San José State University | Date: November 5, 2022

# Midterm

CS 157C - NoSQL Database Systems | Submitted by: Deep Shah (I.D.: 016662932)

**Index**


1. Deployment Plan 2
2. Accessing Instances through SSH 3
3. MongoDB Installation 3
4. Data Directories 4
5. Private & Public IPs of Nodes 4
6. Config Server Set-up 5
7. Connect to Mongos 7
8. Shard Set-up 7
9. Adding Shards 10
10. Shard Key & Sharding Strategy 11
11. Adding Data from Public Dataset 11
12. Running Queries on Data 13
    1. Find documents in a given range 13
    2. Query with $elemMatch with two conditions 15
    3. Query with $in, $nin, or $all 17
    4. Query involving aggregate() 19
    5. Query involving mapReduce() 19
    6. Update query 20
13. Replication of Shards 20
14. List of Hosts 24


Link to Doc/PDF

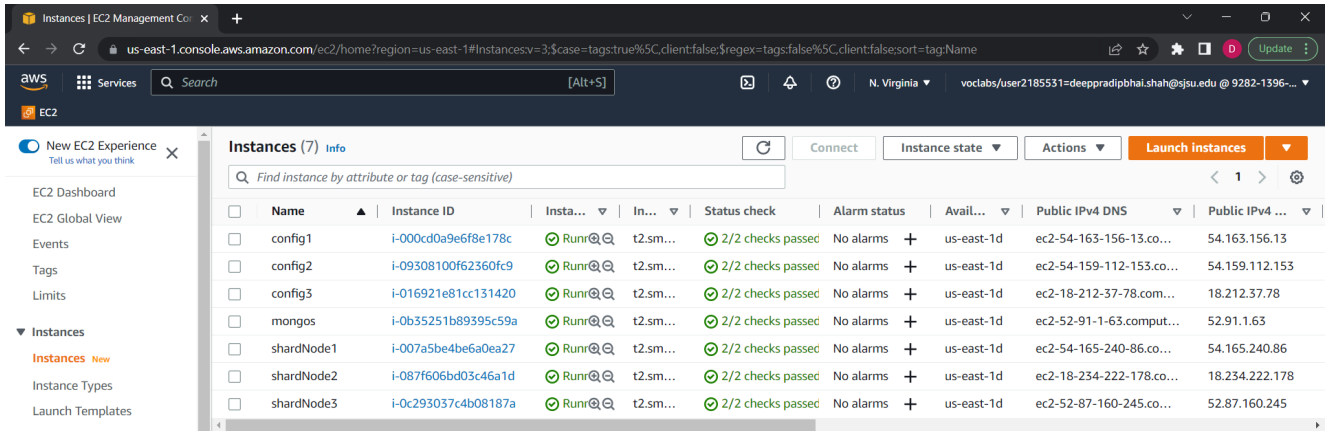
## 1. Deployment Plan

Number of Nodes: 7

**Deployment Summary**

| Service | Instance/ Node Name | IP Address (Private) | Port | Replica Set Name | Daemon | DB Path | Replica Priority |
|---|---|---|---|---|---|---|---|
| Shard Controller | mongos | 172.31.80.79 | 27020 | - | mongos | - | - |
| Config Server Primary | config1 | 172.31.90.97 | 27019 | crs | mongod | /data/db | 1 |
| Config Server Secondary 1 | config2 | 172.31.89.25 | 27019 | crs | mongod | /data/db | 1 |
| Config Server Secondary 2 | config3 | 172.31.86.144 | 27019 | crs | mongod | /data/db | 1 |
| Shard a Primary | shardNode1 | 172.31.93.248 | 27021 | a | mongod | /data/a | 4 |
| Shard b Secondary 1 | | 172.31.93.248 | 27022 | b | mongod | /data/b | 1 |
| Shard c Secondary 1 | | 172.31.93.248 | 27023 | c | mongod | /data/c | 1 |
| Shard a Secondary 1 | shardNode2 | 172.31.84.139 | 27021 | a | mongod | /data/a | 1 |
| Shard b Primary | | 172.31.84.139 | 27022 | b | mongod | /data/b | 4 |
| Shard c Secondary 2 | | 172.31.84.139 | 27023 | c | mongod | /data/c | 1 |
| Shard a Secondary 2 | shardNode3 | 172.31.90.164 | 27021 | a | mongod | /data/a | 1 |
| Shard b Secondary 2 | | 172.31.90.164 | 27022 | b | mongod | /data/b | 1 |
| Shard c Primary | | 172.31.90.164 | 27023 | c | mongod | /data/c | 4 |

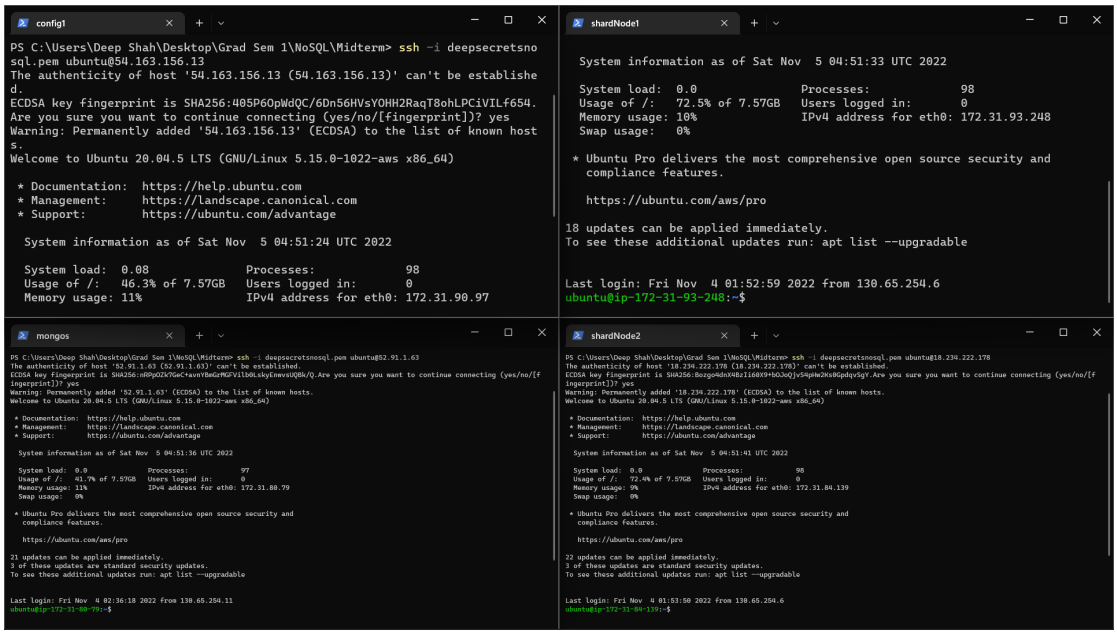
Nodes Set-up in AWS:



