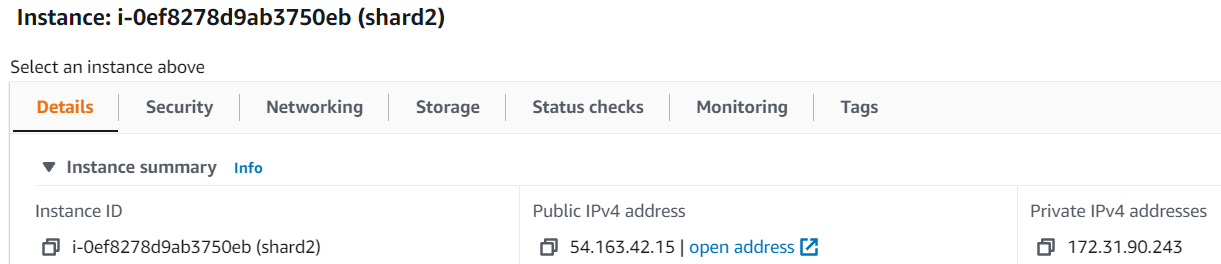
Shard2:

Public

54.163.42.15

Private

172.31.90.243



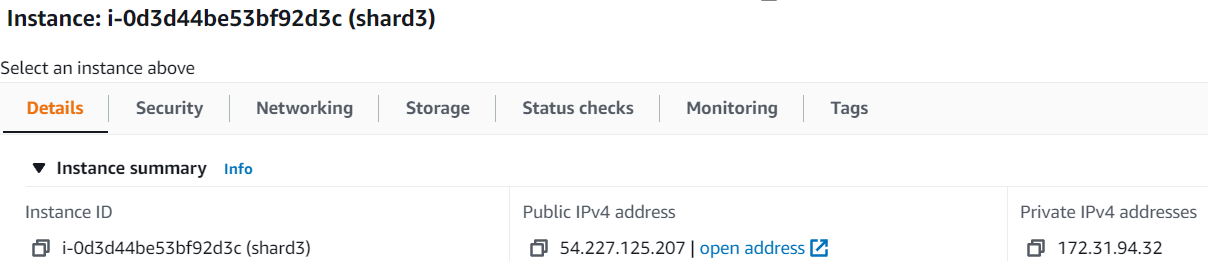
Shard3:

Public

54.227.125.207

Private

172.31.94.32



5. Replica sets in Config servers:

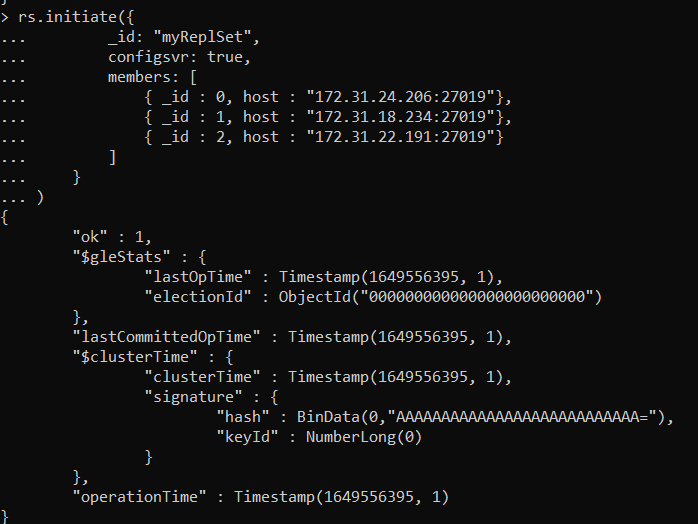
Created a replica set names myReplSet and store the database in data/db on port 27019 and binded the ip 0.0.0.0 with configsvr as config server

Used the following command on three config servers:

sudo mongod --configsvr --replSet myReplSet --dbpath /data/db --port 27019 --logpath /var/log/mongodb/mongod.log --bind\_ip 0.0.0.0 --fork

After that initiated the replica set by connecting to the mongo shell on port 27019

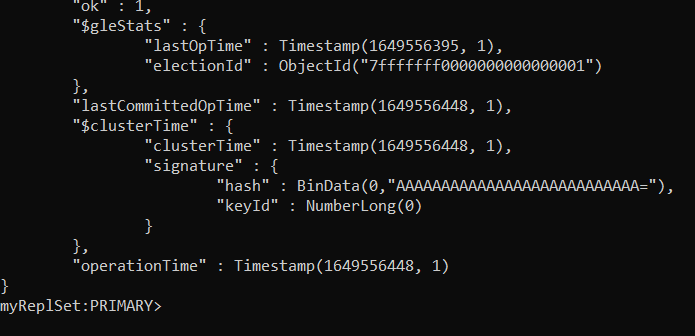
ScreenShot of rs.initiate():



Screenshot of replica set status after initialization:







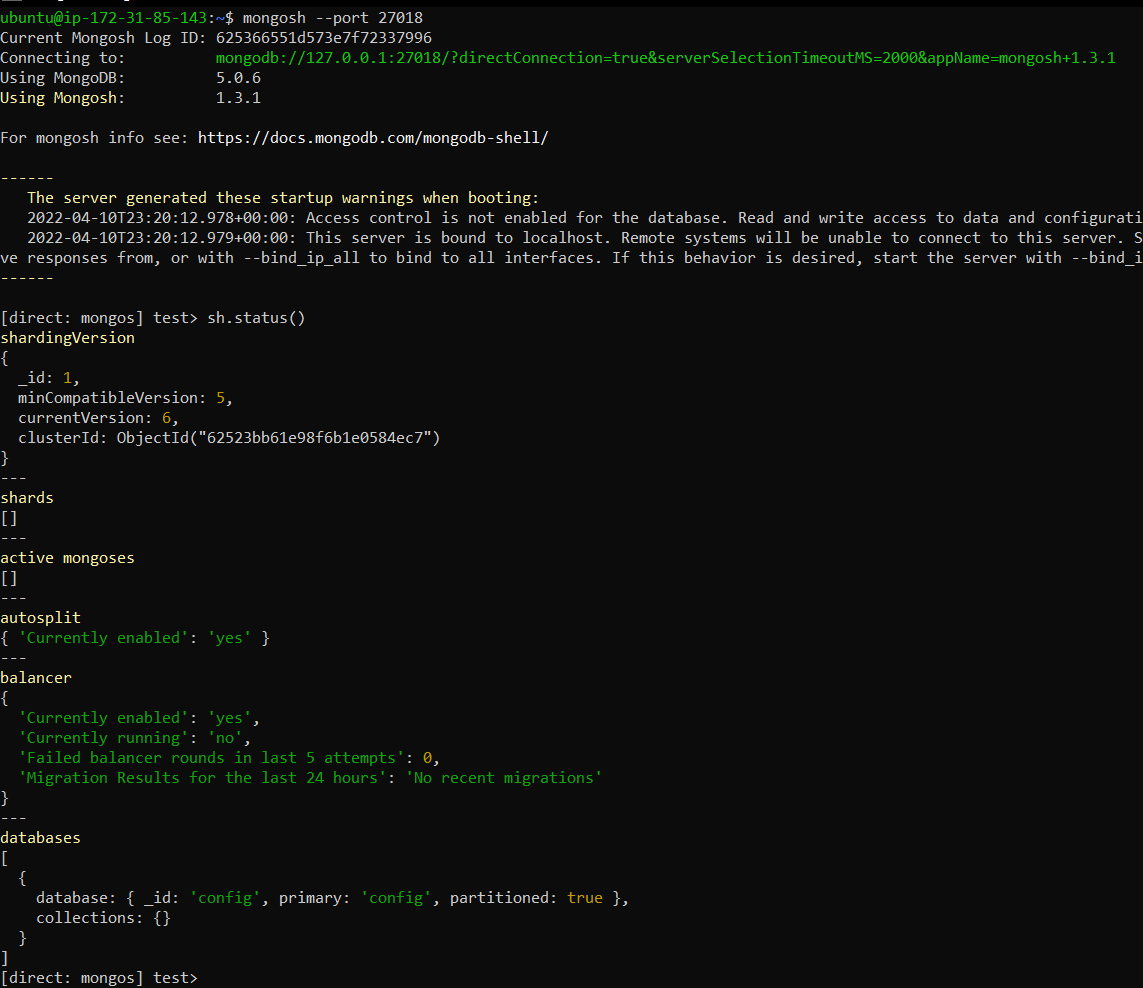
7. Connecting mongos to config server:

Using ssh logged on to the mongos server and connected the config servers using their private IPs and port to the mongos using the below command on port 27018

mongos --configdb myReplSet/172.31.24.206:27019,172.31.18.234:27019,172.31.22.191:27019 --port 27018

Then logged into mongo shell on port 27018 of mongos instance in the new terminal and got

The shard status before adding the shards.



8. Setting up the shards:

Created a replica sets named rs0, rs1, and rs2 and stored the database in data/db0, data/db1, and data/db2 on ports 27020, 27021, and 27022 and bound the ip 0.0.0.0 with shardsvr as a config server

Used the following command on each of three shard servers:

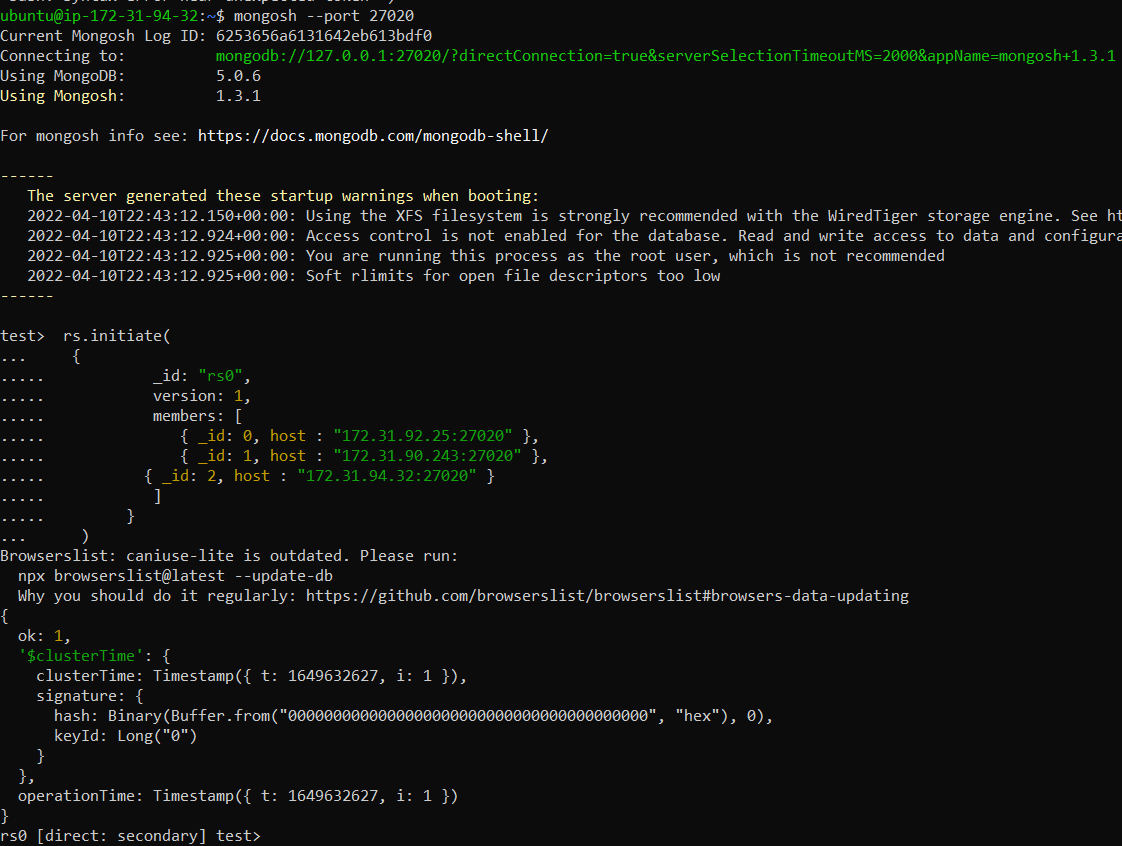
sudo mongod --shardsvr --replSet "rs0" --dbpath /data/db0 --port 27020 --bind\_ip 0.0.0.0

