Spring 2017 CS185C: Introduction to NoSQL Databases
Midterm Report

Sidharth Mishra
Instructor: Dr. Kim

Tasks:

1. Set up 3 nodes in AWS

The following steps were taken to set up each of the 3 nodes in Amazon Web Services:

1. Logged into AWS `EC2 Management Console` [https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=instanceId]
2. Clicked on `Launch Instance`
3. Selected `Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-a58d0dc5` 64-Bit as the operating system for the node.
4. Selected the `General Purpose` `t2.micro` instance with 1 CPU and 1GB memory and clicked on `Review and Launch`.
5. Reviewed the storage and `ssh` port details and clicked on `Launch`.
6. When prompted to make certificate `.pem` file, created a new `.pem` file and downloaded it onto the host machine.
7. Changed the permissions on the `.pem` file to read-only i.e `chmod 400 <filename>.pem` on each of the 3 `.pem` files.
8. Waited for the instances to start before trying to `ssh` into them.

Instance Name	Instance ID	Instance Type	Public DNS (IPv4)	IPv4 Public IP	Key Name	Security Groups
machine1	i-05cec69dd84a1d737	t2.micro	ec2-54-245-184-166.us-west-2.compute.amazonaws.com	54.245.184.166	instance2	launch-wizard-1
machine2	i-018754f8dbbd3297e	t2.micro	ec2-54-148-246-51.us-west-2.compute.amazonaws.com	54.148.246.51	instance1	launch-wizard-4
machine3	i-05bef5c64661c91eb	t2.micro	ec2-52-39-43-165.us-west-2.compute.amazonaws.com	52.39.43.165	instance1	launch-wizard-3

2. Access these instances (nodes) through SSH

1. `ssh` into `machine1`

```
```

```
sidmishraw@Sidharths-MBP ~> ssh -i ~/Downloads/instance2.pem
ubuntu@ec2-54-245-184-166.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-54-245-184-166.us-west-2.compute.amazonaws.com (54.245.184.166)'
can't be established.
ECDSA key fingerprint is SHA256:h1h1SZs3w4qrEpCNpo1+DJgqJUjv6+xy84H85OM79Tg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-245-184-166.us-west-2.compute.amazonaws.com,54.245.184.166'
(ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-66-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

22 packages can be updated.
0 updates are security updates.


*** System restart required ***
Last login: Wed Mar 15 18:55:26 2017 from 107.204.169.68
ubuntu@ec2-54-202-76-188:~$
```

2. `ssh` into `machine2`

```
sidmishraw@Sidharths-MBP ~> ssh -i ~/Downloads/instance1.pem
ubuntu@ec2-54-148-246-51.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-54-148-246-51.us-west-2.compute.amazonaws.com (54.148.246.51)'
can't be established.
ECDSA key fingerprint is SHA256:4u4H8r43q/BStOMPjXiGIvRigSQhz1nN94DUANE+7bo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-148-246-51.us-west-2.compute.amazonaws.com,54.148.246.51'
(ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-64-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

8 packages can be updated.
0 updates are security updates.
```

```
*** System restart required ***
Last login: Sat Mar 11 07:01:03 2017 from 107.204.169.68
ubuntu@ec2-52-35-55-89:~$ ls
data
ubuntu@ec2-52-35-55-89:~$ pwd
/home/ubuntu
ubuntu@ec2-52-35-55-89:~$
```

3. `ssh` into `machine3`

```
sidmishraw@Sidharths-MBP ~> ssh -i ~/Downloads/instance1.pem
ubuntu@ec2-52-39-43-165.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-52-39-43-165.us-west-2.compute.amazonaws.com (52.39.43.165)'
can't be established.
ECDSA key fingerprint is SHA256:PSpwUufv1J/Jd76q3tUPLRkMfziK9AYR3Td31T9+IyQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-52-39-43-165.us-west-2.compute.amazonaws.com,52.39.43.165'
(ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-66-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

23 packages can be updated.
0 updates are security updates.


*** System restart required ***
Last login: Sat Mar 11 07:21:17 2017 from 107.204.169.68
ubuntu@ec2-52-41-163-132:~$
```

3. Install MongoDB in each node

Added mongodb repository for versions 3.4.x to apt (package manager) and installed it using the commands below:

1.
   ```
   sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
   0C49F3730359A14518585931BC711F9BA15703C6
   ```

2.
   ```
   echo "deb [ arch=amd64 ] http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.4
   ```

    multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
    ```
3. ```
   sudo apt-get update
   sudo apt-get install -y mongodb-org
   ```

The above commands were executed on each of the 3 instances in AWS.

machine1:

```
ubuntu@ec2-54-202-76-188:~$ mongod --version
db version v3.4.2
git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
OpenSSL version: OpenSSL 1.0.2g  1 Mar 2016
allocator: tcmalloc
modules: none
build environment:
    distmod: ubuntu1404
    distarch: x86_64
    target_arch: x86_64
ubuntu@ec2-54-202-76-188:~$
```

machine2:

```
ubuntu@ec2-52-35-55-89:~$ mongod --version
db version v3.4.2
git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
OpenSSL version: OpenSSL 1.0.2g  1 Mar 2016
allocator: tcmalloc
modules: none
build environment:
    distmod: ubuntu1404
    distarch: x86_64
    target_arch: x86_64
ubuntu@ec2-52-35-55-89:~$
```

machine3:

```
ubuntu@ec2-52-41-163-132:~$ mongod --version
db version v3.4.2
git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
OpenSSL version: OpenSSL 1.0.2g  1 Mar 2016
allocator: tcmalloc
modules: none
build environment:
    distmod: ubuntu1404
    distarch: x86_64
```

```
    target_arch: x86_64
ubuntu@ec2-52-41-163-132:~$
```

4. Create a directory to store database in each node

Created `midterm/db/data` directory in `/home/ubuntu` directory on each of the 3 nodes in AWS using the command:
```mkdir -p midterm/db/data```

machine1:
```

ubuntu@ec2-54-202-76-188 ~> pwd
/home/ubuntu
ubuntu@ec2-54-202-76-188 ~> ls
data/  logs/  midterm/  old_data/  workspace/
ubuntu@ec2-54-202-76-188 ~> ls midterm/
db/
ubuntu@ec2-54-202-76-188 ~> ls midterm/db/
config/  data/
ubuntu@ec2-54-202-76-188 ~>
```

machine2:
```

ubuntu@ec2-52-35-55-89 ~> pwd
/home/ubuntu
ubuntu@ec2-52-35-55-89 ~> ls
data/  logs/  midterm/  old_data/
ubuntu@ec2-52-35-55-89 ~> ls midterm/
db/
ubuntu@ec2-52-35-55-89 ~> ls midterm/db/
config/  data/
ubuntu@ec2-52-35-55-89 ~>
```

machine3:
```

ubuntu@ec2-52-41-163-132 ~> pwd
/home/ubuntu
ubuntu@ec2-52-41-163-132 ~> ls
arbiter_data/  data/  logs/  midterm/  old_data/
ubuntu@ec2-52-41-163-132 ~> ls midterm/
db/
ubuntu@ec2-52-41-163-132 ~> ls midterm/db/
config/  data/
ubuntu@ec2-52-41-163-132 ~>
```

~~~~
```
~~~~

5. Specify Public and Private IP Addresses of three AWS instances used in your solution.