## 2. Accessing Instances through SSH

Command:
ssh -i secret_key.pem ubuntu@{public_ip}

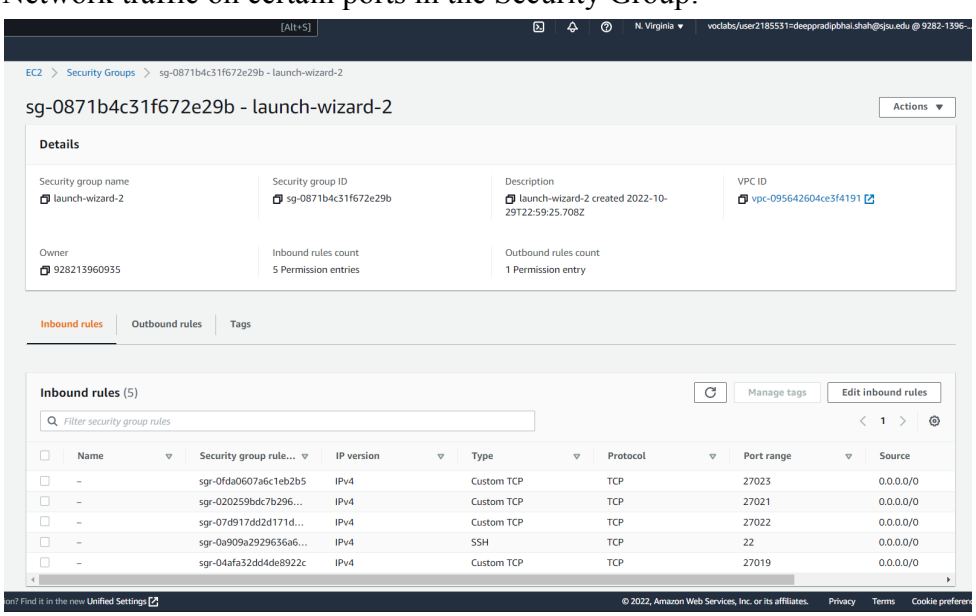
Accessing some instances through SSH:



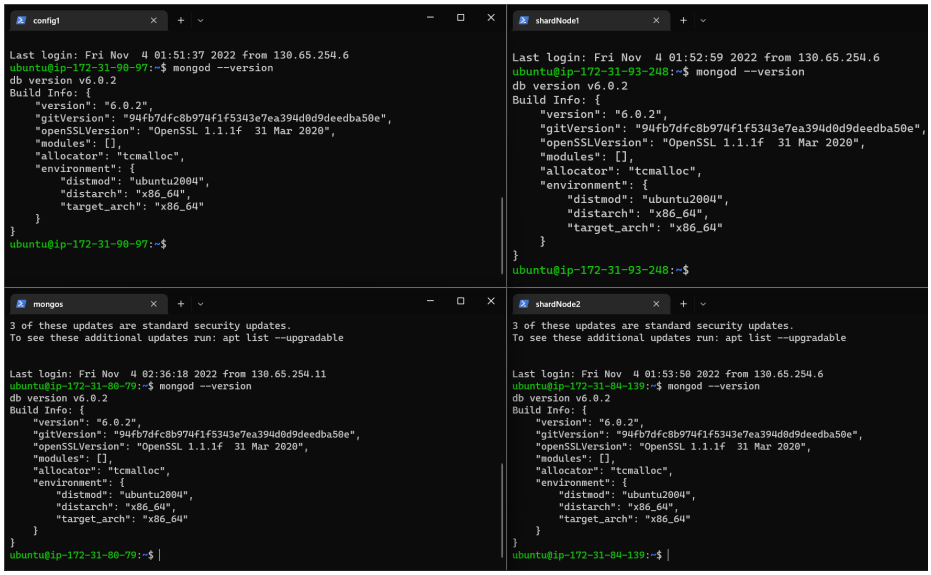

## 3. MongoDB Installation

Step Followed: Install MongoDB on Ubuntu
Also, enable the inbound network in the security group of each node.

Allowing Inbound Network traffic on certain ports in the Security Group:



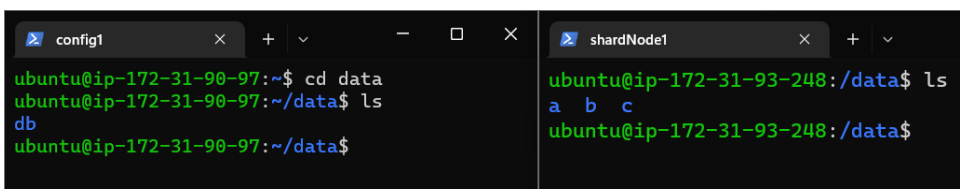
Checking the mongod version to verify MongoDB Installation:




## 4. Data Directories

Command:
sudo mkdir -p /data/db
sudo mkdir -p /data/a /data/b /data/c

Checking the data directories created:




## 5. Private & Public IPs of Nodes

| Instance Name | Private IP Address | Public IP Address |
|---|---|---|
| mongos | 172.31.80.79 | 52.91.1.63 |
| config1 | 172.31.90.97 | 54.163.156.13 |
| config2 | 172.31.89.25 | 54.159.112.153 |
| config3 | 172.31.86.144 | 18.212.37.78 |
| shardNode1 | 172.31.93.248 | 54.165.240.86 |
| shardNode2 | 172.31.84.139 | 18.234.222.178 |
| shardNode3 | 172.31.90.164 | 52.87.160.245 |

## 6. Config Server Set-up

Commands:

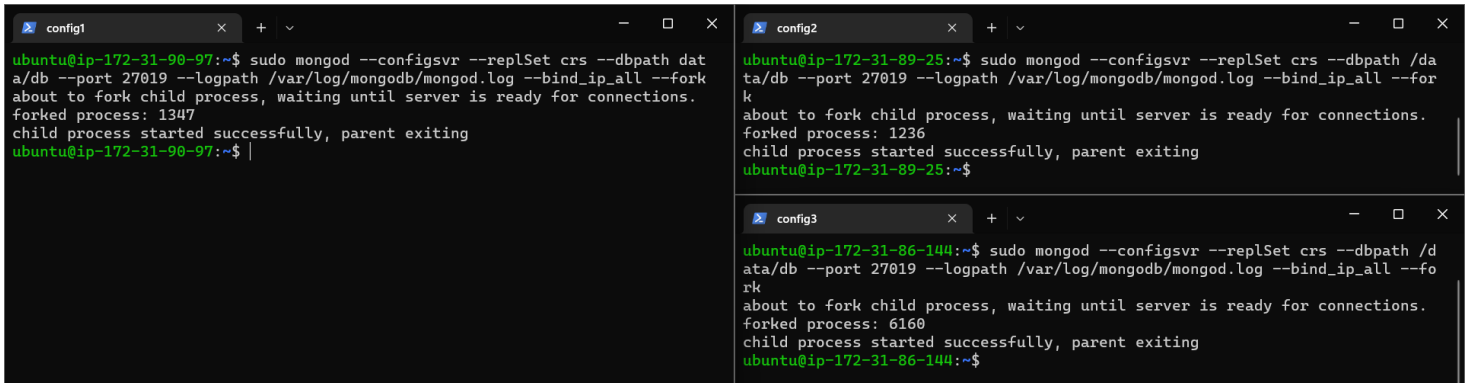
Setup Config servers (to be run in each node):
sudo mongod --configsvr --replSet crs --dbpath /data/db --port 27019 --logpath /var/log/mongodb/mongod.log --bind_ip_all --fork

Initiate Replica Set:
rs.initiate({_id: "crs", configsvr: true, members : [ {_id: 0, host: "172.31.90.97:27019"}, {_id: 1, host: "172.31.89.25:27019"}, {_id: 2, host: "172.31.86.144:27019"} ] } )

Steps:

1. In each of the instances config1, config2, config3, run the above command to Setup the Config Servers.
2. Use command: mongosh -port 27019  in any one instance.
3. Initiate the replica set using the rs.initiate command above.