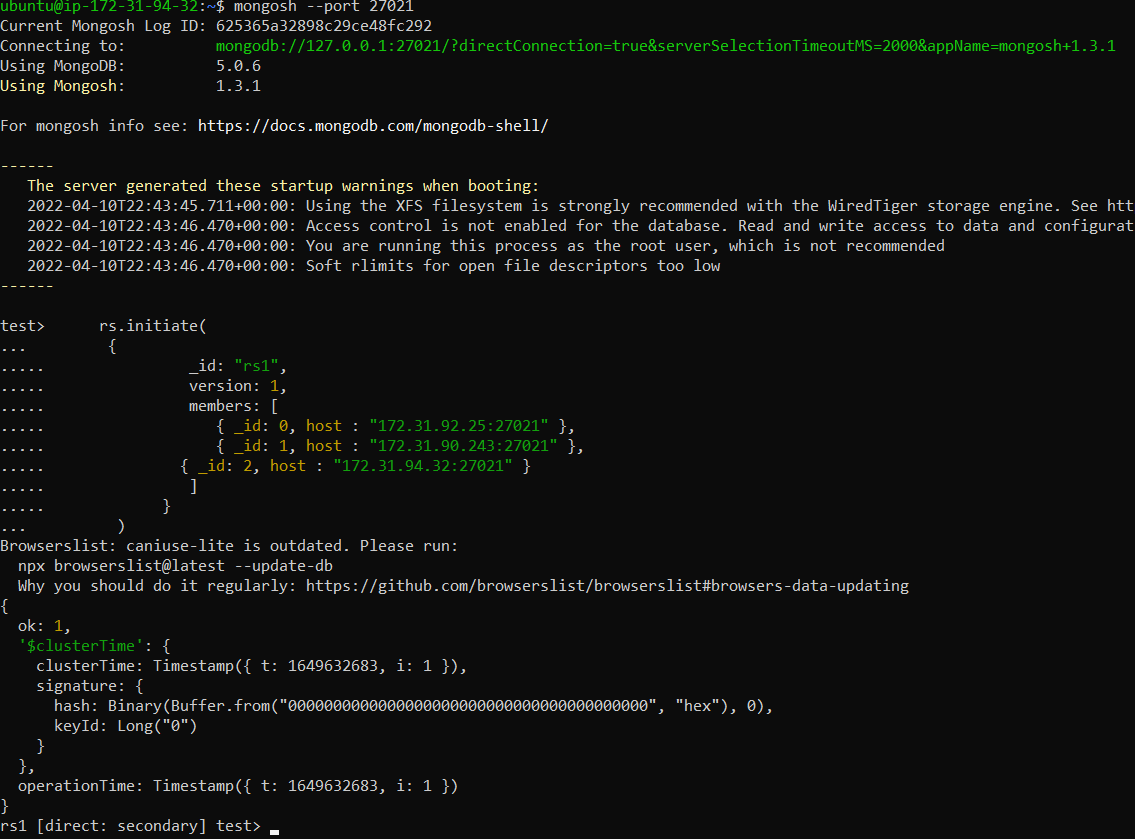
sudo mongod --shardsvr --replSet "rs1" --dbpath /data/db1 --port 27021 --bind\_ip 0.0.0.0

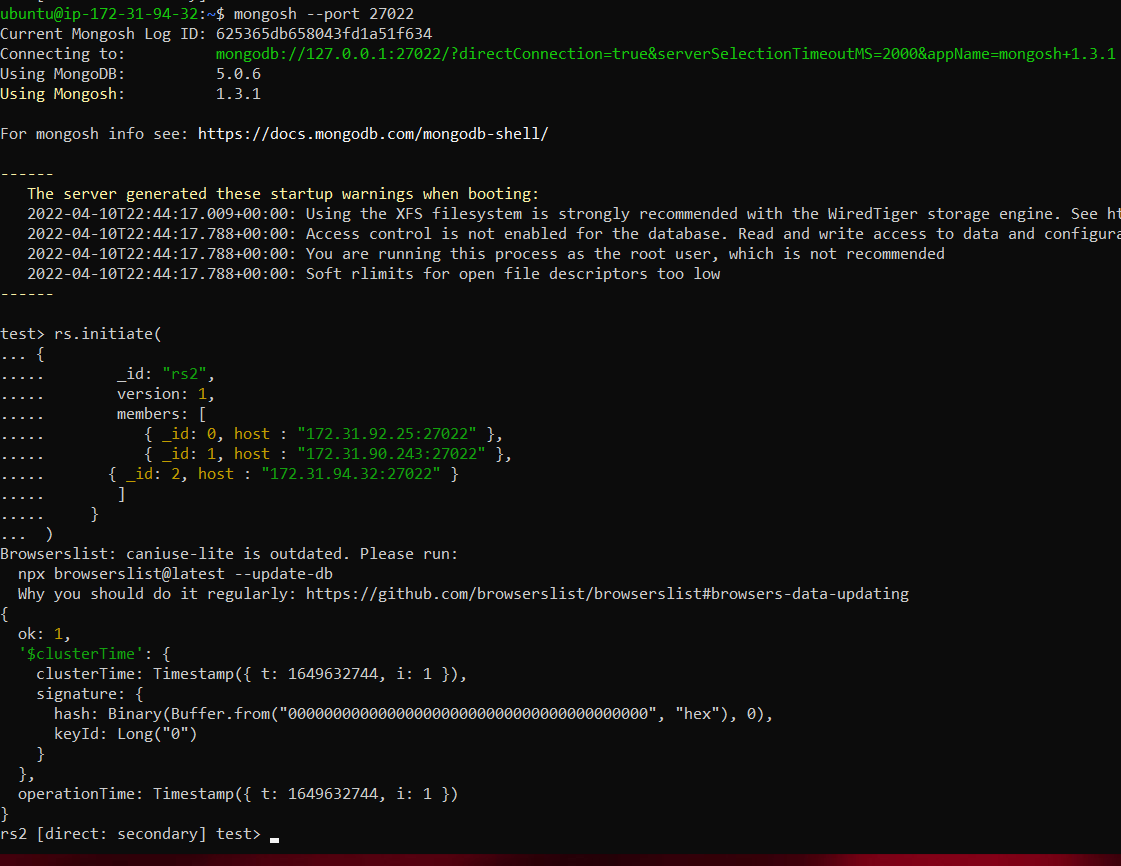
sudo mongod --shardsvr --replSet "rs2" --dbpath /data/db2 --port 27022 --bind\_ip 0.0.0.0

After that initiated the replica set by connecting to the mongo shell on ports 27020, 27021 and 27022

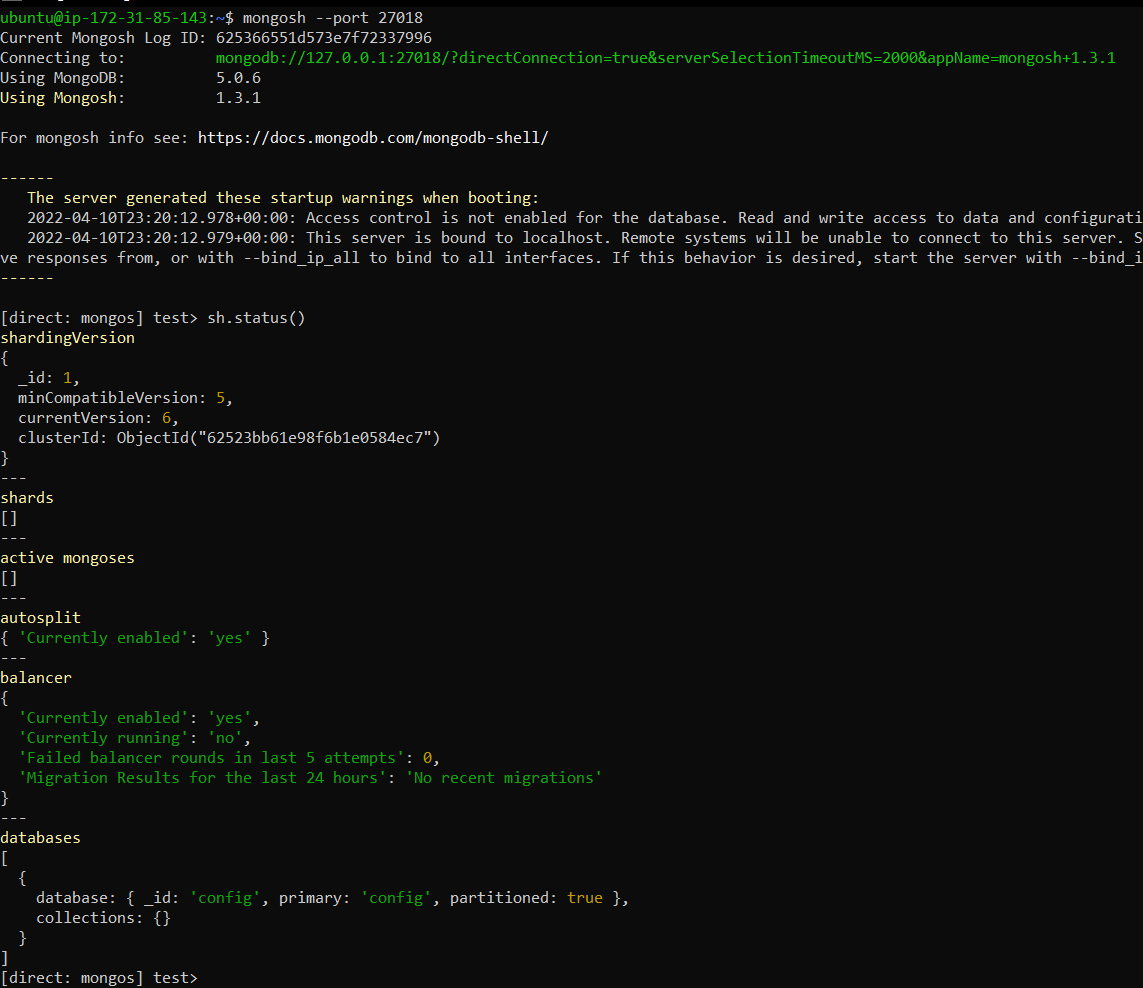
Proofs:







Shard status before adding the shards:



9. Adding the shards to mongos

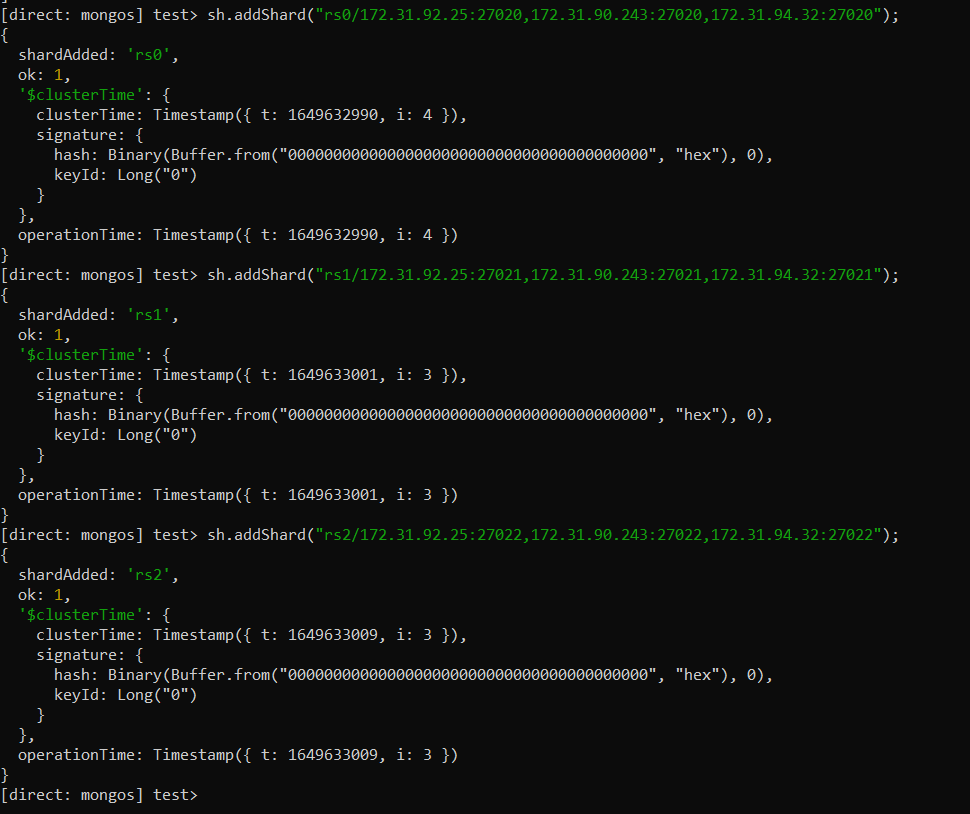
Connected to mongos using ssh and then ran mongo shell on port 27018 and then added the shard replica sets(replicaset name followed by their private Ips) using the following commands:

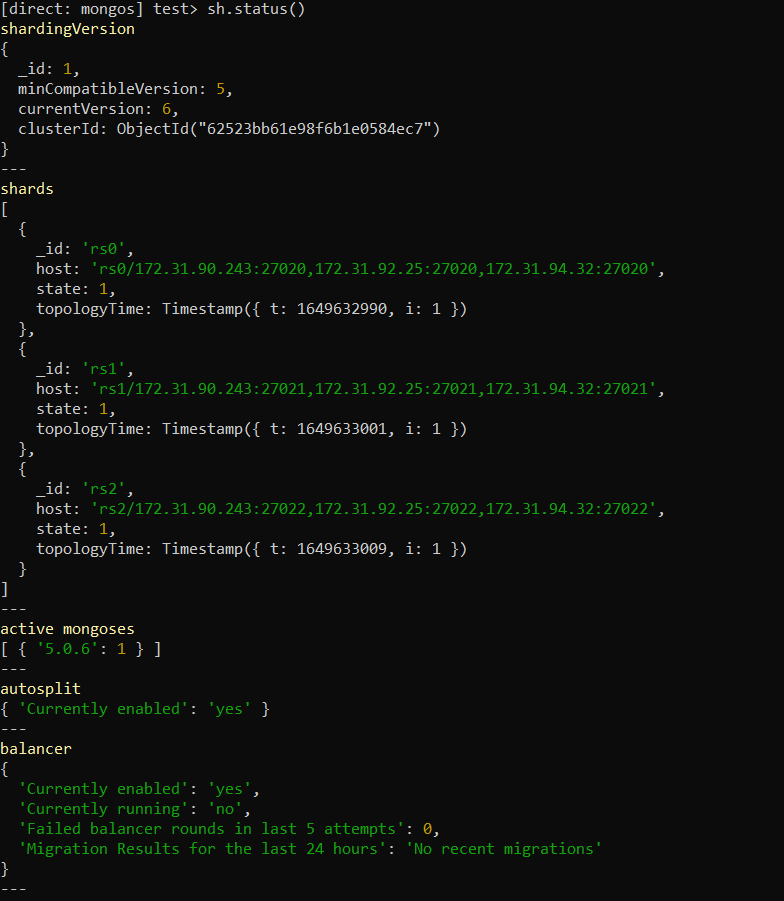
sh.addShard("rs0/172.31.92.25:27020,172.31.90.243:27020,172.31.94.32:27020");

sh.addShard("rs1/172.31.92.25:27021,172.31.90.243:27021,172.31.94.32:27021");

sh.addShard("rs2/172.31.92.25:27022,172.31.90.243:27022,172.31.94.32:27022");

Proof:

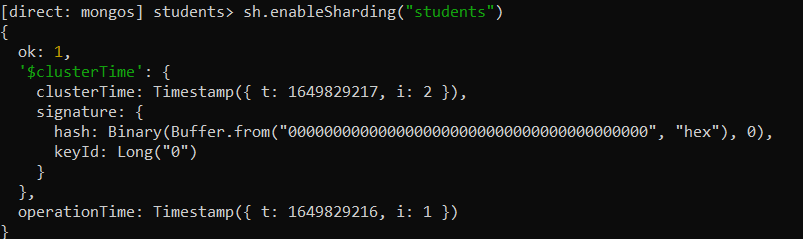


Shard status after adding:

10. Enabling Sharding:

Enable the sharding on the “students” database using the following command:

sh.enableSharding("students")

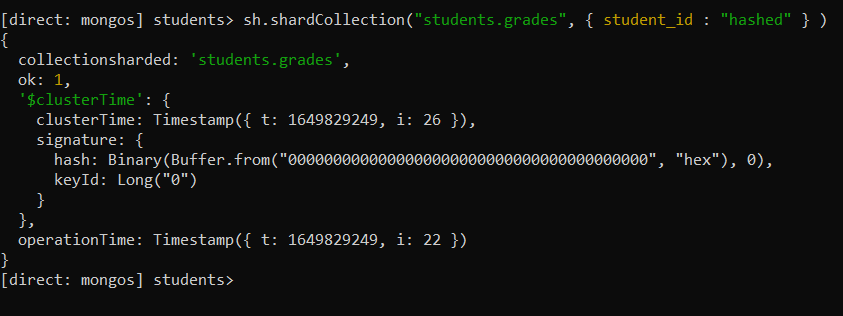


Then gave the details for the shard key

I have used the student\_id as my shard key on grades collection and “hashed” as my sharding strategy

Command:

sh.shardCollection("students.grades", { "student\_id" : "hashed" } )



11. Populating data

I have used the MongoDB sample training dataset grades.json for the midterm exam. I have procured this dataset from <https://github.com/neelabalan/mongodb-sample-dataset> repository.

Sample data in grades.json:

{ \_id: ObjectId("56d5f7eb604eb380b0d8d8d1"), student\_id: 0, scores: [ { type: 'exam', score: 6.267513889635468 }, { type: 'quiz', score: 23.8466262779109 }, { type: 'homework', score: 42.52700970652198 }, { type: 'homework', score: 76.22758120978754 } ], class\_id: 39 }

The dataset had 100000 documents.

Firstly, I connected to the mongos instance using ssh in a new terminal. Then cloned the git repository using the command

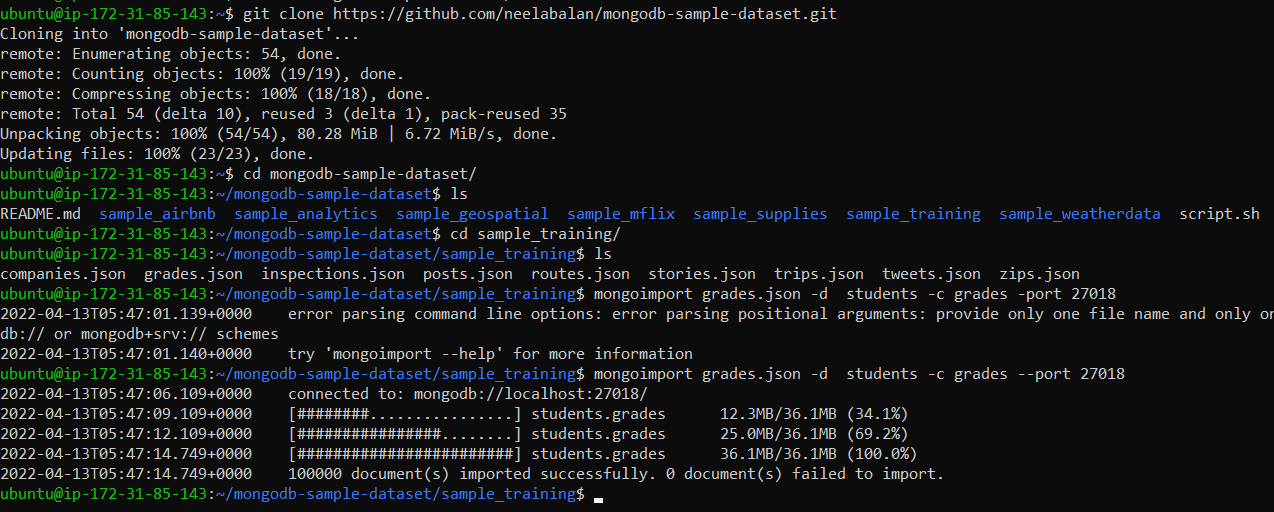
**git clone** [**https://github.com/neelabalan/mongodb-sample-dataset.git**](https://github.com/neelabalan/mongodb-sample-dataset.git)

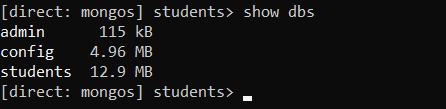
Then imported the data to the mongos using the command

**mongoimport grades.json -d students -c grades --port 27018**

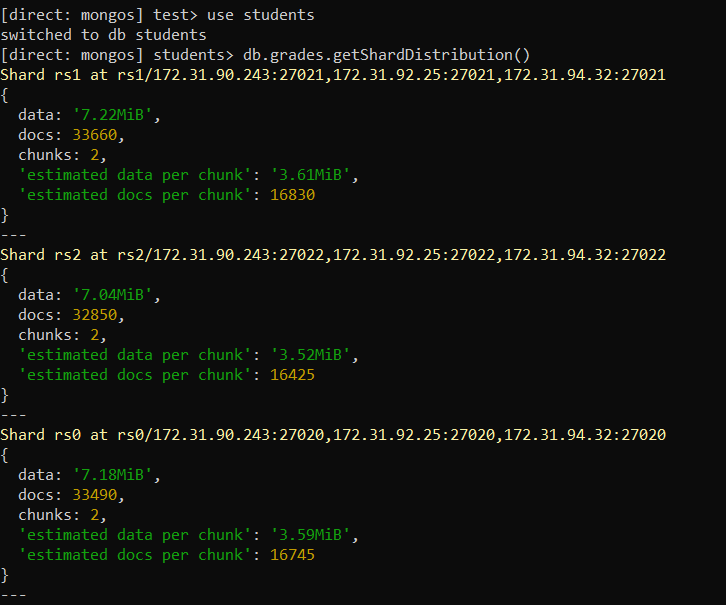
Database is student and collection is grades

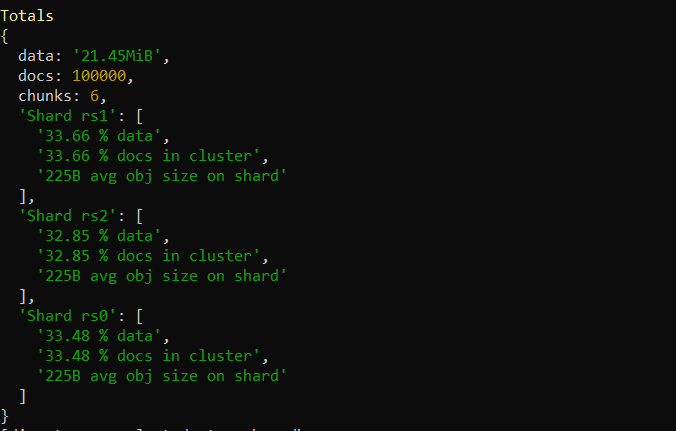
Proof:



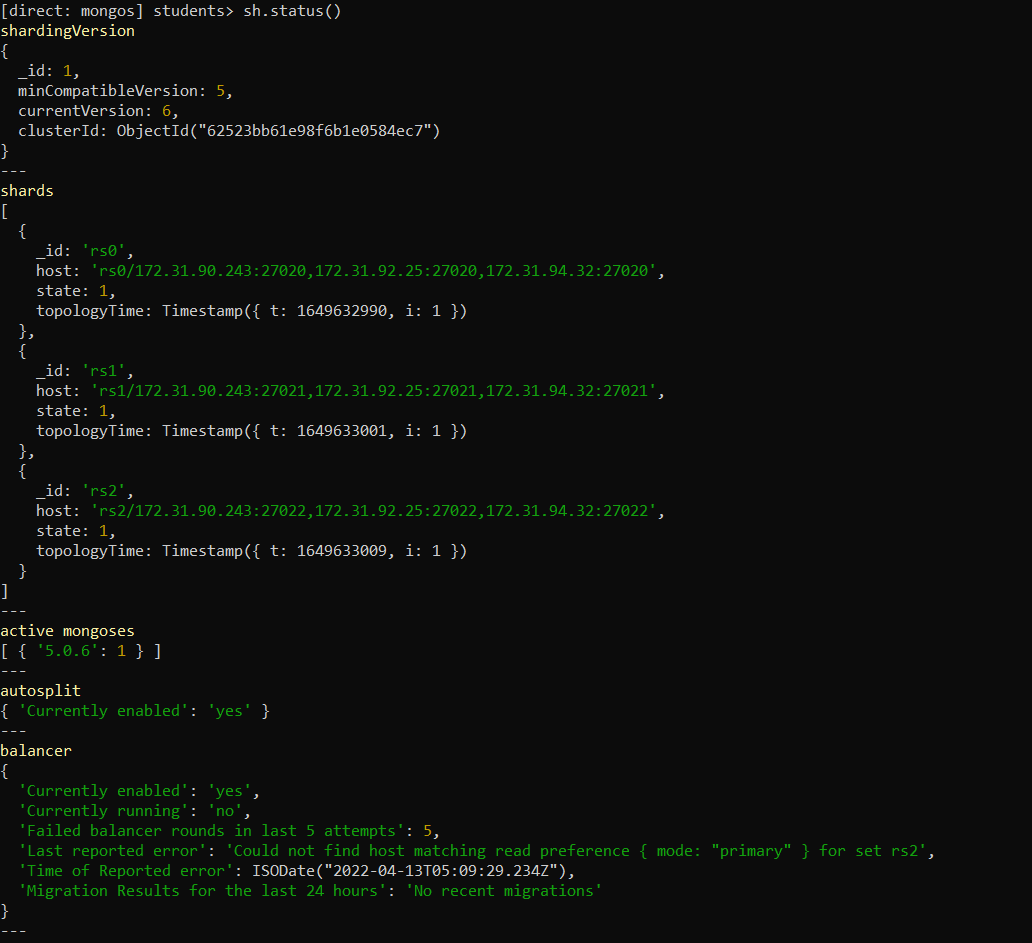


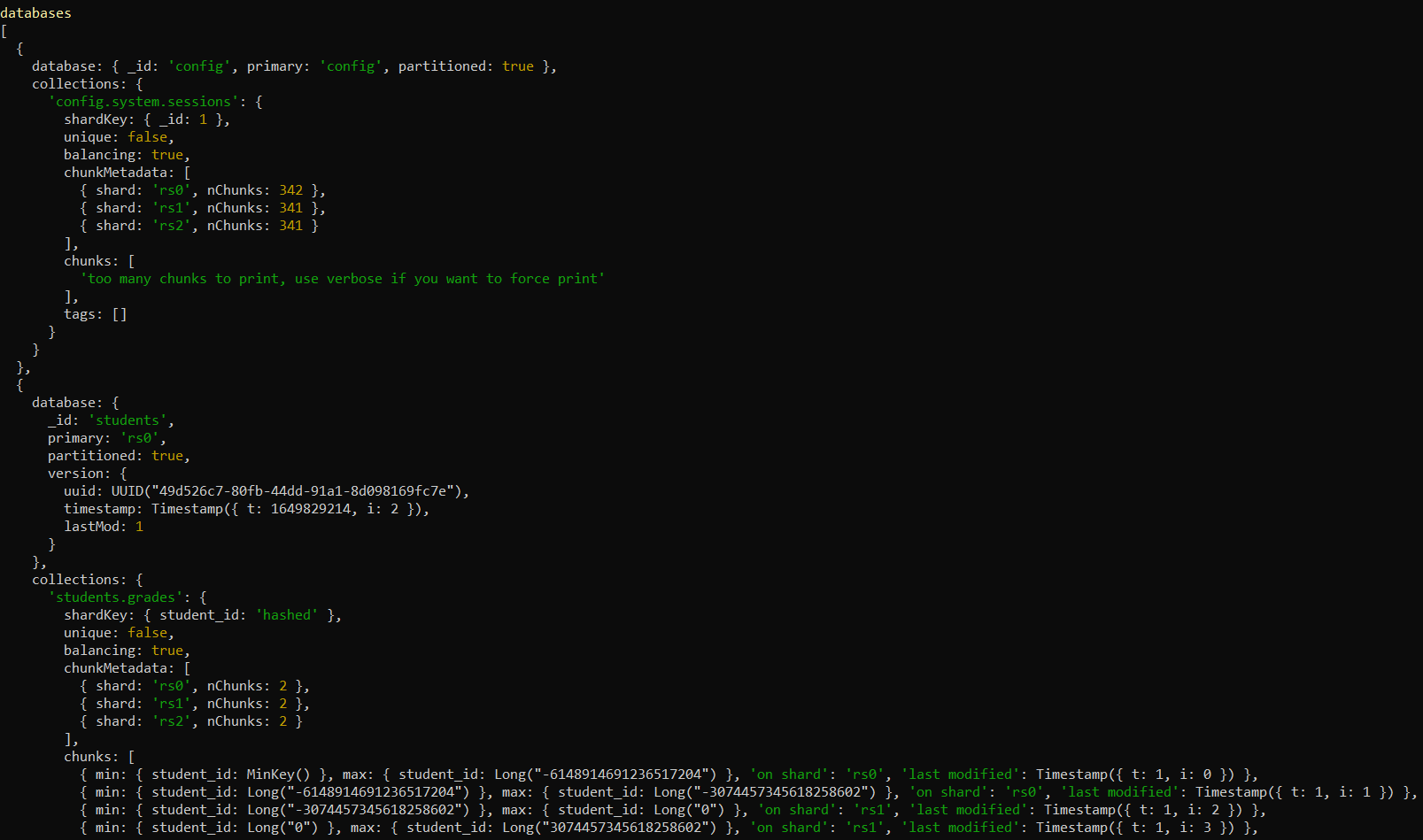
Dataset distribution in the shards

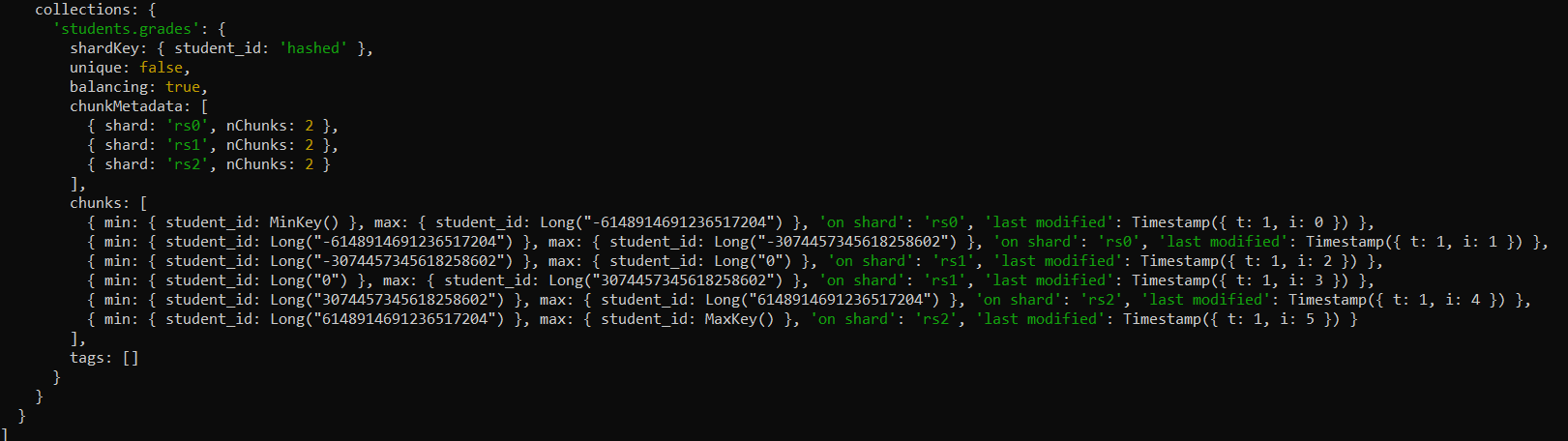




Shard status after data import







12. Queries

1. A range query to find documents in a given range.

Query:

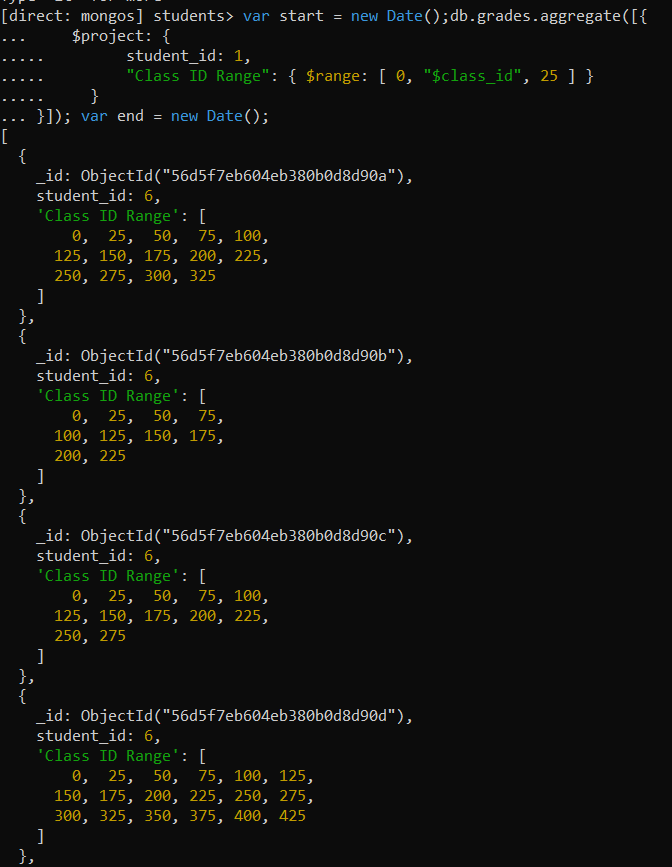
db.grades.aggregate([{

$project: {

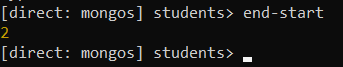
student\_id: 1,

"Class ID Range": { $range: [ 0, "$class\_id", 25 ] } }}])

Output:



Execution time:



Shard details:



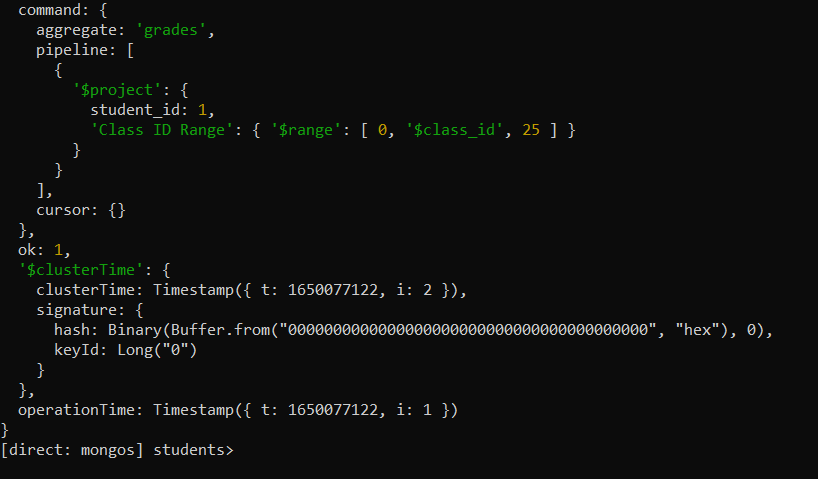












1. A query involving $elemMatch involving at least two conditions.

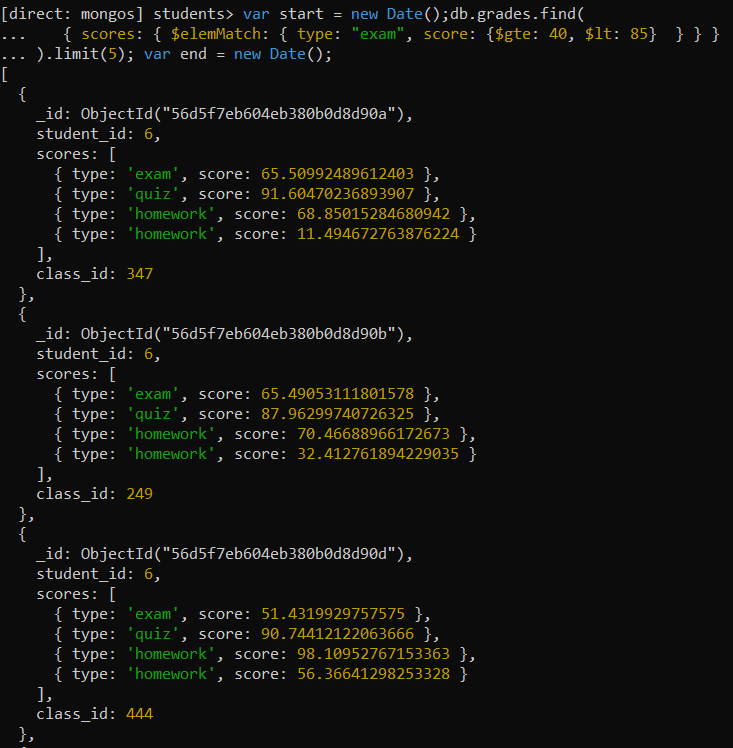
Query:

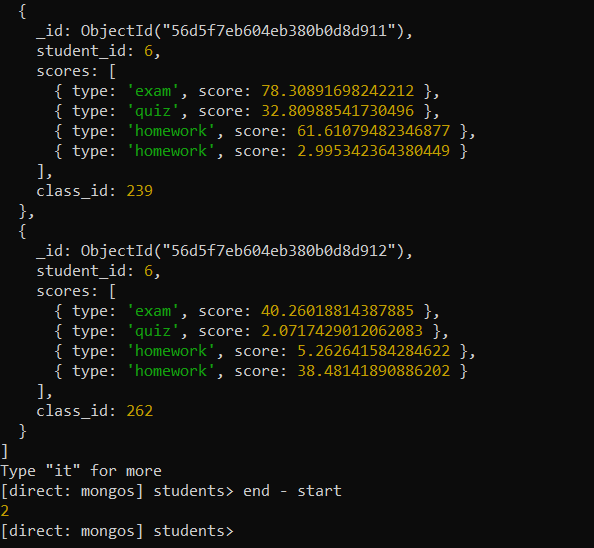
db.grades.find(

{ scores: { $elemMatch: { type: "exam", score: {$gte: 40, $lt: 85} } } }

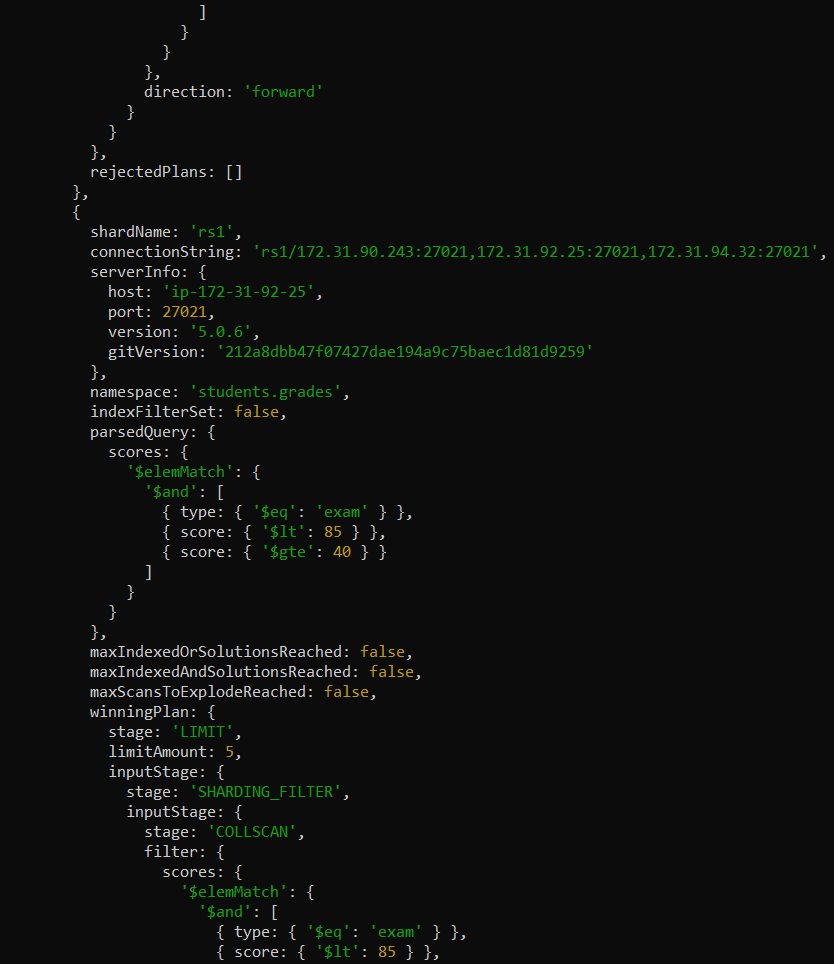
)

Output and Execution time:



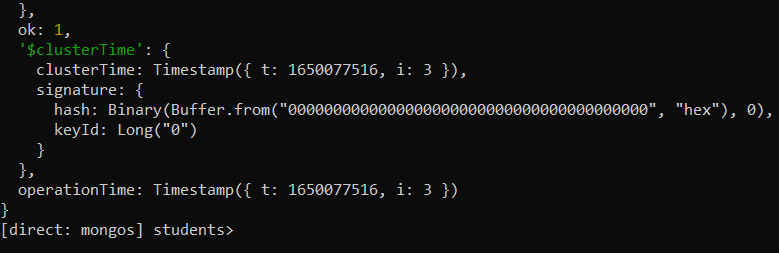


Shard Details:









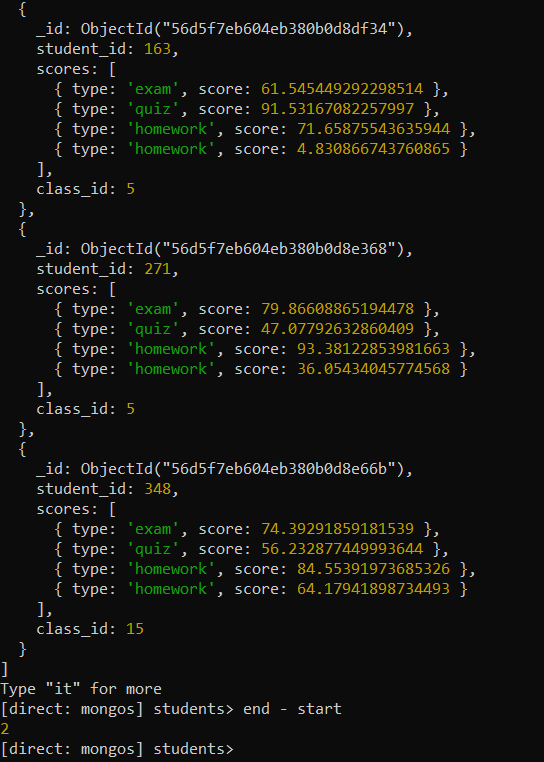
1. A query involving $in

Query:

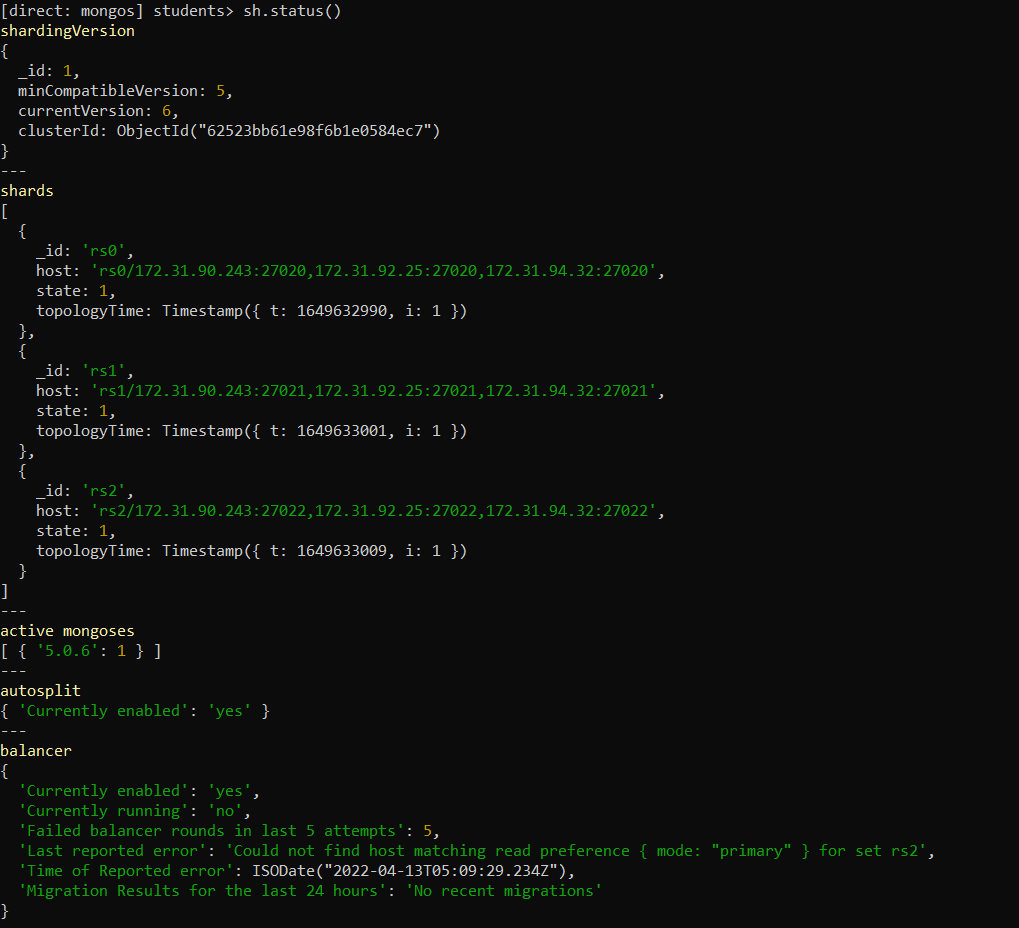
db.grades.find( { class\_id: { $in: [ 5, 15 ] } })

Output and Execution time:

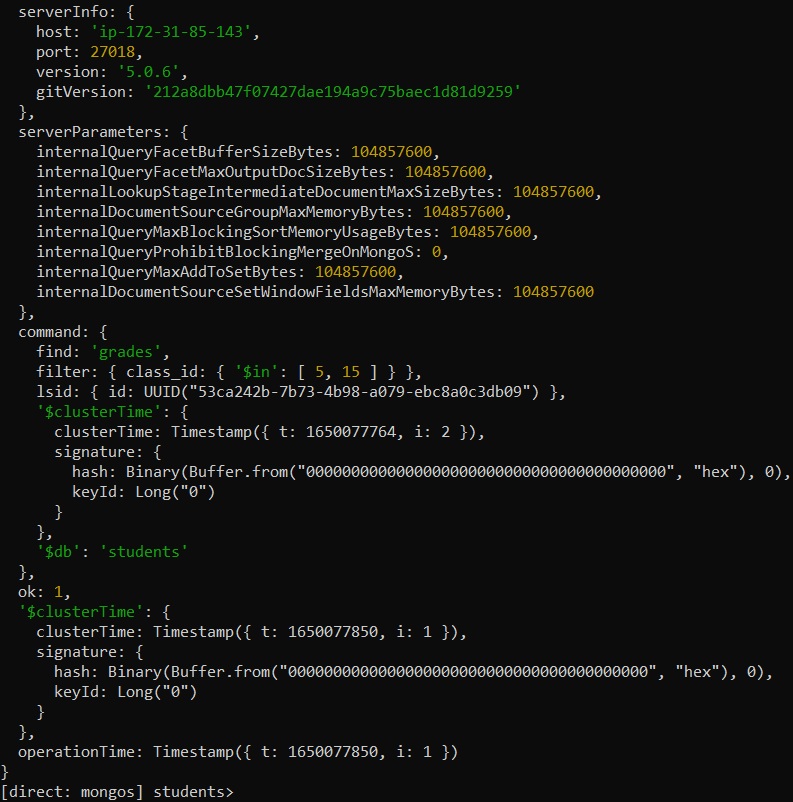




Shard Details:







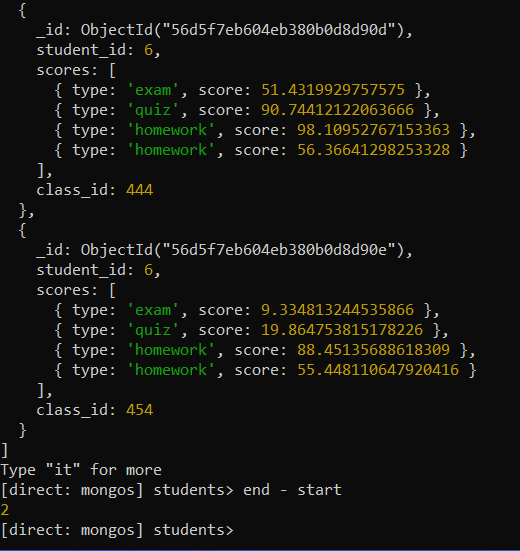
A query involving $nin

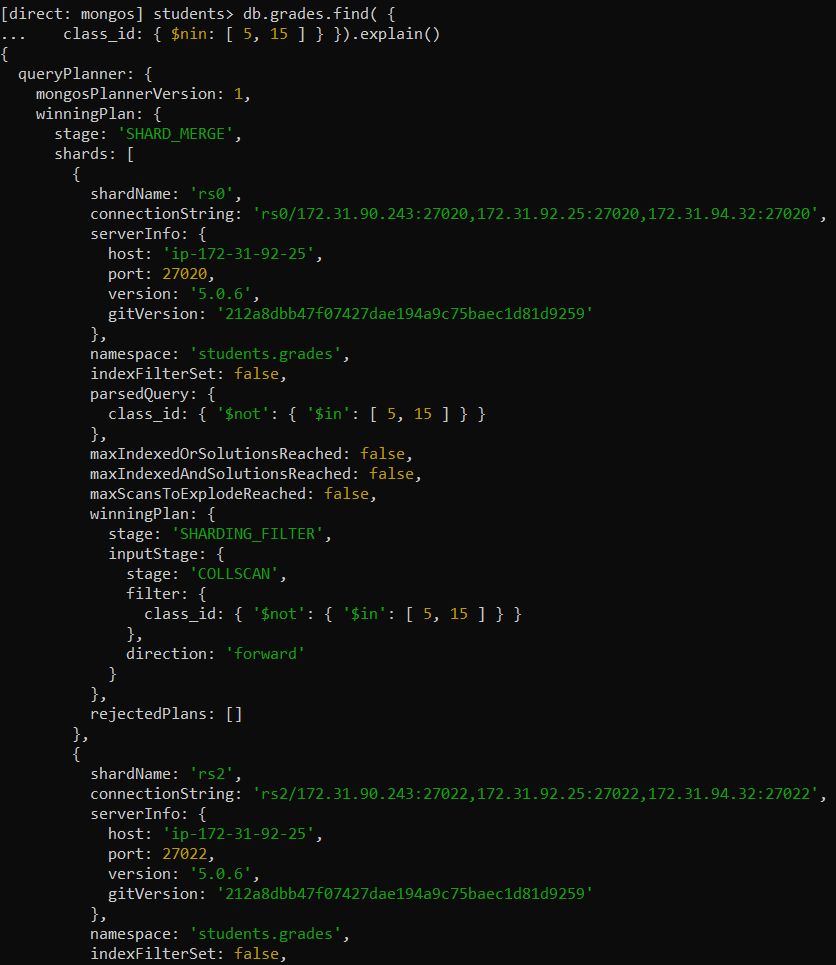
Query:

db.grades.find( { class\_id: { $nin: [ 5, 15 ] } })

Output and Execution time:





Shard details





A query involving $all

Query:

db.grades.find(

{ scores: {$all:[

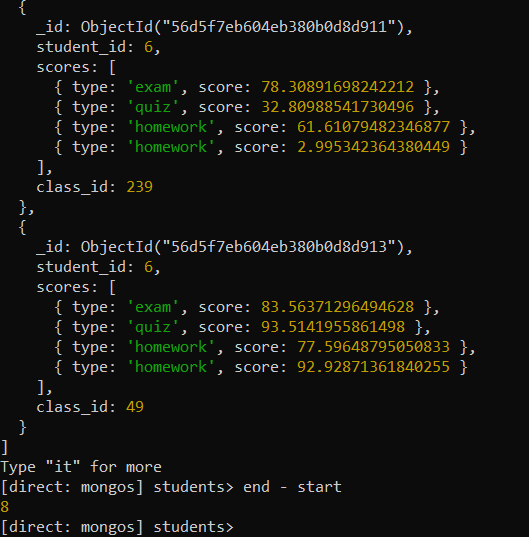
{ $elemMatch: { type: "exam", score: {$gte: 40, $lt: 85} } },

{ $elemMatch: { type: "homework", score: {$gte: 40, $lt: 85} } }

]}})

Output and Execution time:





Shard Details:

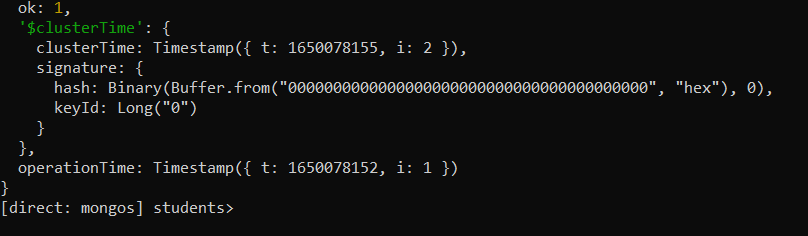












1. A query involving aggregate()

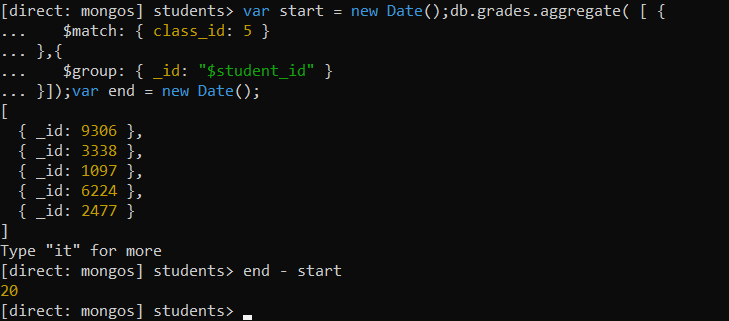
Query

db.grades.aggregate( [ {

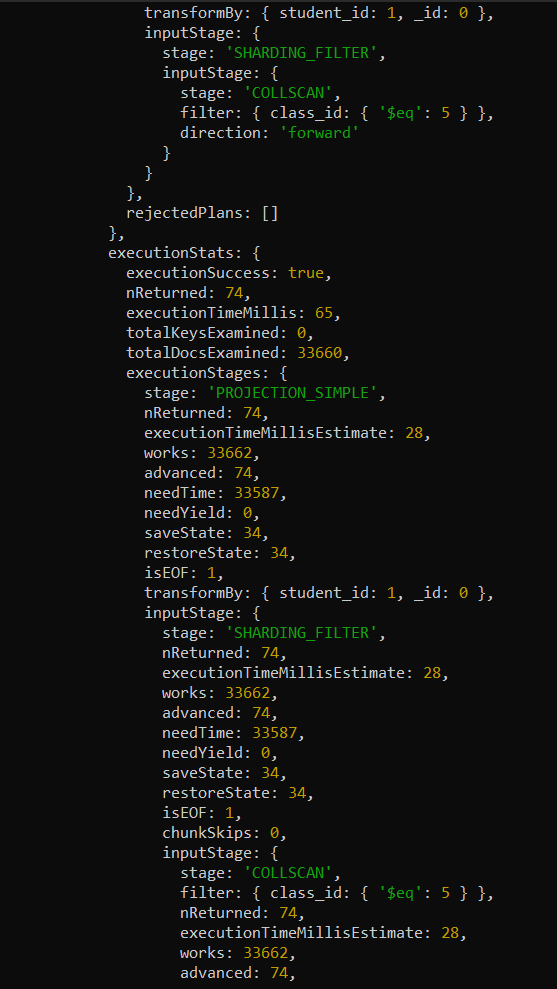
$match: { class\_id: 5 }},

{ $group: { \_id: "$student\_id" }}])

Output and execution time:



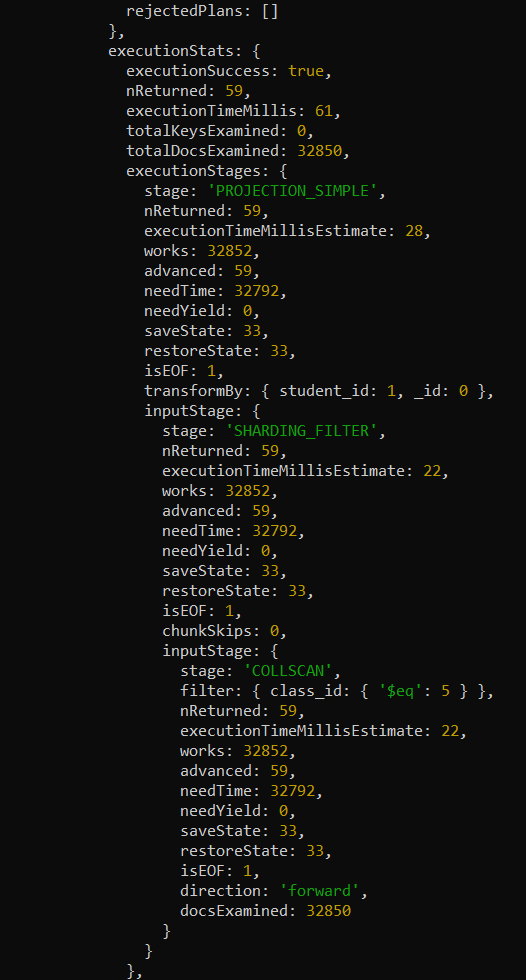
Shard details













1. A query involving mapReduce()

Query

var mapFunction = function(){emit(this.student\_id,1)}

var reduceFunction = function(key,values){return Array.sum(values)}

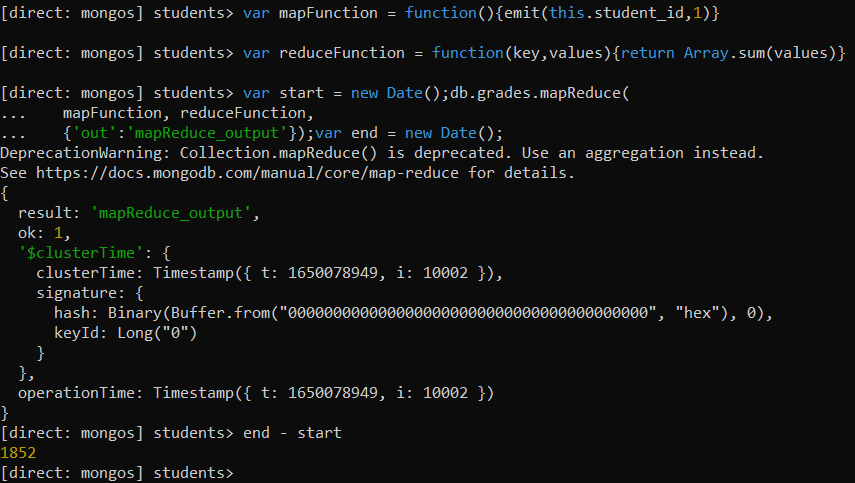
db.grades.mapReduce(

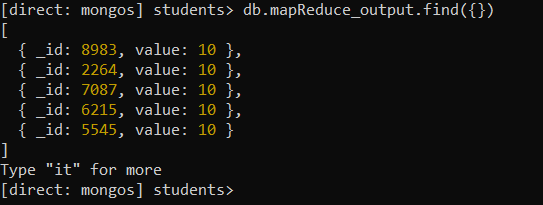
mapFunction, reduceFunction,

{'out':'mapReduce\_output'})

db.mapReduce\_output.find({})