~~~~
Note: AWS changes the public IP addresses/DNS(s) for free-tier machines after every
restart.(Unless you have an Elastic IP setup)
~~~~

Instance Name	Instance ID	Public IPv4 IP	Private IPv4 IP	Launch Time
machine1	i-05cec69dd84a1d737	54.245.184.166	172.31.20.85	March 31, 2017 at 3:55:52 PM UTC-7
machine2	i-018754f8dbbd3297e	54.148.246.51	172.31.27.128	March 31, 2017 at 3:56:28 PM UTC-7
machine3	i-05bef5c64661c91eb	52.39.43.165	172.31.24.48	March 31, 2017 at 3:56:28 PM UTC-7

The public IP for each of the instances was obtained from the EC2 Management Console and the private IP was obtained from the `inet addr` of `eth0` by using the `ifconfig` command from the terminal after doing a `ssh` into the instances. Alternatively, the private IP can also be obtained by using the `hostname -I` command.

6. Setup and launch three config servers

~~~~
Note: I'm using the pattern of one config server on each AWS instance and they belong to the
same replica set. Also, before beginning I forwarded the necessary ports on all the instances.
~~~~

The steps taken to setup and launch three config servers are as follows:

1. Created `midterm/db/config/data` folder to hold the data for the config server on each of the 3 instances in AWS. This will be used as the argument to the `--dbpath` flag while starting mongod in `configsvr` mode.
2. Used the following commands to start the config server on each of the 3 instances in AWS.

machine1:

~~~~
```
mongod --port 27022 --dbpath ~/midterm/db/config/data --configsvr --replSet config_midterm
--smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
```
```
ubuntu@ec2-54-202-76-188 ~> jobs
Job     Group   CPU     State   Command
1       9847    0%      running mongod --port 27022 --dbpath ~/midterm/db/config/data --configsvr
--replSet config_midterm --smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
ubuntu@ec2-54-202-76-188 ~>
```
~~~~

machine2:
```
mongod --port 27023 --dbpath ~/midterm/db/config/data --configsvr --replSet config_midterm
--smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
```
```
ubuntu@ec2-52-35-55-89 ~> jobs
Job     Group   CPU     State   Command
1       10640   0%      running mongod --port 27023 --dbpath ~/midterm/db/config/data --configsvr
--replSet config_midterm --smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
ubuntu@ec2-52-35-55-89 ~>
```

machine3
```
mongod --port 27024 --dbpath ~/midterm/db/config/data --configsvr --replSet config_midterm
--smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
```
```
ubuntu@ec2-52-41-163-132 ~> jobs
Job     Group   CPU     State   Command
1       10251   0%      running mongod --port 27024 --dbpath ~/midterm/db/config/data --configsvr
--replSet config_midterm --smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
ubuntu@ec2-52-41-163-132 ~>
```

3. Connected to the config server on machine1 through mongo from my host machine(macbook) through the terminal and initiated the replica-set named `config_midterm`. Then, added the config servers on machine2 and machine3 into the config server replica-set(CSRS).

```
sidmishraw@Sidharths-MBP ~> mongo --host ec2-54-245-184-166.us-west-2.compute.amazonaws.com
--port 27022
MongoDB shell version v3.4.2
connecting to: mongodb://ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022/
MongoDB server version: 3.4.2
Server has startup warnings:
2017-04-01T02:17:31.713+0000 I STORAGE  [initandlisten]
2017-04-01T02:17:31.713+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem
is strongly recommended with the WiredTiger storage engine
2017-04-01T02:17:31.713+0000 I STORAGE  [initandlisten] **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten]
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not
enabled for the database.
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten] **          Read and write access to
data and configuration is unrestricted.
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten]
```

```
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten]
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten]
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2017-04-01T02:17:31.773+0000 I CONTROL  [initandlisten]
>
> show dbs
2017-03-31T19:21:43.547-0700 E QUERY    [thread1] Error: listDatabases failed:{
        "ok" : 0,
        "errmsg" : "not master and slaveOk=false",
        "code" : 13435,
        "codeName" : "NotMasterNoSlaveOk"
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:62:1
shellHelper.show@src/mongo/shell/utils.js:755:19
shellHelper@src/mongo/shell/utils.js:645:15
@(shellhelp2):1:1
> rs.initiate()
{
        "info2" : "no configuration specified. Using a default configuration for the set",
        "me" : "ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022",
        "ok" : 1
}
config_midterm:SECONDARY> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
config_midterm:PRIMARY>
config_midterm:PRIMARY> rs.conf()
{
        "_id" : "config_midterm",
        "version" : 1,
        "configsvr" : true,
        "protocolVersion" : NumberLong(1),
        "members" : [
                {
                        "_id" : 0,
                        "host" : "ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {
                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                }
        ],
        "settings" : {
                "chainingAllowed" : true,
                "heartbeatIntervalMillis" : 2000,
                "heartbeatTimeoutSecs" : 10,
                "electionTimeoutMillis" : 10000,
                "catchUpTimeoutMillis" : 2000,
                "getLastErrorModes" : {

                },
                "getLastErrorDefaults" : {
                        "w" : 1,
                        "wtimeout" : 0
                },
                "replicaSetId" : ObjectId("58df0ebf1b37f6eef630124d")
        }
}
config_midterm:PRIMARY>
config_midterm:PRIMARY> rs.status()
{
        "set" : "config_midterm",
        "date" : ISODate("2017-04-01T02:29:07.751Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "configsvr" : true,
        "heartbeatIntervalMillis" : NumberLong(2000),
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1491013741, 1),
                        "t" : NumberLong(1)
                },
                "readConcernMajorityOpTime" : {
                        "ts" : Timestamp(1491013741, 1),
                        "t" : NumberLong(1)
                },
                "appliedOpTime" : {
                        "ts" : Timestamp(1491013741, 1),
                        "t" : NumberLong(1)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1491013741, 1),
                        "t" : NumberLong(1)
                }
        },
        "members" : [
                {
                        "_id" : 0,
                        "name" : "ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 696,
                        "optime" : {
                                "ts" : Timestamp(1491013741, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2017-04-01T02:29:01Z"),
                        "electionTime" : Timestamp(1491013311, 2),
                        "electionDate" : ISODate("2017-04-01T02:21:51Z"),
                        "configVersion" : 1,
                        "self" : true
                }
        ],
        "ok" : 1
}
config_midterm:PRIMARY>
```

Adding config server on machine2 to the `config_midterm` replica set:

```
config_midterm:PRIMARY> rs.add("ec2-54-148-246-51.us-west-2.compute.amazonaws.com:27023")
{ "ok" : 1 }
```

Adding config server on machine3 to the `config_midterm` replica set:

```
config_midterm:PRIMARY> rs.add("ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024")
{ "ok" : 1 }
```

The result of `rs.conf()` and `rs.status()` after addition of config servers on machine2 and machine3 to the `config_midterm` replica set:

```
config_midterm:PRIMARY> rs.conf()
{
        "_id" : "config_midterm",
        "version" : 3,
        "configsvr" : true,
        "protocolVersion" : NumberLong(1),
        "members" : [
                {
```

```
                "_id" : 0,
                "host" : "ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022",
                "arbiterOnly" : false,
                "buildIndexes" : true,
                "hidden" : false,
                "priority" : 1,
                "tags" : {

                },
                "slaveDelay" : NumberLong(0),
                "votes" : 1
        },
        {
                "_id" : 1,
                "host" : "ec2-54-148-246-51.us-west-2.compute.amazonaws.com:27023",
                "arbiterOnly" : false,
                "buildIndexes" : true,
                "hidden" : false,
                "priority" : 1,
                "tags" : {