Running the Config Servers:




Config Server Replica Set:

config1

```
crs [direct: secondary] test> rs.status()
{
  set: 'crs',
  date: ISODate("2022-11-05T05:51:51.063Z"),
  myState: 2,
  term: Long("13"),
  syncSourceHost: '172.31.89.25:27019',
  syncSourceId: 1,
  configsvr: true,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1667627510, i: 1 }), t: Long("13") },
    lastCommittedWallTime: ISODate("2022-11-05T05:51:50.055Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1667627510, i: 1 }), t: Long("13") }
,
    appliedOpTime: { ts: Timestamp({ t: 1667627511, i: 1 }), t: Long("13") },
    durableOpTime: { ts: Timestamp({ t: 1667627511, i: 1 }), t: Long("13") },
    lastAppliedWallTime: ISODate("2022-11-05T05:51:51.055Z"),
    lastDurableWallTime: ISODate("2022-11-05T05:51:51.055Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1667627496, i: 1 }),
  electionParticipantMetrics: {
    votedForCandidate: true,
    electionTerm: Long("13"),
    lastVoteDate: ISODate("2022-11-05T05:48:51.971Z"),
    electionCandidateMemberId: 1,
    voteReason: '',
    lastAppliedOpTimeAtElection: { ts: Timestamp({ t: 1667540936, i: 1 }), t: Long("11")
 },    maxAppliedOpTimeInSet: { ts: Timestamp({ t: 1667540936, i: 1 }), t: Long("11") },

    priorityAtElection: 1,
    newTermStartDate: ISODate("2022-11-05T05:48:51.992Z"),
    newTermAppliedDate: ISODate("2022-11-05T05:48:52.034Z")
  },
  members: [
    {
      _id: 0,
      name: '172.31.90.97:27019',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 196,
      optime: { ts: Timestamp({ t: 1667627511, i: 1 }), t: Long("13") },
      optimeDate: ISODate("2022-11-05T05:51:51.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T05:51:51.055Z"),
      lastDurableWallTime: ISODate("2022-11-05T05:51:51.055Z"),
      syncSourceHost: '172.31.89.25:27019',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 1,
      configTerm: 13,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: '172.31.89.25:27019',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 189,
      optime: { ts: Timestamp({ t: 1667627510, i: 1 }), t: Long("13") },
      optimeDurable: { ts: Timestamp({ t: 1667627510, i: 1 }), t: Long("13") },
      optimeDate: ISODate("2022-11-05T05:51:50.000Z"),
      optimeDurableDate: ISODate("2022-11-05T05:51:50.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T05:51:50.055Z"),
      lastDurableWallTime: ISODate("2022-11-05T05:51:50.055Z"),
      lastHeartbeat: ISODate("2022-11-05T05:51:50.565Z"),
      lastHeartbeatRecv: ISODate("2022-11-05T05:51:50.045Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1667627331, i: 1 }),
      electionDate: ISODate("2022-11-05T05:48:51.000Z"),
      configVersion: 1,
      configTerm: 13
    },
    {
      _id: 2,
      name: '172.31.86.144:27019',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 183,
      optime: { ts: Timestamp({ t: 1667627510, i: 1 }), t: Long("13") },
      optimeDurable: { ts: Timestamp({ t: 1667627510, i: 1 }), t: Long("13") },
      optimeDate: ISODate("2022-11-05T05:51:50.000Z"),
      optimeDurableDate: ISODate("2022-11-05T05:51:50.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T05:51:50.055Z"),
      lastDurableWallTime: ISODate("2022-11-05T05:51:50.055Z"),
      lastHeartbeat: ISODate("2022-11-05T05:51:50.565Z"),
      lastHeartbeatRecv: ISODate("2022-11-05T05:51:50.564Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.89.25:27019',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 1,
      configTerm: 13
    }
  ],
  ok: 1,
  lastCommittedOpTime: Timestamp({ t: 1667627510, i: 1 }),
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1667627511, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1667627511, i: 1 })
}
crs [direct: secondary] test>
```

## 7. Connect to Mongos

Command:
mongos --configdb crs/172.31.90.97:27019,172.31.89.25:27019,172.31.86.144:27019 --port 27020 --bind_ip_all

Steps: In mongos instance, run the above command.

Connecting Mongos to each config server:

```
mongos
ubuntu@ip-172-31-80-79:~$ mongos --configdb crs/172.31.90.97:27019,172.31.89.25:27019,172.31.86.144:27019 --port 27020 --bind_ip_all
{"t":{"$date":"2022-11-05T06:56:39.625+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"-","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify -
-sslDisabledProtocols 'none'"}
{"t":{"$date":"2022-11-05T06:56:39.626+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main","msg":"Initialized wire specification","attr":{"spec":{"incomingExtern
alClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":17},"outgoing":{"minWireVersion":17,"maxWireVersion":17},
"isInternalClient":true}}}
{"t":{"$date":"2022-11-05T06:56:39.633+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is required,
 set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
{"t":{"$date":"2022-11-05T06:56:39.633+00:00"},"s":"I",  "c":"HEALTH",   "id":5936503, "ctx":"main","msg":"Fault manager changed state ","attr":{"state":"StartupCheck"}}
{"t":{"$date":"2022-11-05T06:56:39.633+00:00"},"s":"W",  "c":"CONTROL",  "id":22120,   "ctx":"main","msg":"Access control is not enabled for the database. Read and write
access to data and configuration is unrestricted","tags":["startupWarnings"]}
{"t":{"$date":"2022-11-05T06:56:39.633+00:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"mongosMain","msg":"Build Info","attr":{"buildInfo":{"version":"6.0.2","gitVe
rsion":"94fb7dfc8b974f1f5343e7ea394d0d9deedba50e","openSSLVersion":"OpenSSL 1.1.1f  31 Mar 2020","modules":[],"allocator":"tcmalloc","environment":{"distmod":"ubuntu2004"
,"distarch":"x86_64","target_arch":"x86_64"}}}}
{"t":{"$date":"2022-11-05T06:56:39.633+00:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"mongosMain","msg":"Operating System","attr":{"os":{"name":"Ubuntu","version"
:"20.04"}}}
```

## 8. Shard Set-up

Shard Set-up: There are 3 shards *a*, *b*, and *c* in each of the nodes shardNode1, shardNode2, and shardNode3. They are deployed in a replica set; hence, each node has a replica for each shard.
Primary for *a* is shardNode1. Also, the replica set of *a* has priority 4 for shardNode1, and priority 1 for others.
Primary for *b* is shardNode2. Also, the replica set of *b* has priority 4 for shardNode2, and priority 1 for others.
Primary for *c* is shardNode3. Also, the replica set of *c* has priority 4 for shardNode3, and priority 1 for others.