Output and execution time:



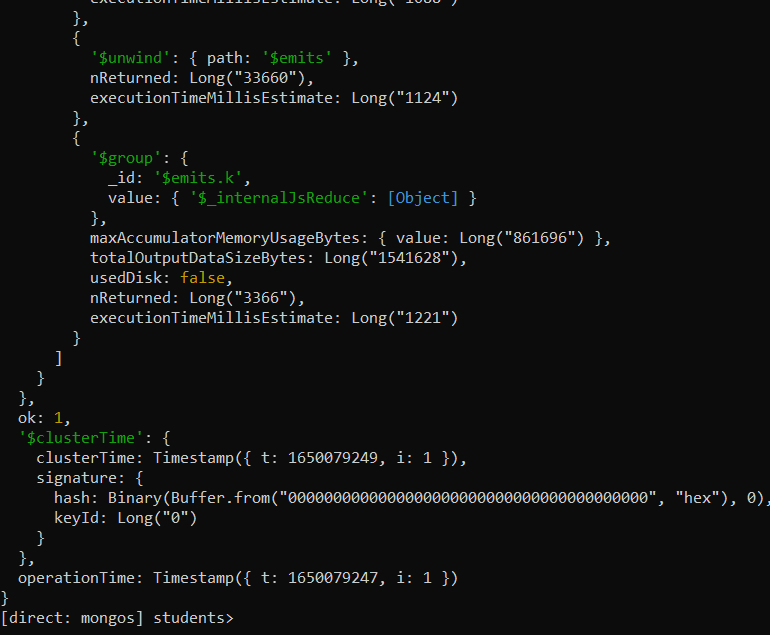


Shard details









1. A update

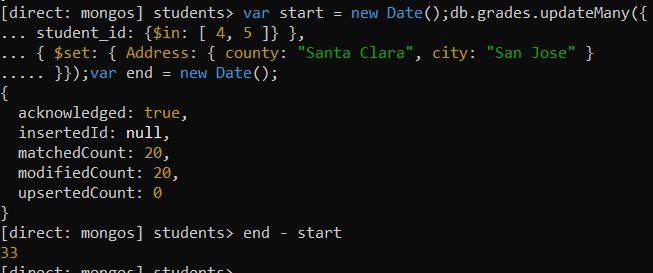
Query

db.grades.updateMany({

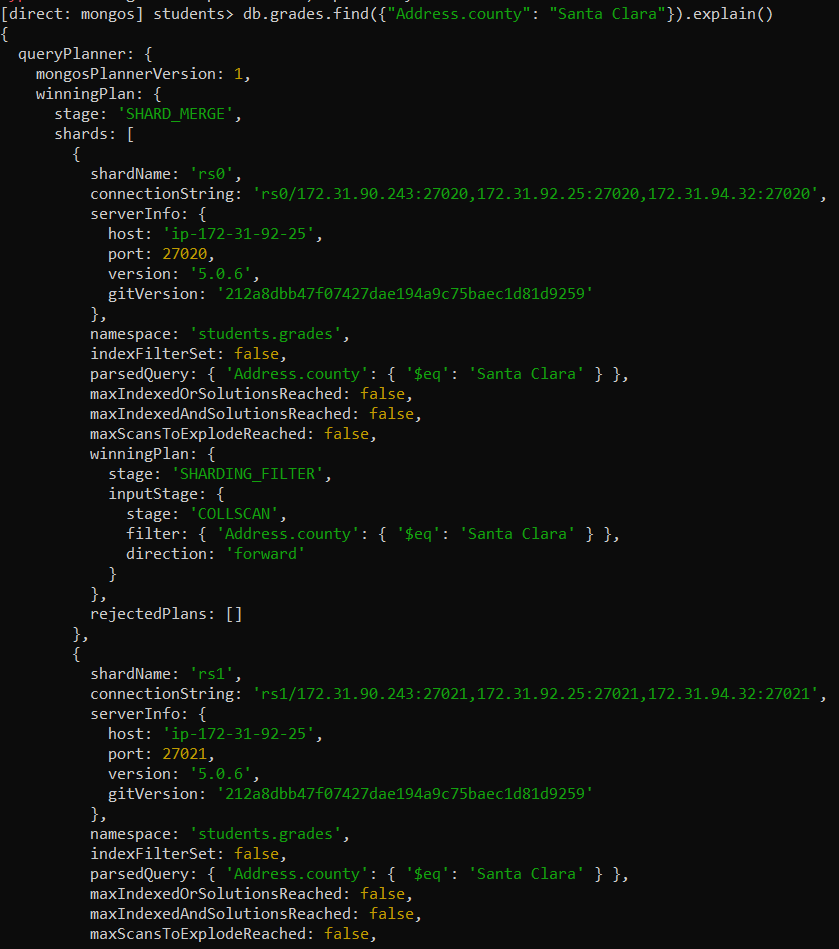
student\_id: {$in: [ 4, 5 ]} },

{ $set: { Address: { county: "Santa Clara", city: "San Jose" }}})

Output and execution time:



Shard details:

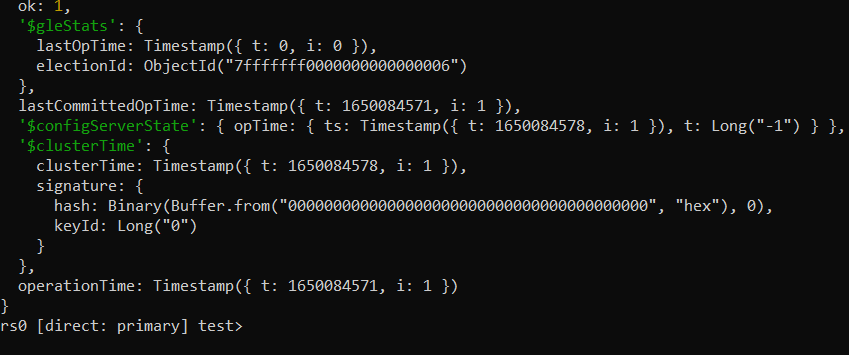






13. Shard Replication:

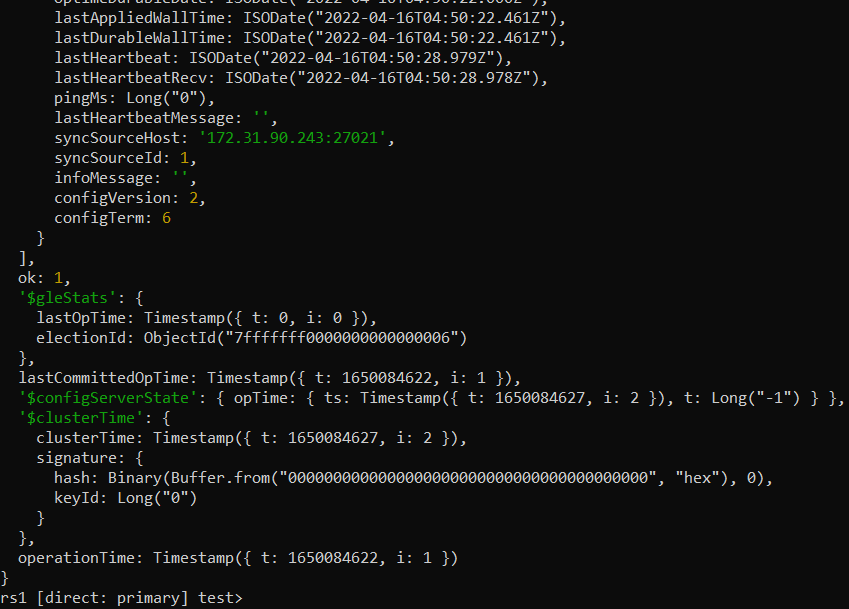
rs.status() of rs0



Rs1 rs.status():