                },
                "slaveDelay" : NumberLong(0),
                "votes" : 1
        },
        {
                "_id" : 2,
                "host" : "ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024",
                "arbiterOnly" : false,
                "buildIndexes" : true,
                "hidden" : false,
                "priority" : 1,
                "tags" : {

                },
                "slaveDelay" : NumberLong(0),
                "votes" : 1
        }
],
"settings" : {
        "chainingAllowed" : true,
        "heartbeatIntervalMillis" : 2000,
        "heartbeatTimeoutSecs" : 10,
        "electionTimeoutMillis" : 10000,
        "catchUpTimeoutMillis" : 2000,
        "getLastErrorModes" : {

        },
        "getLastErrorDefaults" : {
                "w" : 1,
                        "wtimeout" : 0
                },
                "replicaSetId" : ObjectId("58df0ebf1b37f6eef630124d")
        }
}
config_midterm:PRIMARY> rs.status()
{
        "set" : "config_midterm",
        "date" : ISODate("2017-04-01T02:33:35.670Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "configsvr" : true,
        "heartbeatIntervalMillis" : NumberLong(2000),
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1491014013, 1),
                        "t" : NumberLong(1)
                },
                "readConcernMajorityOpTime" : {
                        "ts" : Timestamp(1491014013, 1),
                        "t" : NumberLong(1)
                },
                "appliedOpTime" : {
                        "ts" : Timestamp(1491014013, 1),
                        "t" : NumberLong(1)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1491014013, 1),
                        "t" : NumberLong(1)
                }
        },
        "members" : [
                {
                        "_id" : 0,
                        "name" : "ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 964,
                        "optime" : {
                                "ts" : Timestamp(1491014013, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2017-04-01T02:33:33Z"),
                        "electionTime" : Timestamp(1491013311, 2),
                        "electionDate" : ISODate("2017-04-01T02:21:51Z"),
                        "configVersion" : 3,
                        "self" : true
                },
                {
                        "_id" : 1,
                        "name" : "ec2-54-148-246-51.us-west-2.compute.amazonaws.com:27023",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 193,
                        "optime" : {
                                "ts" : Timestamp(1491014013, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDurable" : {
                                "ts" : Timestamp(1491014013, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2017-04-01T02:33:33Z"),
                        "optimeDurableDate" : ISODate("2017-04-01T02:33:33Z"),
                        "lastHeartbeat" : ISODate("2017-04-01T02:33:34.773Z"),
                        "lastHeartbeatRecv" : ISODate("2017-04-01T02:33:33.775Z"),
                        "pingMs" : NumberLong(0),
                        "syncingTo" :
"ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022",
                        "configVersion" : 3
                },
                {
                        "_id" : 2,
                        "name" : "ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 118,
                        "optime" : {
                                "ts" : Timestamp(1491014013, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDurable" : {
                                "ts" : Timestamp(1491014013, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2017-04-01T02:33:33Z"),
                        "optimeDurableDate" : ISODate("2017-04-01T02:33:33Z"),
                        "lastHeartbeat" : ISODate("2017-04-01T02:33:34.773Z"),
                        "lastHeartbeatRecv" : ISODate("2017-04-01T02:33:34.092Z"),
                        "pingMs" : NumberLong(0),
                        "syncingTo" :
"ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022",
                        "configVersion" : 3
                }
        ],
        "ok" : 1
}
```

```
config_midterm:PRIMARY>
```

7. Connect mongos to each config server.

```
Note - `--chunkSize` is an invalid flag for `mongos` and according to the
docs[https://docs.mongodb.com/manual/tutorial/modify-chunk-size-in-sharded-cluster/], the
default chunksize is set to 64MB and it can be modified from the mongo shell. I'll modify the
chunk size after adding the shards to `mongos`(because mongo doesn’t let you modify the chunk
size without any shards added to the cluster).
```

Started the `mongos` (shard controller) on machine1 using the following command:

```
mongos --configdb config_midterm/ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
--port 27021 > ~/logs/midterm_shard_controller.log 2>&1 &
```

After starting `mongos` on machine1 on port `27021`, it automatically connects to other config servers on machine1 (port `27022`), machine2(port `27023`) and machine3(port `27024`) in the replica-set `config_midterm`.

```
ubuntu@ec2-54-202-76-188 ~> mongos --configdb
config_midterm/ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022 --port 27021 >
~/logs/midterm_shard_controller.log 2>&1 &
ubuntu@ec2-54-202-76-188 ~> jobs
Job     Group   CPU     State   Command
2       10721   1%      running mongos --configdb
config_midterm/ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022 --port 27021 >
~/logs/midterm_shard_controller.log 2>&1 &
1       9847    0%      running mongod --port 27022 --dbpath ~/midterm/db/config/data --configsvr
--replSet config_midterm --smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
ubuntu@ec2-54-202-76-188 ~> tail -f logs/midterm_shard_controller.log
2017-04-01T03:59:25.264+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to
ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.273+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.274+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.274+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.276+0000 W SHARDING [replSetDistLockPinger] pinging failed for distributed
lock pinger :: caused by :: LockStateChangeFailed: findAndModify query predicate didn't match
any lock document
2017-04-01T03:59:27.274+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to
ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024
2017-04-01T03:59:27.285+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024
```

```
2017-04-01T03:59:27.287+0000 I NETWORK  [thread2] waiting for connections on port 27021
2017-04-01T03:59:27.332+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to
ec2-54-148-246-51.us-west-2.compute.amazonaws.com:27023
2017-04-01T03:59:27.345+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-54-148-246-51.us-west-2.compute.amazonaws.com:27023
ubuntu@ec2-54-202-76-188 ~> cat logs/midterm_shard_controller.log
2017-04-01T03:59:25.227+0000 W SHARDING [main] Running a sharded cluster with fewer than 3
config servers should only be done for testing purposes and is not recommended for production.
2017-04-01T03:59:25.236+0000 I CONTROL  [main]
2017-04-01T03:59:25.236+0000 I CONTROL  [main] ** WARNING: Access control is not enabled for
the database.
2017-04-01T03:59:25.236+0000 I CONTROL  [main] **          Read and write access to data and
configuration is unrestricted.
2017-04-01T03:59:25.236+0000 I CONTROL  [main]
2017-04-01T03:59:25.236+0000 I SHARDING [mongosMain] mongos version v3.4.2
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] git version:
3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] OpenSSL version: OpenSSL 1.0.2g  1 Mar
2016
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] allocator: tcmalloc
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] modules: none
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] build environment:
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain]     distmod: ubuntu1404
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain]     distarch: x86_64
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain]     target_arch: x86_64
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] db version v3.4.2
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] git version:
3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] OpenSSL version: OpenSSL 1.0.2g  1 Mar
2016
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] allocator: tcmalloc
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] modules: none
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] build environment:
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain]     distmod: ubuntu1404
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain]     distarch: x86_64
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain]     target_arch: x86_64
2017-04-01T03:59:25.236+0000 I CONTROL  [mongosMain] options: { net: { port: 27021 },
sharding: { configDB:
"config_midterm/ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022" } }
2017-04-01T03:59:25.263+0000 I NETWORK  [mongosMain] Starting new replica set monitor for
config_midterm/ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.263+0000 I SHARDING [thread1] creating distributed lock ping thread for
process
ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27021:1491019165:-7500641746664576567
(sleeping for 30000ms)
2017-04-01T03:59:25.264+0000 I NETWORK  [mongosMain] changing hosts to
config_midterm/ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024,ec2-54-148-246-51.us-wes
t-2.compute.amazonaws.com:27023,ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022 from
config_midterm/ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
```