Commands:

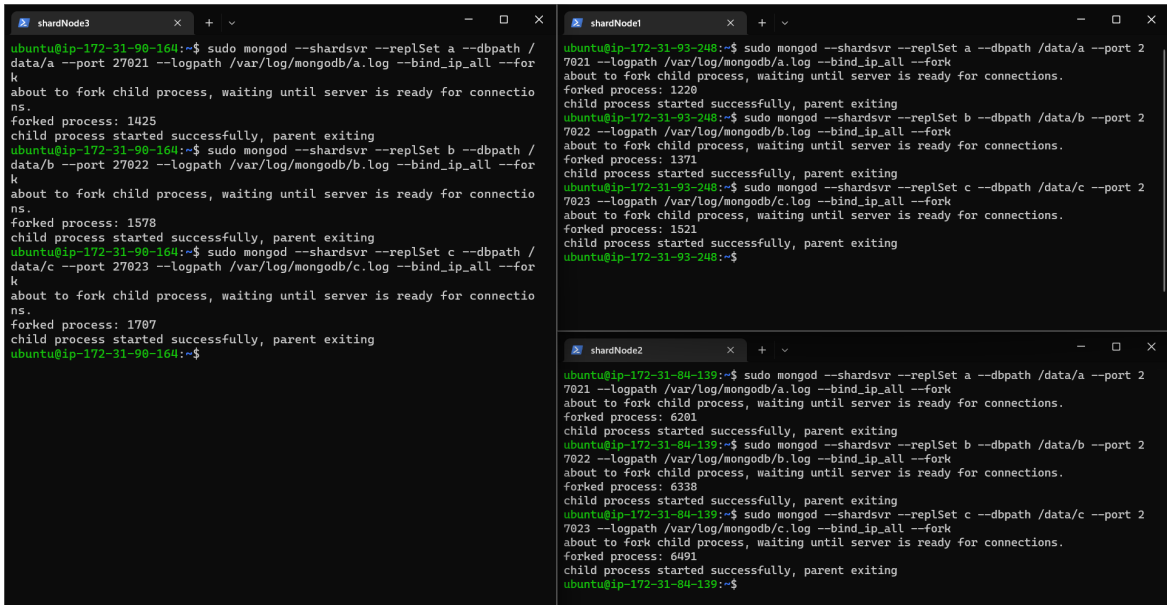
```
sudo mongod --shardsvr --replSet a --dbpath /data/a --port 27021 --logpath /var/log/mongodb/a.log
--bind_ip_all --fork

sudo mongod --shardsvr --replSet b --dbpath /data/b --port 27022 --logpath /var/log/mongodb/b.log
--bind_ip_all --fork

sudo mongod --shardsvr --replSet c --dbpath /data/c --port 27023 --logpath /var/log/mongodb/c.log
--bind_ip_all --fork
```

Run the above commands in each of the nodes shardNode1, shardNode2 and shardNode3 to run the shard servers.

Launching 3 shards in a replica set:




To initiate replica set for shard *a*, run commands:

mongosh -port 27021

rs.initiate({_id: "a", members : [ {_id: 0, host: "172.31.93.248:27021"}, {_id: 1, host: "172.31.84.139:27021"}, {_id: 2, host: "172.31.90.164:27021"} ] } )

Change Priority of replica set member:
conf = rs.conf()
conf.members[0].priority = 4
rs.reconfig(conf)

To initiate replica set for shard *b*, run commands:

mongosh -port 27022

rs.initiate({_id: "b", members : [ {_id: 0, host: "172.31.93.248:27022"}, {_id: 1, host: "172.31.84.139:27022"}, {_id: 2, host: "172.31.90.164:27022"} ] } )

Change Priority of replica set member:
conf = rs.conf()
conf.members[1].priority = 4
rs.reconfig(conf)

To initiate replica set for shard *c*, run commands:

mongosh -port 27023

rs.initiate({_id: "c", members : [ {_id: 0, host: "172.31.93.248:27023"}, {_id: 1, host: "172.31.84.139:27023"}, {_id: 2, host: "172.31.90.164:27023"} ] } )

Change Priority of replica set member:
conf = rs.conf()
conf.members[2].priority = 4
rs.reconfig(conf)

I changed the priority of the replica set to ensure that the primary of every node is on a different shard and one node is handling requests from only one shard when all nodes are healthy. Also, I kept replicas on different nodes to ensure the availability of all shards in case of failure.

Shard status from mongos:
(Before adding shards)

```
ubuntu@ip-172-31-80-79:~$ mongosh -port 27020
Current Mongosh Log ID: 6360c161a09640d313114d69
Connecting to:          mongodb://127.0.0.1:27020/?directConnection=true&serverSelectionTim
eoutMS=2000&appName=mongosh+1.6.0
Using MongoDB:          6.0.2
Using Mongosh:          1.6.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2022-11-01T06:46:15.471+00:00: Access control is not enabled for the database. Read and
write access to data and configuration is unrestricted
------

[direct: mongos] test> sh.status()
shardingVersion
{
  _id: 1,
  minCompatibleVersion: 5,
  currentVersion: 6,
  clusterId: ObjectId("6360713d8b3f67da20865b71")
}
---
shards
[]
---
active mongoses
[]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently enabled': 'yes',
  'Currently running': 'no',
  'Failed balancer rounds in last 5 attempts': 0,
  'Migration Results for the last 24 hours': 'No recent migrations'
}
---
databases
[
  {
    database: { _id: 'config', primary: 'config', partitioned: true },
    collections: {}
  }
]
[direct: mongos] test>
```

## 9. Adding Shards

In mongos instance, run commands:

```
mongosh -port 27020

sh.addShard( "a/172.31.93.248:27021,172.31.84.139:27021,172.31.90.164:27021" )
sh.addShard( "b/172.31.93.248:27022,172.31.84.139:27022,172.31.90.164:27022" )
sh.addShard( "c/172.31.93.248:27023,172.31.84.139:27023,172.31.90.164:27023" )
```

Shard status After adding shards:

```
ubuntu@ip-172-31-80-79:~$ mongosh -port 27020
Current Mongosh Log ID: 6360c7f0967d65c120c438f0
Connecting to:          mongodb://127.0.0.1:27020/?directConnection=true&serverSele
ctionTimeoutMS=2000&appName=mongosh+1.6.0
Using MongoDB:          6.0.2
Using Mongosh:          1.6.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2022-11-01T06:46:15.471+00:00: Access control is not enabled for the database. R
ead and write access to data and configuration is unrestricted
------

[direct: mongos] test> sh.status()
shardingVersion
{
  _id: 1,
  minCompatibleVersion: 5,
  currentVersion: 6,
  clusterId: ObjectId("6360713d8b3f67da20865b71")
}
---
shards
[
  {
    _id: 'a',
    host: 'a/172.31.84.139:27021,172.31.90.164:27021,172.31.93.248:27021',
    state: 1,
    topologyTime: Timestamp({ t: 1667285858, i: 1 })
  },
  {
    _id: 'b',
    host: 'b/172.31.84.139:27022,172.31.90.164:27022,172.31.93.248:27022',
    state: 1,
    topologyTime: Timestamp({ t: 1667285956, i: 2 })
  },
  {
    _id: 'c',
    host: 'c/172.31.84.139:27023,172.31.90.164:27023,172.31.93.248:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1667285967, i: 7 })
```

```
    state: 1,
    topologyTime: Timestamp({ t: 1667285967, i: 7 })
  }
]
---
active mongoses
[ { '6.0.2': 1 } ]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently enabled': 'yes',
  'Currently running': 'no',
  'Failed balancer rounds in last 5 attempts': 0,
  'Migration Results for the last 24 hours': { '342': 'Success' }
}
---
databases
[
  {
    database: { _id: 'config', primary: 'config', partitioned: true },
    collections: {
      'config.system.sessions': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [
          { shard: 'a', nChunks: 682 },
          { shard: 'b', nChunks: 171 },
          { shard: 'c', nChunks: 171 }
        ],
        chunks: [
          'too many chunks to print, use verbose if you want to force print'
        ],
        tags: []
      }
    }
  }
]
[direct: mongos] test>
```