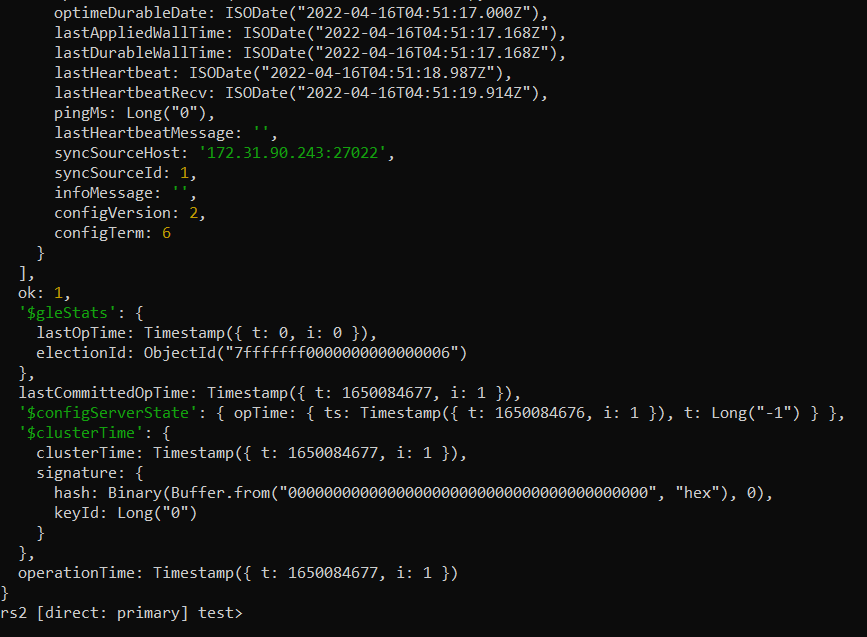




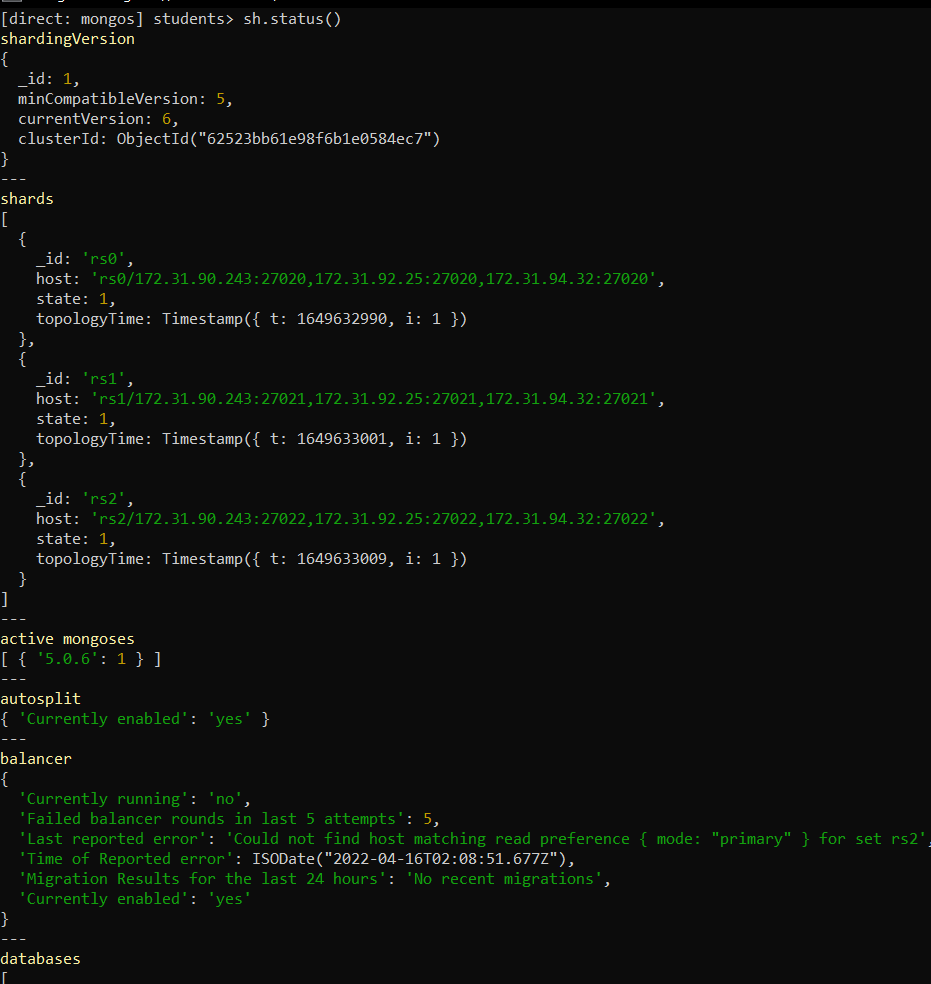


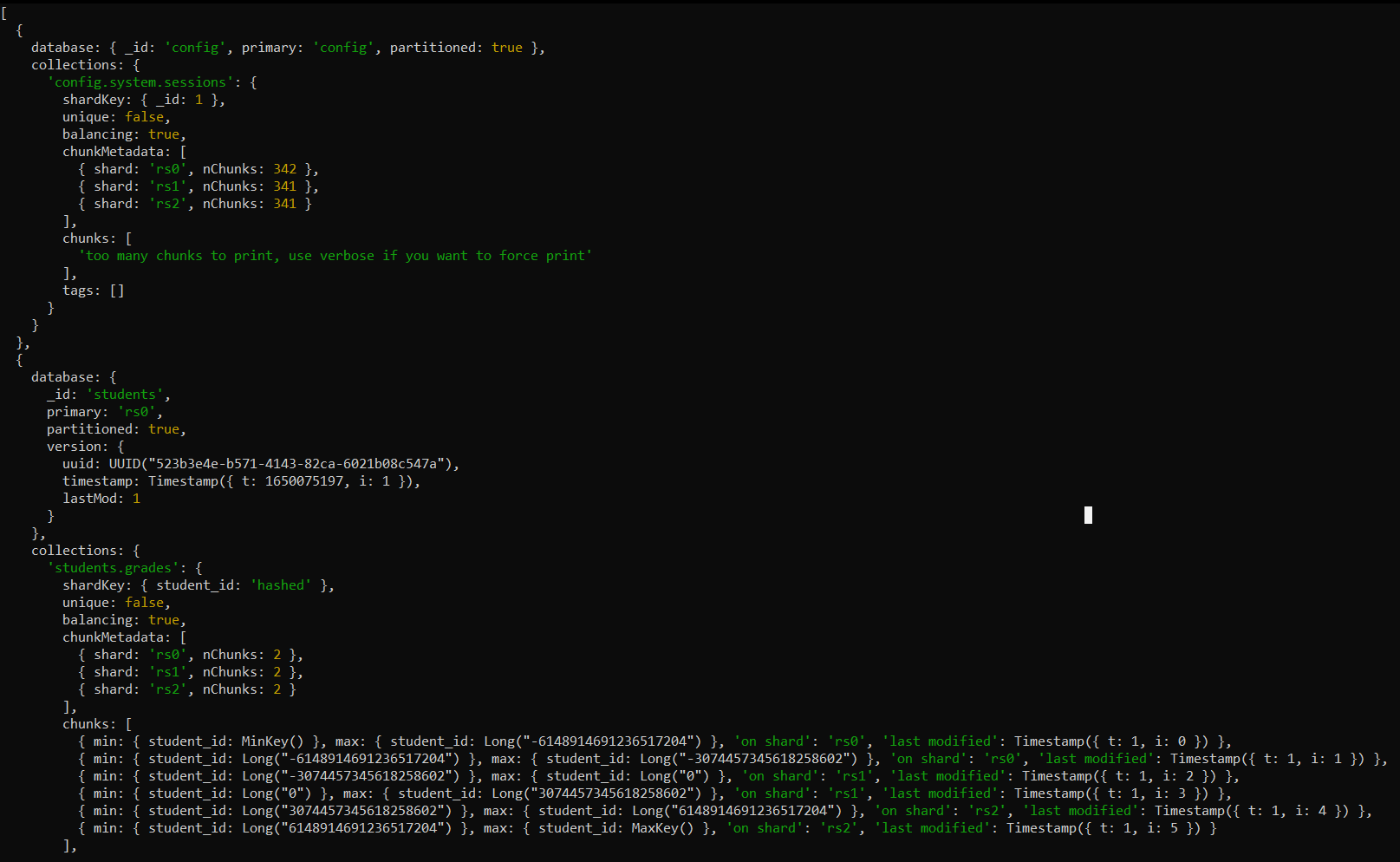
Rs2 rs.status()





Shard status:





14. List of hosts

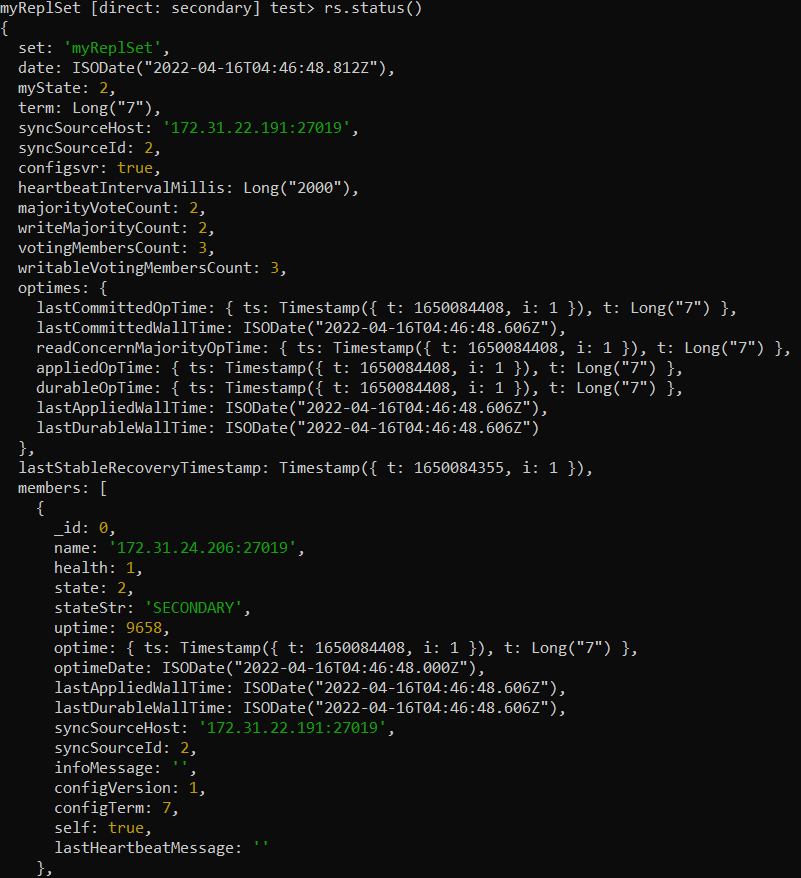
172.31.85.143:27018 -> mongos

172.31.22.191:27019 -> config server 1 -> myReplSet Primary

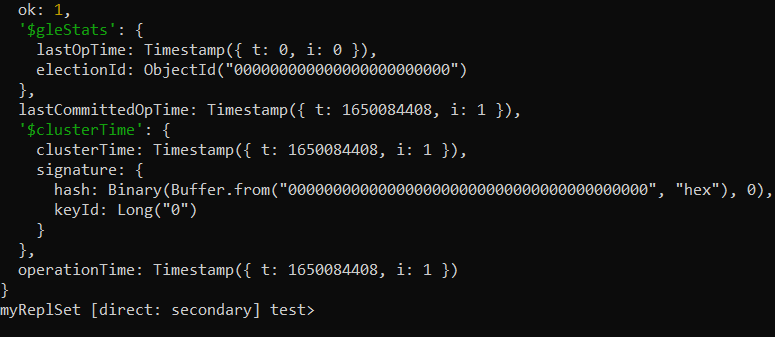
172.31.18.234:27019 -> config server 2 -> myReplSet Secondary

172.31.24.206:27019 -> config server 3 -> myReplSet Secondary

Proof:







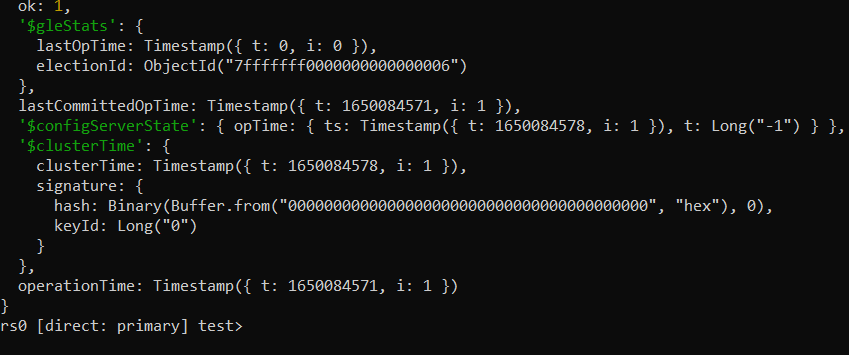
172.31.92.25:27020 -> shard server 1 -> rs0 Primary

172.31.90.243:27020 -> shard server 2 -> rs0 Secondary

172.31.94.32:27020 -> shard server 3 -> rs0 Secondary

Proof





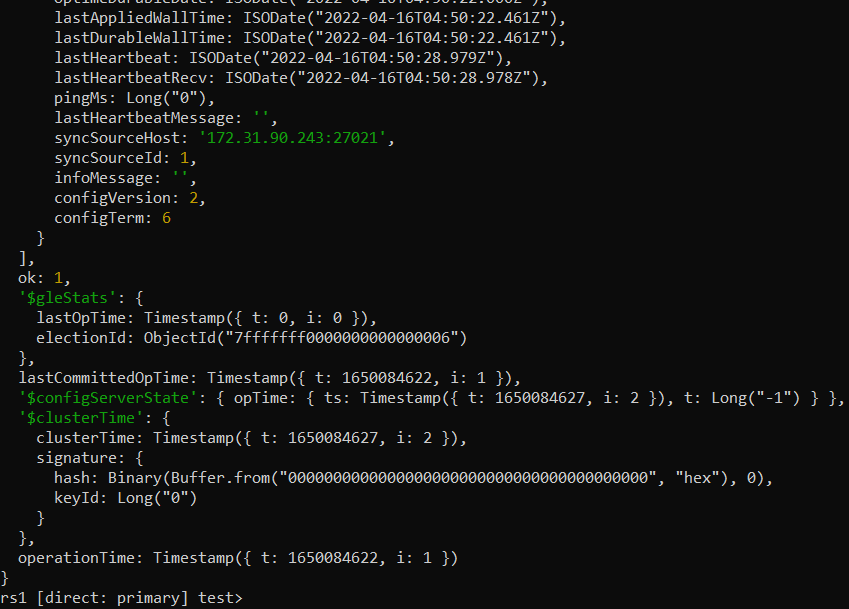
172.31.92.25:27021 -> shard server 1 -> rs1 Primary

172.31.90.243:27021 -> shard server 2 -> rs1 Secondary

172.31.94.32:27021 -> shard server 3 -> rs1 Secondary

Proof:





172.31.92.25:27022 -> shard server 1 -> rs2 Primary

172.31.90.243:27022 -> shard server 2 -> rs2 Secondary

172.31.94.32:27022 -> shard server 3 -> rs2 Secondary

Proof