```
2017-04-01T03:59:25.264+0000 I SHARDING [mongosMain] Updating ShardRegistry connection string
for shard config from: config_midterm/ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
to:
config_midterm/ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024,ec2-54-148-246-51.us-wes
t-2.compute.amazonaws.com:27023,ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.264+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to
ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.264+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to
ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.264+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to
ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.273+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.274+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.274+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022
2017-04-01T03:59:25.276+0000 W SHARDING [replSetDistLockPinger] pinging failed for distributed
lock pinger :: caused by :: LockStateChangeFailed: findAndModify query predicate didn't match
any lock document
2017-04-01T03:59:27.274+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to
ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024
2017-04-01T03:59:27.285+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-52-39-43-165.us-west-2.compute.amazonaws.com:27024
2017-04-01T03:59:27.287+0000 I NETWORK  [thread2] waiting for connections on port 27021
2017-04-01T03:59:27.332+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to
ec2-54-148-246-51.us-west-2.compute.amazonaws.com:27023
2017-04-01T03:59:27.345+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully
connected to ec2-54-148-246-51.us-west-2.compute.amazonaws.com:27023
ubuntu@ec2-54-202-76-188 ~>
```

8. Setup and launch each of three shards: make sure to include the result of sh.status() before adding shards which will be done in the next task.

Followed the following steps in setting up and launching shard servers:

1. Created a directory for each shard on each of the 3 AWS instances [machine1, machine2, machine3]. Followed the naming convention `~/midterm/db/data/shardX` where X = {0, 1, 2}

machine1:

```
ubuntu@ec2-54-202-76-188 ~> mkdir midterm/db/data/shard0
ubuntu@ec2-54-202-76-188 ~>
```

machine2:

```
ubuntu@ec2-52-35-55-89 ~> mkdir midterm/db/data/shard1
ubuntu@ec2-52-35-55-89 ~>
```

machine3:

```
ubuntu@ec2-52-41-163-132 ~> mkdir midterm/db/data/shard2
ubuntu@ec2-52-41-163-132 ~>
```

2. Spun up `mongod` process on each of the AWS instances using the commands below. These were the shard servers, one shard server on each of machine1, machine2 and machine3:

machine1:

```
mongod --shardsvr --dbpath midterm/db/data/shard0 --port 28021 >
~/logs/midterm_shard_server1.log 2>&1 &
```

```
ubuntu@ec2-54-202-76-188 ~> jobs
Job     Group   CPU     State   Command
2       10721   0%      running mongos --configdb
config_midterm/ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27022 --port 27021 >
~/logs/midterm_shard_controller.log 2>&1 &
1       9847    0%      running mongod --port 27022 --dbpath ~/midterm/db/config/data --configsvr
--replSet config_midterm --smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
ubuntu@ec2-54-202-76-188 ~> mongod --dbpath midterm/db/data/shard0 --port 28021 >
~/logs/midterm_shard_server1.log 2>&1 &
ubuntu@ec2-54-202-76-188 ~> cat logs/midterm_shard_server1.log
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten] MongoDB starting : pid=11215
port=28021 dbpath=midterm/db/data/shard0 64-bit
host=ec2-54-245-184-166.us-west-2.compute.amazonaws.com
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten] git version:
3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2g  1 Mar
2016
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten] modules: none
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten] build environment:
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten]     distmod: ubuntu1404
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten]     distarch: x86_64
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten]     target_arch: x86_64
```

~~~
2017-04-01T04:50:25.796+0000 I CONTROL  [initandlisten] options: { net: { port: 28021 }, storage: { dbPath: "midterm/db/data/shard0" } }
2017-04-01T04:50:25.821+0000 I STORAGE  [initandlisten]
2017-04-01T04:50:25.821+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2017-04-01T04:50:25.821+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2017-04-01T04:50:25.821+0000 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=256M,session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten]
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten]
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten]
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten]
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2017-04-01T04:50:25.882+0000 I CONTROL  [initandlisten]
2017-04-01T04:50:25.894+0000 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'midterm/db/data/shard0/diagnostic.data'
2017-04-01T04:50:25.912+0000 I INDEX    [initandlisten] build index on: admin.system.version properties: { v: 2, key: { version: 1 }, name: "incompatible_with_version_32", ns: "admin.system.version" }
2017-04-01T04:50:25.912+0000 I INDEX    [initandlisten]    building index using bulk method; build may temporarily use up to 500 megabytes of RAM
2017-04-01T04:50:25.912+0000 I INDEX    [initandlisten] build index done.  scanned 0 total records. 0 secs
2017-04-01T04:50:25.913+0000 I COMMAND  [initandlisten] setting featureCompatibilityVersion to 3.4
2017-04-01T04:50:25.913+0000 I NETWORK  [thread1] waiting for connections on port 28021
ubuntu@ec2-54-202-76-188 ~>
```
~~~

machine2:

~~~
```
mongod --shardsvr --dbpath midterm/db/data/shard1 --port 28022 > ~/logs/midterm_shard_server2.log 2>&1 &
```
~~~

```
ubuntu@ec2-52-35-55-89 ~> jobs
Job     Group   CPU     State   Command
1       10640   0%      running mongod --port 27023 --dbpath ~/midterm/db/config/data --configsvr
--replSet config_midterm --smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
ubuntu@ec2-52-35-55-89 ~> mongod --dbpath midterm/db/data/shard1 --port 28022 >
~/logs/midterm_shard_server2.log 2>&1 &
ubuntu@ec2-52-35-55-89 ~> cat logs/midterm_shard_server2.log
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten] MongoDB starting : pid=11267
port=28022 dbpath=midterm/db/data/shard1 64-bit
host=ec2-54-148-246-51.us-west-2.compute.amazonaws.com
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten] git version:
3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2g  1 Mar
2016
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten] modules: none
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten] build environment:
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten]     distmod: ubuntu1404
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten]     distarch: x86_64
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten]     target_arch: x86_64
2017-04-01T04:52:21.288+0000 I CONTROL  [initandlisten] options: { net: { port: 28022 },
storage: { dbPath: "midterm/db/data/shard1" } }
2017-04-01T04:52:21.313+0000 I STORAGE  [initandlisten]
2017-04-01T04:52:21.313+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem
is strongly recommended with the WiredTiger storage engine
2017-04-01T04:52:21.313+0000 I STORAGE  [initandlisten] **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
2017-04-01T04:52:21.313+0000 I STORAGE  [initandlisten] wiredtiger_open config:
create,cache_size=256M,session_max=20000,eviction=(threads_max=4),config_base=false,statistics
=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idl
e_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten]
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not
enabled for the database.
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten] **          Read and write access to
data and configuration is unrestricted.
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten]
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten]
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten]
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2017-04-01T04:52:21.355+0000 I CONTROL  [initandlisten]
```

~~~
2017-04-01T04:52:21.365+0000 I FTDC     [initandlisten] Initializing full-time diagnostic data
capture with directory 'midterm/db/data/shard1/diagnostic.data'
2017-04-01T04:52:21.380+0000 I INDEX    [initandlisten] build index on: admin.system.version
properties: { v: 2, key: { version: 1 }, name: "incompatible_with_version_32", ns:
"admin.system.version" }
2017-04-01T04:52:21.380+0000 I INDEX    [initandlisten]    building index using bulk method;
build may temporarily use up to 500 megabytes of RAM
2017-04-01T04:52:21.380+0000 I INDEX    [initandlisten] build index done.  scanned 0 total
records. 0 secs
2017-04-01T04:52:21.380+0000 I COMMAND  [initandlisten] setting featureCompatibilityVersion to
3.4
2017-04-01T04:52:21.380+0000 I NETWORK  [thread1] waiting for connections on port 28022
ubuntu@ec2-52-35-55-89 ~>
```
~~~

machine3:

~~~
```
mongod --shardsvr --dbpath midterm/db/data/shard2 --port 28023 >
~/logs/midterm_shard_server3.log 2>&1 &
```
~~~

~~~
```
ubuntu@ec2-52-41-163-132 ~> jobs
Job     Group   CPU     State   Command
1       10251   0%      running mongod --port 27024 --dbpath ~/midterm/db/config/data --configsvr
--replSet config_midterm --smallfiles --oplogSize 128 > ~/logs/midterm_config.log 2>&1 &
ubuntu@ec2-52-41-163-132 ~> mongod --dbpath midterm/db/data/shard2 --port 28023 >
~/logs/midterm_shard_server3.log 2>&1 &
ubuntu@ec2-52-41-163-132 ~> cat logs/midterm_shard_server3.log
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten] MongoDB starting : pid=10959
port=28023 dbpath=midterm/db/data/shard2 64-bit
host=ec2-52-39-43-165.us-west-2.compute.amazonaws.com
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten] git version:
3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2g  1 Mar
2016
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten] modules: none
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten] build environment:
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten]     distmod: ubuntu1404
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten]     distarch: x86_64
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten]     target_arch: x86_64
2017-04-01T04:54:09.320+0000 I CONTROL  [initandlisten] options: { net: { port: 28023 },
storage: { dbPath: "midterm/db/data/shard2" } }
2017-04-01T04:54:09.345+0000 I STORAGE  [initandlisten]
2017-04-01T04:54:09.345+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem
is strongly recommended with the WiredTiger storage engine
~~~

```
2017-04-01T04:54:09.345+0000 I STORAGE  [initandlisten] **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
2017-04-01T04:54:09.345+0000 I STORAGE  [initandlisten] wiredtiger_open config:
create,cache_size=256M,session_max=20000,eviction=(threads_max=4),config_base=false,statistics
=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idl
e_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-04-01T04:54:09.391+0000 I CONTROL  [initandlisten]
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not
enabled for the database.
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten] **          Read and write access to
data and configuration is unrestricted.
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten]
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten]
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten]
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2017-04-01T04:54:09.392+0000 I CONTROL  [initandlisten]
2017-04-01T04:54:09.409+0000 I FTDC     [initandlisten] Initializing full-time diagnostic data
capture with directory 'midterm/db/data/shard2/diagnostic.data'
2017-04-01T04:54:09.426+0000 I INDEX    [initandlisten] build index on: admin.system.version
properties: { v: 2, key: { version: 1 }, name: "incompatible_with_version_32", ns:
"admin.system.version" }
2017-04-01T04:54:09.426+0000 I INDEX    [initandlisten]    building index using bulk method;
build may temporarily use up to 500 megabytes of RAM
2017-04-01T04:54:09.427+0000 I INDEX    [initandlisten] build index done.  scanned 0 total
records. 0 secs
2017-04-01T04:54:09.427+0000 I COMMAND  [initandlisten] setting featureCompatibilityVersion to
3.4
2017-04-01T04:54:09.427+0000 I NETWORK  [thread1] waiting for connections on port 28023
ubuntu@ec2-52-41-163-132 ~>
```

3. Added the ports `28021`, `28022` and `28023` to the security-group on all the instances and forwarded them. This would allow `mongos` and other components to communicate amongst themselves.
4. Connected to `mongos` running on machine1 using the mongo shell from my host machine (macbook) and ran the `sh.status()` command and `db.printShardingStatus()` command before telling about the sharding servers to the `mongos`.

```
sidmishraw@Sidharths-MBP ~> mongo --host ec2-54-245-184-166.us-west-2.compute.amazonaws.com
--port 27021
MongoDB shell version v3.4.2
```

```
connecting to: mongodb://ec2-54-245-184-166.us-west-2.compute.amazonaws.com:27021/
MongoDB server version: 3.4.2
Server has startup warnings:
2017-04-01T03:59:25.236+0000 I CONTROL  [main]
2017-04-01T03:59:25.236+0000 I CONTROL  [main] ** WARNING: Access control is not enabled for
the database.
2017-04-01T03:59:25.236+0000 I CONTROL  [main] **          Read and write access to data and
configuration is unrestricted.
2017-04-01T03:59:25.236+0000 I CONTROL  [main]
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("58df0ec01b37f6eef630125c")
}
  shards:
  active mongoses:
        "3.4.2" : 1
 autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
                Balancer lock taken at Fri Mar 31 2017 19:21:53 GMT-0700 (PDT) by
ConfigServer:Balancer
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:

mongos> db.printShardingStatus()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("58df0ec01b37f6eef630125c")
}
  shards:
  active mongoses:
        "3.4.2" : 1
 autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
                Balancer lock taken at Fri Mar 31 2017 19:21:53 GMT-0700 (PDT) by
ConfigServer:Balancer
```

```
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:

mongos>
```

9. Add shards: make sure to include the result of sh.status() after adding shards

Connected to `mongos` running on machine1 on port `27021` and added the 3 shard servers and then did `sh.status()` to get the status of the shards.

```
mongos> sh.addShard("ec2-54-245-184-166.us-west-2.compute.amazonaws.com:28021")
{ "shardAdded" : "shard0000", "ok" : 1 }
mongos> sh.addShard("ec2-54-148-246-51.us-west-2.compute.amazonaws.com:28022")
{ "shardAdded" : "shard0001", "ok" : 1 }
mongos> sh.addShard("ec2-52-39-43-165.us-west-2.compute.amazonaws.com:28023")
{ "shardAdded" : "shard0002", "ok" : 1 }
mongos>
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("58df0ec01b37f6eef630125c")
}
  shards:
        {  "_id" : "shard0000",  "host" :
"ec2-54-245-184-166.us-west-2.compute.amazonaws.com:28021",  "state" : 1 }
        {  "_id" : "shard0001",  "host" :
"ec2-54-148-246-51.us-west-2.compute.amazonaws.com:28022",  "state" : 1 }
        {  "_id" : "shard0002",  "host" :
"ec2-52-39-43-165.us-west-2.compute.amazonaws.com:28023",  "state" : 1 }
  active mongoses:
        "3.4.2" : 1
 autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
                Balancer lock taken at Fri Mar 31 2017 19:21:53 GMT-0700 (PDT) by
ConfigServer:Balancer
        Failed balancer rounds in last 5 attempts:  5
        Last reported error:  Cannot accept sharding commands if not started with --shardsvr
        Time of Reported error:  Fri Mar 31 2017 22:31:44 GMT-0700 (PDT)
        Migration Results for the last 24 hours:
                No recent migrations
```

~~~~
  databases:

mongos>
```
~~~~

10. Enable shards: explain the nature of the shard key (ascending, random, or location based) and the sharding strategy (range-based or hash based)

1. Before enabling shards on the database, set the chunk size to 1MB

~~~~
```
mongos> use config
switched to db config
mongos> db
config
mongos> db.getCollectionNames()
[
        "actionlog",
        "changelog",
        "chunks",
        "lockpings",
        "locks",
        "migrations",
        "mongos",
        "shards",
        "tags",
        "version"
]
mongos> db.settings.save({"_id": "chunksize", "value": 1})
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : "chunksize" })
mongos>
```
~~~~

2. Explicitly told `mongos` that the database `archivesdb` needs to be sharded.
3. Then, enabled sharding on `month4` collection of `archivesdb` and used `hashed` sharding for it on the `_id` field of the collection. The collection was going to hold the archives from New York Times archives API for the month of April, 2000. The `_id` was going to be an `uuid` so it didn’t make sense ordering them, hence, went with hashing strategy.