## 10. Shard Key & Sharding Strategy

Shard Key: Random (Field: ‘title’, index: hashed)
Sharding Strategy: Hash-based

Commands:

```
use nosql_mid
sh.enableSharding("nosql_mid")
db.books.createIndex({"title":"hashed"})
sh.shardCollection("nosql_mid.books", {title:"hashed"})
use config
db.settings.updateOne(
  { _id: "chunksize" },
  { $set: { _id: "chunksize", value: 1} },
  { upsert: true }
)
```

## 11. Adding Data from Public Dataset

Dataset URL: https://www.kaggle.com/datasets/opalskies/large-books-metadata-dataset-50-mill-entries
File from above URL: list.json

Collection Description:
The collection is about suggestions on books. Every document in this collection has fields like the number of voters, number of likes, an array of descriptions of suggested books, a list of tags, and so on.

One Random Document from the above Dataset:


```
_id: 29
title: "Best Books of 2007"
description: "The best books published during 2007.See best-of-year lists for other ..."
description_html: "
    The best books published during 2007.<br><br/>See best-of-year ..."
num_pages: 10
num_books: 916
num_voters: 1178
created_date: "June 21st, 2008"
tags: Array
  0: "2007"
  1: "best"
num_likes: 30
created_by: Object
  name: "deleted user"
  id: ""
num_comments: 6
books: Array
  0: Object
    book_id: "136251"
    title: "Harry Potter and the Deathly Hallows"
    author_id: "1077326"
    author: "J.K. Rowling"
    position: Object
      ranking: 1
      score: 37269
      votes: 376
  1: Object
  2: Object
```


To insert data:

1. Download the list.json file from the link above and convert it to json format if needed.
2. Connect to mongos server from the MongoDB-Compass desktop application.
3. Import data in the collection ‘books’.

Shard status After populating Data in a sharded collection:

mongos

```
[direct: mongos] nosql_mid> sh.status()
shardingVersion
{
  _id: 1,
  minCompatibleVersion: 5,
  currentVersion: 6,
  clusterId: ObjectId("6360713d8b3f67da20865b71")
}
---
shards
[
  {
    _id: 'a',
    host: 'a/172.31.84.139:27021,172.31.90.164:27021,172.31.93.248:27021',
    state: 1,
    topologyTime: Timestamp({ t: 1667285858, i: 1 })
  },
  {
    _id: 'b',
    host: 'b/172.31.84.139:27022,172.31.90.164:27022,172.31.93.248:27022',
    state: 1,
    topologyTime: Timestamp({ t: 1667285956, i: 2 })
  },
  {
    _id: 'c',
    host: 'c/172.31.84.139:27023,172.31.90.164:27023,172.31.93.248:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1667285967, i: 7 })
  }
]
---
active mongoses
[ { '6.0.2': 1 } ]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently enabled': 'yes',
  'Currently running': 'no',
  'Failed balancer rounds in last 5 attempts': 0,
  'Migration Results for the last 24 hours': 'No recent migrations'
}
---
databases
[
  {
    database: {
      _id: 'books',
      primary: 'a',
      partitioned: false,
      version: {
        uuid: new UUID("b7f30595-c31e-4572-b9f8-6e6dcccfb8f9"),
        timestamp: Timestamp({ t: 1667363566, i: 7 }),
        lastMod: 1
      }
    },
    collections: {}
  },
  {
    database: { _id: 'config', primary: 'config', partitioned: true },
    collections: {
      'config.system.sessions': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [
          { shard: 'a', nChunks: 342 },
          { shard: 'b', nChunks: 341 },
          { shard: 'c', nChunks: 341 }
        ],
        chunks: [
          'too many chunks to print, use verbose if you want to force print'
        ],
        tags: []
      }
    }
  },
  {
    database: {
      _id: 'nosql_mid',
      primary: 'a',
      partitioned: false,
      version: {
        uuid: new UUID("8b024bad-aa49-4e39-9d88-45db841abd4b"),
        timestamp: Timestamp({ t: 1667363694, i: 3 }),
        lastMod: 1
      }
    },
    collections: {
      'nosql_mid.books': {
        shardKey: { title: 'hashed' },
        unique: false,
        balancing: true,
        chunkMetadata: [
          { shard: 'a', nChunks: 406 },
          { shard: 'b', nChunks: 405 },
          { shard: 'c', nChunks: 405 }
        ],
        chunks: [
          'too many chunks to print, use verbose if you want to force print'
        ],
        tags: []
      }
    }
  },
  {
    database: {
      _id: 'test',
      primary: 'c',
      partitioned: false,
      version: {
        uuid: new UUID("936d4893-98b7-4189-983d-6da0a60b14f1"),
        timestamp: Timestamp({ t: 1667370651, i: 1 }),
        lastMod: 1
      }
    },
    collections: {}
  },
```

Checking the shard distribution:

```
mongos
[direct: mongos] nosql_mid> db.books.getShardDistribution()
Shard a at a/172.31.84.139:27021,172.31.90.164:27021,172.31.93.248:27021
{
  data: '405.21MiB',
  docs: 28512,
  chunks: 406,
  'estimated data per chunk': '1022KiB',
  'estimated docs per chunk': 70
}
---
Shard c at c/172.31.84.139:27023,172.31.90.164:27023,172.31.93.248:27023
{
  data: '398.69MiB',
  docs: 26320,
  chunks: 405,
  'estimated data per chunk': '1008KiB',
  'estimated docs per chunk': 64
}
---
Shard b at b/172.31.84.139:27022,172.31.90.164:27022,172.31.93.248:27022
{
  data: '395.12MiB',
  docs: 26680,
  chunks: 405,
  'estimated data per chunk': '999KiB',
  'estimated docs per chunk': 65
}
---
Totals
{
  data: '1.17GiB',
  docs: 81512,
  chunks: 1216,
  'Shard a': [
    '33.79 % data',
    '34.97 % docs in cluster',
    '14KiB avg obj size on shard'
  ],
  'Shard c': [
    '33.25 % data',
    '32.28 % docs in cluster',
    '15KiB avg obj size on shard'
  ],
  'Shard b': [
    '32.95 % data',
    '32.73 % docs in cluster',
    '15KiB avg obj size on shard'
  ]
}
[direct: mongos] nosql_mid>
```

## 12. Running Queries on Data

1. Find documents in a given range
   Command:

   db.books.find({ num_likes: { $gte:3000, $lte:3500}}, {_id:1, title:1, num_likes:1}).explain('executionStats')