~~~~
```
mongos> sh.enableSharding("archivesdb")
{ "ok" : 1 }
mongos> sh.shardCollection("archivesdb.month4", {"_id": "hashed"})
{ "collectionsharded" : "archivesdb.month4", "ok" : 1 }
mongos>
```
~~~~

11. Populate data in the cluster: Explain your collection and include the code to populate data and sh.status() after populating data

Used the following python script to populate data from NYT archives API for the month of April, 2000:

~~~
```python
# midterm_data_populator.py
# -*- coding: utf-8 -*-
# @Author: Sidharth Mishra
# @Date:   2017-03-31 22:48:18
# @Last Modified by:   Sidharth Mishra
# @Last Modified time: 2017-03-31 23:43:28



'''
This script is used to populate data into the sharding cluster made for midterm take home
exam.
The data populated will be for 1 month from NYT archives API.
'''



# Python Standard Library imports
from urllib.request import urlopen
from json import loads
from json import dumps
from datetime import datetime



# pymongo imports
from pymongo import MongoClient



# Constants
__HOSTNAME__ = 'ec2-54-245-184-166.us-west-2.compute.amazonaws.com'
__PORT__ = '27021'
__NYT_API_BASE_PATH__ = 'http://api.nytimes.com/svc/'
__API_KEY__ = 'c9e169f173f34249a02bc0aff8d50fb9'
__ARCHIVES_DOC_KEY__ = 'docs'
__RESPONSE__ = 'response'
__DB__ = 'archivesdb'
__COLLECTION__ = 'month4'
~~~

```
# Article fields
__WEB_URL__ = 'web_url'
__SNIPPET__ = 'snippet'
__LEAD_PARAGRAPH__ = 'lead_paragraph'
__ABSTRACT__ = 'abstract'
__PRINT_PAGE__ = 'print_page'
__BLOG__ = 'blog'
__SOURCE__ = 'source'
__MULTIMEDIA__ = 'multimedia'
__HEADLINE__ = 'headline'
__HEADLINE_MAIN__ = 'main'
__KEYWORDS__ = 'keywords'
__KEYWORDS_NAME__ = 'name'
__KEYWORDS_VALUE__ = 'value'
__PUB_DATE__ = 'pub_date'
__DOCUMENT_TYPE__ = 'document_type'
__NEWS_DESK__ = 'news_desk'
__SECTION_NAME__ = 'section_name'
__SUBSECTION_NAME__ = 'subsection_name'
__BYLINE__ = 'byline'
__PERSON__ = 'person'
__FIRSTNAME__ = 'firstname'
__MIDDLENAME__ = 'middlename'
__LASTNAME__ = 'lastname'
__RANK__ = 'rank'
__ROLE__ = 'role'
__ORGANIZATION__ = 'organization'
__ORIGINAL__ = 'original'
__TYPE_OF_MATERIAL__ = 'type_of_material'
__ID__ = '_id'
__WORD_COUNT__ = 'word_count'
__SLIDESHOW_CREDITS__ = 'slideshow_credits'



# Connection to MongoDB
def get_client():
  '''
  Connects to the MongoDB instance and returns the MongoDBclient.

  :return: client `MongoClient`
  '''

  client = MongoClient('mongodb://{hostname}:{port}'.format(hostname = __HOSTNAME__, \
    port = __PORT__))
  return client



# Creation of archives dataset
# NYT Archives API for fetching data from NYT
def __archives_api__(year = 2000, month = 4):
  '''
  Calls the NYT archives API and generates the sample dataset inside `nyt_archives.json`.
  Takes year and month as the arguments.

  :param: year `int` - Default value = 2000 - The year of the archives
  :param: month `int` - Default value = 4 - The month of the archives

  :return: archives `list`
  '''

  url_string = '{base_url}archive/v1/{year}/{month}.json?api-key={api_key}'.format(\
    base_url = __NYT_API_BASE_PATH__, year = year, month = month, api_key = __API_KEY__)

  json_string = None

  with urlopen(url_string) as archive_res:
    json_string = archive_res.read()

  return loads(json_string)[__RESPONSE__][__ARCHIVES_DOC_KEY__]



# dumping function
def dump_data_scluster():
  '''
  Calls the NYT archives API and dumps all the archives into the sharded cluster.

  :return: `None`
  '''

  client = get_client()
  archives = __archives_api__()
  db = client.get_database(__DB__)
  db[__COLLECTION__].insert_many(archives)
  print('Created and inserted archives into {} collection..'.format(__COLLECTION__))
  return
```

```
if __name__ == '__main__':
  print('Dumping data into Sharding cluster running on {hostname}:{port}'.format(\
    hostname = __HOSTNAME__, port = __PORT__))
  dump_data_scluster()
  print('Done dumping data. Please check the cluster...')
```

The document of the `month4` collection is an archive metadata from `NYT` and it looks like:

```
mongos> db.month4.find().limit(1).pretty()
{
        "_id" : "4fd1f60b8eb7c8105d7504c0",
        "web_url" :
"http://www.nytimes.com/2000/04/01/business/a-role-model-s-new-clothes.html",
        "snippet" : "Ever mindful of the need to stay au courant, Mattel staged a makeover this
year for Barbie, the 41-year-old fashion doll with $1.3 billion in annual sales. She emerged
with a navel and a smile that shows some teeth, giving her a slightly more natural...",
        "lead_paragraph" : "Ever mindful of the need to stay au courant, Mattel staged a
makeover this year for Barbie, the 41-year-old fashion doll with $1.3 billion in annual sales.
She emerged with a navel and a smile that shows some teeth, giving her a slightly more natural
look. These were not, however, the only concessions to reality that Barbie's designers have
felt compelled to make recently. In the last couple of years, Mattel Inc. has been under
increasing pressure from some parents to lay aside Barbie's trademark vagueness and make her
more career-oriented to build credibility -- not to mention sales -- among the primary-school
set and their two-career parents.",
        "abstract" : "Mattel Inc is trying to update Barbie's image, partly because of
competition from more career-minded role models; in last couple of years, Mattel has been
under increasing pressure from some parents to lay aside Barbie's trademark vagueness and make
her more career-oriented to build credibility among primary-school set and their two-career
parents; Mattel is packaging dolls with literature or CD-ROM's that emphasize education and
other requirements for employability; shoppers can expect to see Jessica the Journalist, Get
Real Girl and Barbie for President; graphs; photos (M)",
        "print_page" : "1",
        "blog" : [ ],
        "source" : "The New York Times",
        "multimedia" : [ ],
        "headline" : {
                "main" : "A Role Model's New Clothes"
        },
        "keywords" : [
                {
                        "name" : "organizations",
                        "value" : "MATTEL INC"
                },
                {
                        "name" : "subject",
                        "value" : "DOLLS"
                },
                {
                        "name" : "subject",
                        "value" : "TOYS"
                },
                {
                        "name" : "subject",
                        "value" : "BARBIE (DOLL)"
                }
        ],
        "pub_date" : "2000-04-01T00:00:00Z",
        "document_type" : "article",
        "news_desk" : "Business/Financial Desk",
        "section_name" : "Business",
        "subsection_name" : null,
        "byline" : {
                "person" : [
                        {
                                "firstname" : "Constance",
                                "middlename" : "L.",
                                "lastname" : "HAYS",
                                "rank" : 1,
                                "role" : "reported",
                                "organization" : ""
                        }
                ],
                "original" : "By CONSTANCE L. HAYS"
        },
        "type_of_material" : "News",
        "word_count" : 1820,
        "slideshow_credits" : null
}
```