   Query Output:

```
mongos
[direct: mongos] nosql_mid> db.books.find({ num_likes: { $gte:3000, $lte:3500}}, {_id:1, title:1, num_likes:1})
[
  {
    _id: 50,
    title: 'The Best Epic Fantasy (fiction)',
    num_likes: 3078
  },
  { _id: 10762, title: 'Best Book Boyfriends', num_likes: 3092 },
  {
    _id: 72680,
    title: 'Gilmore Girls Complete Reading List',
    num_likes: 3491
  },
  {
    _id: 952,
    title: '1001 Books You Must Read Before You Die',
    num_likes: 3386
  },
  { _id: 15, title: 'Best Historical Fiction', num_likes: 3178 },
  {
    _id: 1058,
    title: 'Microhistory: Social Histories of Just One Thing',
    num_likes: 3158
  }
]
[direct: mongos] nosql_mid>
```

   Execution Stats:
   Execution Time: 3900
   Documents returned per shard:
   - a: 1
   - b: 3
   - c: 2

   Part of Explain command output showing the relevant data:

mongos


```
executionStats: {
  nReturned: 6,
  executionTimeMillis: 3900,
  totalKeysExamined: 0,
  totalDocsExamined: 81512,
  executionStages: {
    stage: 'SHARD_MERGE',
    nReturned: 6,
    executionTimeMillis: 3900,
    totalKeysExamined: 0,
    totalDocsExamined: 81512,
    totalChildMillis: Long("9394"),
    shards: [
      {
        shardName: 'b',
        executionSuccess: true,
        nReturned: 3,
        executionTimeMillis: 2260,
        totalKeysExamined: 0,
        totalDocsExamined: 26680,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 3,
          executionTimeMillisEstimate: 2150,
          works: 26682,
          advanced: 3,
          needTime: 26678,
          needYield: 0,
          saveState: 112,
          restoreState: 112,
          isEOF: 1,
          transformBy: { _id: 1, title: 1, num_likes: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 3,
            executionTimeMillisEstimate: 2150,
            works: 26682,
            advanced: 3,
            needTime: 26678,
            needYield: 0,
            saveState: 112,
            restoreState: 112,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                '$and': [
                  { num_likes: { '$lte': 3500 } },
                  { num_likes: { '$gte': 3000 } }
                ]
              },
              nReturned: 3,
              executionTimeMillisEstimate: 2150,
              works: 26682,
              advanced: 3,
              needTime: 26678,
              needYield: 0,
              saveState: 112,
              restoreState: 112,
              isEOF: 1,
              direction: 'forward',
              docsExamined: 26680
            }
          }
        }
      },
      {
        shardName: 'a',
        executionSuccess: true,
        nReturned: 1,
        executionTimeMillis: 3236,
        totalKeysExamined: 0,
        totalDocsExamined: 28512,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 1,
          executionTimeMillisEstimate: 2882,
          works: 28514,
          advanced: 1,
          needTime: 28512,
          needYield: 0,
          saveState: 152,
          restoreState: 152,
          isEOF: 1,
          transformBy: { _id: 1, title: 1, num_likes: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 1,
            executionTimeMillisEstimate: 2882,
            works: 28514,
            advanced: 1,
            needTime: 28512,
            needYield: 0,
            saveState: 152,
            restoreState: 152,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                '$and': [
                  { num_likes: { '$lte': 3500 } },
                  { num_likes: { '$gte': 3000 } }
                ]
              },
              nReturned: 1,
              executionTimeMillisEstimate: 2882,
              works: 28514,
              advanced: 1,
              needTime: 28512,
              needYield: 0,
              saveState: 152,
              restoreState: 152,
              isEOF: 1,
              direction: 'forward',
              docsExamined: 28512
            }
          }
        }
      },
      {
        shardName: 'c',
        executionSuccess: true,
        nReturned: 2,
        executionTimeMillis: 3898,
        totalKeysExamined: 0,
        totalDocsExamined: 26320,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 2,
          executionTimeMillisEstimate: 3822,
          works: 26322,
          advanced: 2,
          needTime: 26319,
          needYield: 0,
          saveState: 202,
          restoreState: 202,
          isEOF: 1,
          transformBy: { _id: 1, title: 1, num_likes: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 2,
            executionTimeMillisEstimate: 3822,
            works: 26322,
            advanced: 2,
            needTime: 26319,
            needYield: 0,
            saveState: 202,
            restoreState: 202,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                '$and': [
                  { num_likes: { '$lte': 3500 } },
                  { num_likes: { '$gte': 3000 } }
                ]
              },
              nReturned: 2,
              executionTimeMillisEstimate: 3822,
              works: 26322,
              advanced: 2,
              needTime: 26319,
              needYield: 0,
              saveState: 202,
              restoreState: 202,
              isEOF: 1,
              direction: 'forward',
              docsExamined: 26320
            }
          }
        }
      }
    ]
  }
},
serverInfo: {
  host: 'ip-172-31-80-79',
  port: 27020,
  version: '6.0.2',
  gitVersion: '94fb7dfc8b974f1f5343e7ea394d0d9deedba50e'
},
serverParameters: {
```

2. Query with $elemMatch with two conditions
   Command:

```
db.books.find({num_pages: {$in: [17, 71]}, num_books:{$nin: [1655, 1640, 7023]}, tags: {$all:['fiction', 'best']}}, {_id:0, title:1}).explain('executionStats')
```

   Query Output:

```
mongos

[direct: mongos] nosql_mid> db.books.find({num_pages: {$in: [17, 71]}, num_books:{$nin: [1655, 1640, 7023]}, tags: {$all:['fiction', 'best']}}, {_id:0, title:1})
[
  { title: 'Best Adult Vampire Books' },
  { title: 'Best Short Stories' },
  { title: 'Best Books of 2008' }
]
[direct: mongos] nosql_mid>
```

   Execution Stats:
   Execution Time: 314
   Documents returned per shard:
      a: 0
      b: 2
      c: 1

   Part of Explain command output showing the relevant data:
   (next page)