The result of `sh.status()` after populating the data using the script:

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("58df0ec01b37f6eef630125c")
}
  shards:
        {  "_id" : "shard0000",  "host" :
"ec2-54-245-184-166.us-west-2.compute.amazonaws.com:28021",  "state" : 1 }
        {  "_id" : "shard0001",  "host" :
"ec2-54-148-246-51.us-west-2.compute.amazonaws.com:28022",  "state" : 1 }
        {  "_id" : "shard0002",  "host" :
"ec2-52-39-43-165.us-west-2.compute.amazonaws.com:28023",  "state" : 1 }
```

```
  active mongoses:
        "3.4.2" : 1
 autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
                Balancer lock taken at Fri Mar 31 2017 19:21:53 GMT-0700 (PDT) by
ConfigServer:Balancer
        Failed balancer rounds in last 5 attempts:  5
        Last reported error:  Cannot accept sharding commands if not started with --shardsvr
        Time of Reported error:  Fri Mar 31 2017 23:12:37 GMT-0700 (PDT)
        Migration Results for the last 24 hours:
                2 : Success
  databases:
        {  "_id" : "archivesdb",  "primary" : "shard0000",  "partitioned" : true }
                archivesdb.month4
                        shard key: { "_id" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard0000      6
                                shard0001      6
                                shard0002      6
                        { "_id" : { "$minKey" : 1 } } -->> { "_id" :
NumberLong("-7994834805591274086") } on : shard0000 Timestamp(3, 8)
                        { "_id" : NumberLong("-7994834805591274086") } -->> { "_id" :
NumberLong("-6708533428051894629") } on : shard0000 Timestamp(3, 9)
                        { "_id" : NumberLong("-6708533428051894629") } -->> { "_id" :
NumberLong("-6148914691236517204") } on : shard0000 Timestamp(3, 10)
                        { "_id" : NumberLong("-6148914691236517204") } -->> { "_id" :
NumberLong("-4944659153516507066") } on : shard0000 Timestamp(3, 11)
                        { "_id" : NumberLong("-4944659153516507066") } -->> { "_id" :
NumberLong("-3705147641915660260") } on : shard0000 Timestamp(3, 12)
                        { "_id" : NumberLong("-3705147641915660260") } -->> { "_id" :
NumberLong("-3074457345618258602") } on : shard0000 Timestamp(3, 13)
                        { "_id" : NumberLong("-3074457345618258602") } -->> { "_id" :
NumberLong("-1791707141705514461") } on : shard0001 Timestamp(3, 14)
                        { "_id" : NumberLong("-1791707141705514461") } -->> { "_id" :
NumberLong("-604804985821992753") } on : shard0001 Timestamp(3, 15)
                        { "_id" : NumberLong("-604804985821992753") } -->> { "_id" :
NumberLong(0) } on : shard0001 Timestamp(3, 16)
                        { "_id" : NumberLong(0) } -->> { "_id" :
NumberLong("1271222889174364606") } on : shard0001 Timestamp(3, 17)
                        { "_id" : NumberLong("1271222889174364606") } -->> { "_id" :
NumberLong("2581022735825228243") } on : shard0001 Timestamp(3, 18)
                        { "_id" : NumberLong("2581022735825228243") } -->> { "_id" :
NumberLong("3074457345618258602") } on : shard0001 Timestamp(3, 19)
                        { "_id" : NumberLong("3074457345618258602") } -->> { "_id" :
NumberLong("4525575298507555253") } on : shard0002 Timestamp(3, 20)
```

```
                      { "_id" : NumberLong("4525575298507555253") } -->> { "_id" :
NumberLong("5761142528651830521") } on : shard0002 Timestamp(3, 21)
                      { "_id" : NumberLong("5761142528651830521") } -->> { "_id" :
NumberLong("6148914691236517204") } on : shard0002 Timestamp(3, 22)
                      { "_id" : NumberLong("6148914691236517204") } -->> { "_id" :
NumberLong("7371690909503060797") } on : shard0002 Timestamp(3, 23)
                      { "_id" : NumberLong("7371690909503060797") } -->> { "_id" :
NumberLong("8660275627863292262") } on : shard0002 Timestamp(3, 24)
                      { "_id" : NumberLong("8660275627863292262") } -->> { "_id" : { "$maxKey"
: 1 } } on : shard0002 Timestamp(3, 25)

mongos>
```

12. Generate one query and show which shard served the query

Used the following query and used its explanation to find the shard that served the query. Since I did a `regex` search, mongo had to do a `SHARD_MERGE` as it had to look into all 3 shards. The details can be found in the `winningPlan` field of the `explain` JSON.