mongos

```
executionStats: {
  nReturned: 3,
  executionTimeMillis: 314,
  totalKeysExamined: 0,
  totalDocsExamined: 81512,
  executionStages: {
    stage: 'SHARD_MERGE',
    nReturned: 3,
    executionTimeMillis: 314,
    totalKeysExamined: 0,
    totalDocsExamined: 81512,
    totalChildMillis: Long("682"),
    shards: [
      {
        shardName: 'b',
        executionSuccess: true,
        nReturned: 2,
        executionTimeMillis: 144,
        totalKeysExamined: 0,
        totalDocsExamined: 26680,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 2,
          executionTimeMillisEstimate: 103,
          works: 26682,
          advanced: 2,
          needTime: 26679,
          needYield: 0,
          saveState: 29,
          restoreState: 29,
          isEOF: 1,
          transformBy: { _id: 0, title: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 2,
            executionTimeMillisEstimate: 103,
            works: 26682,
            advanced: 2,
            needTime: 26679,
            needYield: 0,
            saveState: 29,
            restoreState: 29,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                '$and': [
                  { tags: { '$eq': 'fiction' } },
                  { tags: { '$eq': 'best' } },
                  { num_pages: { '$in': [ 17, 71 ] } },
                  {
                    num_books: { '$not': { '$in': [ 1640, 1655, 7023 ] } }
                  }
                ]
              },
              nReturned: 2,
              executionTimeMillisEstimate: 103,
              works: 26682,
              advanced: 2,
              needTime: 26679,
              needYield: 0,
              saveState: 29,
              restoreState: 29,
              isEOF: 1,
              direction: 'forward',
              docsExamined: 26680
            }
          }
        }
      },
      {
        shardName: 'a',
        executionSuccess: true,
        nReturned: 0,
        executionTimeMillis: 226,
        totalKeysExamined: 0,
        totalDocsExamined: 28512,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 0,
          executionTimeMillisEstimate: 176,
          works: 28514,
          advanced: 0,
          needTime: 28513,
          needYield: 0,
          saveState: 31,
          restoreState: 31,
          isEOF: 1,
          transformBy: { _id: 0, title: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 0,
            executionTimeMillisEstimate: 176,
            works: 28514,
            advanced: 0,
            needTime: 28513,
            needYield: 0,
            saveState: 31,
            restoreState: 31,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                '$and': [
                  { tags: { '$eq': 'fiction' } },
                  { tags: { '$eq': 'best' } },
                  { num_pages: { '$in': [ 17, 71 ] } },
                  {
                    num_books: { '$not': { '$in': [ 1640, 1655, 7023 ] } }
                  }
                ]
              },
              nReturned: 0,
              executionTimeMillisEstimate: 176,
              works: 28514,
              advanced: 0,
              needTime: 28513,
              needYield: 0,
              saveState: 31,
              restoreState: 31,
              isEOF: 1,
              direction: 'forward',
              docsExamined: 28512
            }
          }
        }
      },
      {
        shardName: 'c',
        executionSuccess: true,
        nReturned: 1,
        executionTimeMillis: 312,
        totalKeysExamined: 0,
        totalDocsExamined: 26320,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 1,
          executionTimeMillisEstimate: 238,
          works: 26322,
          advanced: 1,
          needTime: 26320,
          needYield: 0,
          saveState: 31,
          restoreState: 31,
          isEOF: 1,
          transformBy: { _id: 0, title: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 1,
            executionTimeMillisEstimate: 238,
            works: 26322,
            advanced: 1,
            needTime: 26320,
            needYield: 0,
            saveState: 31,
            restoreState: 31,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                '$and': [
                  { tags: { '$eq': 'fiction' } },
                  { tags: { '$eq': 'best' } },
                  { num_pages: { '$in': [ 17, 71 ] } },
                  {
                    num_books: { '$not': { '$in': [ 1640, 1655, 7023 ] } }
                  }
                ]
              },
              nReturned: 1,
              executionTimeMillisEstimate: 238,
              works: 26322,
              advanced: 1,
              needTime: 26320,
              needYield: 0,
              saveState: 31,
              restoreState: 31,
              isEOF: 1,
              direction: 'forward',
              docsExamined: 26320
            }
          }
```

3. Query with $in, $nin, or $all

   Command:

   db.books.find({books: {"$elemMatch": {"position.ranking":{$gt: 2000, $lt:2050}, author_id:{$in:["3618", "7628"]}}}}, {_id:0, title:1}).explain('executionStats')

   Query Output:

```
mongos

[direct: mongos] nosql_mid> db.books.find({books: {"$elemMatch": {"position.ranking":{$gt: 2000, $lt:2050}, author_id:{$in:["3618", "7628"]}}}}, {_id:0, title:1})
[
  { title: 'What Book Would You Like To Live In?' },
  { title: 'Best Books of the 21st Century' },
  { title: 'Books that Changed the Way You View Life' },
  {
    title: 'Books I Became So Obsessed With I Stopped Everything Else in My Life to Finish.'
  }
]
[direct: mongos] nosql_mid>
```

   Execution Stats:

   Execution Time: 1218

   Documents returned per shard:

   a: 2

   b: 1

   c: 1

   Part of Explain command output showing the relevant data:

   (next page)

mongos

```
executionStats: {
  nReturned: 4,
  executionTimeMillis: 1218,
  totalKeysExamined: 0,
  totalDocsExamined: 81512,
  executionStages: {
    stage: 'SHARD_MERGE',
    nReturned: 4,
    executionTimeMillis: 1218,
    totalKeysExamined: 0,
    totalDocsExamined: 81512,
    totalChildMillis: Long("3551"),
    shards: [
      {
        shardName: 'b',
        executionSuccess: true,
        nReturned: 1,
        executionTimeMillis: 1137,
        totalKeysExamined: 0,
        totalDocsExamined: 26680,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 1,
          executionTimeMillisEstimate: 1125,
          works: 26682,
          advanced: 1,
          needTime: 26680,
          needYield: 0,
          saveState: 60,
          restoreState: 60,
          isEOF: 1,
          transformBy: { _id: 0, title: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 1,
            executionTimeMillisEstimate: 1125,
            works: 26682,
            advanced: 1,
            needTime: 26680,
            needYield: 0,
            saveState: 60,
            restoreState: 60,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                books: {
                  '$elemMatch': {
                    '$and': [
                      { 'position.ranking': { '$lt': 2050 } },
                      { 'position.ranking': { '$gt': 2000 } },
                      { author_id: { '$in': [ '3618', '7628' ] } }
                    ]
                  }
                }
              },
              nReturned: 1,
              executionTimeMillisEstimate: 1125,
              works: 26682,
              advanced: 1,
              needTime: 26680,
              needYield: 0,
              saveState: 60,
              restoreState: 60,
              isEOF: 1,
              direction: 'forward',
              docsExamined: 26680
            }
          }
        }
      },
      {
        shardName: 'c',
        executionSuccess: true,
        nReturned: 1,
        executionTimeMillis: 1198,
        totalKeysExamined: 0,
        totalDocsExamined: 26320,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 1,
          executionTimeMillisEstimate: 1186,
          works: 26322,
          advanced: 1,
          needTime: 26320,
          needYield: 0,
          saveState: 62,
          restoreState: 62,
          isEOF: 1,
          transformBy: { _id: 0, title: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 1,
            executionTimeMillisEstimate: 1185,
            works: 26322,
            advanced: 1,
            needTime: 26320,
            needYield: 0,
            saveState: 62,
            restoreState: 62,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                books: {
                  '$elemMatch': {
                    '$and': [
                      { 'position.ranking': { '$lt': 2050 } },
                      { 'position.ranking': { '$gt': 2000 } },
                      { author_id: { '$in': [ '3618', '7628' ] } }
                    ]
                  }
                }
              },
              nReturned: 1,
              executionTimeMillisEstimate: 1185,
              works: 26322,
              advanced: 1,
              needTime: 26320,
              needYield: 0,
              saveState: 62,
              restoreState: 62,
              isEOF: 1,
              direction: 'forward',
              docsExamined: 26320
            }
          }
        }
      },
      {
        shardName: 'a',
        executionSuccess: true,
        nReturned: 2,
        executionTimeMillis: 1216,
        totalKeysExamined: 0,
        totalDocsExamined: 28512,
        executionStages: {
          stage: 'PROJECTION_SIMPLE',
          nReturned: 2,
          executionTimeMillisEstimate: 1207,
          works: 28514,
          advanced: 2,
          needTime: 28511,
          needYield: 0,
          saveState: 66,
          restoreState: 66,
          isEOF: 1,
          transformBy: { _id: 0, title: 1 },
          inputStage: {
            stage: 'SHARDING_FILTER',
            nReturned: 2,
            executionTimeMillisEstimate: 1207,
            works: 28514,
            advanced: 2,
            needTime: 28511,
            needYield: 0,
            saveState: 66,
            restoreState: 66,
            isEOF: 1,
            chunkSkips: 0,
            inputStage: {
              stage: 'COLLSCAN',
              filter: {
                books: {
                  '$elemMatch': {
                    '$and': [
                      { 'position.ranking': { '$lt': 2050 } },
                      { 'position.ranking': { '$gt': 2000 } },
                      { author_id: { '$in': [ '3618', '7628' ] } }
                    ]
                  }
                }
              },
              nReturned: 2,
              executionTimeMillisEstimate: 1207,
              works: 28514,
              advanced: 2,
              needTime: 28511,
              needYield: 0,
              saveState: 66,
              restoreState: 66,
              isEOF: 1,
              direction: 'forward',
```