```
mongos> db.month4.find({"_id": {$regex: /.*4fd1f60b8eb7c8105d7.*/}}).limit(1)
{ "_id" : "4fd1f60b8eb7c8105d7504b0", "web_url" :
"http://www.nytimes.com/2000/04/25/opinion/foreign-affairs-reno-for-president.html", "snippet"
: "Frankly, I liked both pictures.    Yup, I gotta confess, that now-famous picture of a U.S.
marshal in Miami pointing an automatic weapon toward Donato Dalrymple and ordering him in the
name of the U.S. government to turn over Elian Gonzalez warmed my...", "lead_paragraph" :
"Frankly, I liked both pictures. Yup, I gotta confess, that now-famous picture of a U.S.
marshal in Miami pointing an automatic weapon toward Donato Dalrymple and ordering him in the
name of the U.S. government to turn over Elian Gonzalez warmed my heart. They should put that
picture up in every visa line in every U.S. consulate around the world, with a caption that
reads: ''America is a country where the rule of law rules. This picture illustrates what
happens to those who defy the rule of law and how far our government and people will go to
preserve it. Come all ye who understand that.''", "abstract" : "Thomas L Friedman Op-Ed column
praises Atty Gen Janet Reno for using armed Immigration and Naturalization agents to seize
Elian Gonzalez from his relatives home in Miami; also urges Sec of State Madeleine Albright to
relax embargo on Cuba (M)", "print_page" : "23", "blog" : [ ], "source" : "The New York
Times", "multimedia" : [ ], "headline" : { "main" : "Foreign Affairs; Reno for President" },
"keywords" : [ { "name" : "persons", "value" : "ALBRIGHT, MADELEINE K" }, { "name" :
"persons", "value" : "GONZALEZ, ELIAN" }, { "name" : "persons", "value" : "RENO, JANET" }, {
"name" : "persons", "value" : "GONZALEZ, JUAN MIGUEL" }, { "name" : "glocations", "value" :
"MIAMI (FLA)" }, { "name" : "glocations", "value" : "CUBA" }, { "name" : "organizations",
"value" : "IMMIGRATION AND NATURALIZATION SERVICE" }, { "name" : "subject", "value" : "UNITED
STATES INTERNATIONAL RELATIONS" }, { "name" : "subject", "value" : "CUBA-INTERNATIONAL
RELATIONS-US" }, { "name" : "subject", "value" : "CHILD CUSTODY AND SUPPORT" }, { "name" :
"subject", "value" : "IMMIGRATION AND REFUGEES" }, { "name" : "subject", "value" : "EMBARGOES
AND ECONOMIC SANCTIONS" }, { "name" : "subject", "value" : "CUBAN-AMERICANS" } ], "pub_date" :
"2000-04-25T00:00:00Z", "document_type" : "article", "news_desk" : "Editorial Desk",
"section_name" : "Opinion", "subsection_name" : null, "byline" : { "person" : [ { "firstname"
: "Thomas", "middlename" : "L.", "lastname" : "FRIEDMAN", "rank" : 1, "role" : "reported",
"organization" : "" } ], "original" : "By THOMAS L. FRIEDMAN" }, "type_of_material" : "Op-Ed",
"word_count" : 817, "slideshow_credits" : null }
mongos> db.month4.find({"_id": {$regex: /.*4fd1f60b8eb7c8105d7.*/}}).limit(1).explain()
{
        "queryPlanner" : {
                "mongosPlannerVersion" : 1,
                "winningPlan" : {
                        "stage" : "SHARD_MERGE",
                        "shards" : [
                                {
                                        "shardName" : "shard0000",
                                        "connectionString" :
"ec2-54-245-184-166.us-west-2.compute.amazonaws.com:28021",
                                        "serverInfo" : {
                                                "host" :
"ec2-54-245-184-166.us-west-2.compute.amazonaws.com",
                                                "port" : 28021,
                                                "version" : "3.4.2",
                                                "gitVersion" :
"3f76e40c105fc223b3e5aac3e20dcd026b83b38b"
                                        },
                                        "plannerVersion" : 1,
                                        "namespace" : "archivesdb.month4",
                                        "indexFilterSet" : false,
                                        "parsedQuery" : {
                                                "_id" : {
                                                        "$regex" : ".*4fd1f60b8eb7c8105d7.*"
                                                }
                                        },
                                        "winningPlan" : {
                                                "stage" : "LIMIT",
                                                "limitAmount" : 1,
                                                "inputStage" : {
                                                        "stage" : "FETCH",
                                                        "inputStage" : {
                                                                "stage" : "SHARDING_FILTER",
                                                                "inputStage" : {
                                                                        "stage" : "IXSCAN",
                                                                                "filter" : {
                                                                                "_id" : {
                                                                                        "$regex" :
".*4fd1f60b8eb7c8105d7.*"

                                                                                }
                                                                        },
                                                                        "keyPattern" : {
                                                                                "_id" : 1
                                                                        },
                                                                        "indexName" : "_id_",
                                                                        "isMultiKey" : false,
                                                                        "multiKeyPaths" : {
```

```
                                                                        "_id" : [ ]
                                                                },
                                                                "isUnique" : true,
                                                                "isSparse" : false,
                                                                "isPartial" : false,
                                                                "indexVersion" : 2,
                                                                "direction" : "forward",
                                                                "indexBounds" : {
                                                                        "_id" : [
                                                                                "[\"\", {})",

"[/.*4fd1f60b8eb7c8105d7.*/, /.*4fd1f60b8eb7c8105d7.*/]"
                                                                        ]
                                                                }
                                                        }
                                                }
                                        }
                                },
                                "rejectedPlans" : [ ]
                        },
                        {
                                "shardName" : "shard0001",
                                "connectionString" :
"ec2-54-148-246-51.us-west-2.compute.amazonaws.com:28022",
                                "serverInfo" : {
                                        "host" :
"ec2-54-148-246-51.us-west-2.compute.amazonaws.com",
                                        "port" : 28022,
                                        "version" : "3.4.2",
                                        "gitVersion" :
"3f76e40c105fc223b3e5aac3e20dcd026b83b38b"
                                },
                                "plannerVersion" : 1,
                                "namespace" : "archivesdb.month4",
                                "indexFilterSet" : false,
                                "parsedQuery" : {
                                        "_id" : {
                                                "$regex" : ".*4fd1f60b8eb7c8105d7.*"
                                        }
                                },
                                "winningPlan" : {
                                        "stage" : "LIMIT",
                                        "limitAmount" : 1,
                                        "inputStage" : {
                                                "stage" : "FETCH",
                                                "inputStage" : {
                                                        "stage" : "SHARDING_FILTER",
                                                        "inputStage" : {
                                                                "stage" : "IXSCAN",
                                                                "filter" : {
```

```
                                                                                "_id" : {
                                                                                        "$regex" :
".*4fd1f60b8eb7c8105d7.*"
                                                                                }
                                                                        },
                                                                        "keyPattern" : {
                                                                                "_id" : 1
                                                                        },
                                                                        "indexName" : "_id_",
                                                                        "isMultiKey" : false,
                                                                        "multiKeyPaths" : {
                                                                                "_id" : [ ]
                                                                        },
                                                                        "isUnique" : true,
                                                                        "isSparse" : false,
                                                                        "isPartial" : false,
                                                                        "indexVersion" : 2,
                                                                        "direction" : "forward",
                                                                        "indexBounds" : {
                                                                                "_id" : [
                                                                                        "[\"\", {})",

"[/.*4fd1f60b8eb7c8105d7.*/, /.*4fd1f60b8eb7c8105d7.*/]"
                                                                                ]
                                                                        }
                                                                }
                                                        }
                                                }
                                        },
                                        "rejectedPlans" : [ ]
                                },
                                {
                                        "shardName" : "shard0002",
                                        "connectionString" :
"ec2-52-39-43-165.us-west-2.compute.amazonaws.com:28023",
                                        "serverInfo" : {
                                                "host" :
"ec2-52-39-43-165.us-west-2.compute.amazonaws.com",
                                                "port" : 28023,
                                                "version" : "3.4.2",
                                                "gitVersion" :
"3f76e40c105fc223b3e5aac3e20dcd026b83b38b"
                                        },
                                        "plannerVersion" : 1,
                                        "namespace" : "archivesdb.month4",
                                        "indexFilterSet" : false,
                                        "parsedQuery" : {
                                                "_id" : {
                                                        "$regex" : ".*4fd1f60b8eb7c8105d7.*"
                                                }
```

```
					},
					"winningPlan" : {
						"stage" : "LIMIT",
						"limitAmount" : 1,
						"inputStage" : {
							"stage" : "FETCH",
							"inputStage" : {
								"stage" : "SHARDING_FILTER",
								"inputStage" : {
									"stage" : "IXSCAN",
									"filter" : {
										"_id" : {
											"$regex" :
".*4fd1f60b8eb7c8105d7.*"
										}
									},
									"keyPattern" : {
										"_id" : 1
									},
									"indexName" : "_id_",
									"isMultiKey" : false,
									"multiKeyPaths" : {
										"_id" : [ ]
									},
									"isUnique" : true,
									"isSparse" : false,
									"isPartial" : false,
									"indexVersion" : 2,
									"direction" : "forward",
									"indexBounds" : {
										"_id" : [
											"[\"\", {})",

"[/.*4fd1f60b8eb7c8105d7.*/, /.*4fd1f60b8eb7c8105d7.*/]"
										]
									}
								}
							}
						}
					},
					"rejectedPlans" : [ ]
				}
			]
		}
	},
	"ok" : 1
}
mongos>
```