4. Query involving aggregate()

Command:

db.books.aggregate([{ $match: { "created_date": { $in: ["May 1st, 2020", "May 2nd, 2020", "May 3rd, 2020"] } } }, { $group: { _id: '$created_date', 'average_votes': { $avg: "$num_voters" } } }, { $sort: { average_votes: -1 } }]).explain('executionStats')

Query Output:

```
mongos

[direct: mongos] nosql_mid> db.books.aggregate([{ $match: { "created_date": { $in: ["May 1st, 2020", "May 2nd, 2020", "May 3rd, 2020"] } } }, { $group: { _id: '$created_date', 'average_votes': { $avg: "$num_voters" } } }, { $sort: { average_votes: -1 } }])
[
  { _id: 'May 2nd, 2020', average_votes: 10.666666666666666 },
  { _id: 'May 3rd, 2020', average_votes: 5.16 },
  { _id: 'May 1st, 2020', average_votes: 3.310344827586207 }
]
[direct: mongos] nosql_mid>
```

Execution Stats:

Execution Time:

a: 175 ms

b: 97 ms

c: 158 ms

Documents returned per shard:

a: 3

b: 3

c: 3

5. Query involving mapReduce()

Command:

var m_start = new Date(); db.books.mapReduce(function(){ emit(this.created_date, this.num_voters)}, function(created_date,num_voters){ return Array.avg(num_voters);}, {out :"average_votes"} ); var m_end = new Date(); execution_10 = m_end - m_start

db.average_votes.find({ "_id": { $in: ["May 1st, 2020", "May 2nd, 2020", "May 3rd, 2020"] } } )

Query Output:

```
[direct: mongos] nosql_mid> db.books.mapReduce(function(){ emit(this.created_date, this.num_voters)}, function(created_date,num_voters){ return Array.avg(num_voters);}, {out :"average_votes"} )
{
  result: 'average_votes',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1667636997, i: 4524 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1667636997, i: 4524 })
}
[direct: mongos] nosql_mid> db.average_votes.find({ "_id": { $in: ["May 1st, 2020", "May 2nd, 2020", "May 3rd, 2020"] } } )
[
  { _id: 'May 1st, 2020', value: 3.3611111111111107 },
  { _id: 'May 2nd, 2020', value: 9.207070707070708 },
  { _id: 'May 3rd, 2020', value: 4.978835978835979 }
]
[direct: mongos] nosql_mid>
```

Execution Stats:

Execution time: 2343 ms

6. Update query
   Command:

   var m_start = new Date(); db.books.updateOne({title: "Best books of April, 2008"}, {$inc: {num_voters: 1, num_likes: 1}}); var m_end = new Date(); execution_10 = m_end - m_start

   Query Output:

```
mongos

[direct: mongos] nosql_mid> db.books.updateOne({title: "Best books of April, 2008"}, {$inc: {num_voters: 1, num_likes: 1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
[direct: mongos] nosql_mid>
```

   Execution Stats:
   Execution time: 12ms

## 13. Replication of Shards

// rs.status image of shards

Replication Status of 3 Shards: shard a, shard b, & shard c:
(next three pages)

shardNode1

```
a [direct: primary] test> rs.status()
{
  set: 'a',
  date: ISODate("2022-11-05T06:38:15.319Z"),
  myState: 1,
  term: Long("14"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
    lastCommittedWallTime: ISODate("2022-11-05T06:38:07.546Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
    appliedOpTime: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
    durableOpTime: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
    lastAppliedWallTime: ISODate("2022-11-05T06:38:07.546Z"),
    lastDurableWallTime: ISODate("2022-11-05T06:38:07.546Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1667630237, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'priorityTakeover',
    lastElectionDate: ISODate("2022-11-05T06:16:37.482Z"),
    electionTerm: Long("14"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1667628996, i: 1 }), t: Long("13") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1667628996, i: 1 }), t: Long("13") },
    numVotesNeeded: 2,
    priorityAtElection: 4,
    electionTimeoutMillis: Long("10000"),
    priorPrimaryMemberId: 2,
    numCatchUpOps: Long("0"),
    newTermStartDate: ISODate("2022-11-05T06:16:37.495Z"),
    wMajorityWriteAvailabilityDate: ISODate("2022-11-05T06:16:38.494Z")
  },
  electionParticipantMetrics: {
    votedForCandidate: true,
    electionTerm: Long("13"),
    lastVoteDate: ISODate("2022-11-05T06:16:26.368Z"),
    electionCandidateMemberId: 2,
    voteReason: '',
    lastAppliedOpTimeAtElection: { ts: Timestamp({ t: 1667540934, i: 2 }), t: Long("11") },
    maxAppliedOpTimeInSet: { ts: Timestamp({ t: 1667540934, i: 2 }), t: Long("11") },
    priorityAtElection: 4
  },
  members: [
    {
      _id: 0,
      name: '172.31.93.248:27021',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 1318,
      optime: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
      optimeDate: ISODate("2022-11-05T06:38:07.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:07.546Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:07.546Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1667628997, i: 1 }),
      electionDate: ISODate("2022-11-05T06:16:37.000Z"),
      configVersion: 2,
      configTerm: 14,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: '172.31.84.139:27021',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 1309,
      optime: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
      optimeDurable: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
      optimeDate: ISODate("2022-11-05T06:38:07.000Z"),
      optimeDurableDate: ISODate("2022-11-05T06:38:07.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:07.546Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:07.546Z"),
      lastHeartbeat: ISODate("2022-11-05T06:38:14.025Z"),
      lastHeartbeatRecv: ISODate("2022-11-05T06:38:14.024Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.90.164:27021',
      syncSourceId: 2,
      infoMessage: '',
      configVersion: 2,
      configTerm: 14
    },
    {
      _id: 2,
      name: '172.31.90.164:27021',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 1315,
      optime: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
      optimeDurable: { ts: Timestamp({ t: 1667630287, i: 1 }), t: Long("14") },
      optimeDate: ISODate("2022-11-05T06:38:07.000Z"),
      optimeDurableDate: ISODate("2022-11-05T06:38:07.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:07.546Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:07.546Z"),
      lastHeartbeat: ISODate("2022-11-05T06:38:14.026Z"),
      lastHeartbeatRecv: ISODate("2022-11-05T06:38:15.163Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.93.248:27021',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 2,
      configTerm: 14
    }
  ],
  ok: 1,
  lastCommittedOpTime: Timestamp({ t: 1667630287, i: 1 }),
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1667630289, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
```

shardNode2

```
b [direct: primary] test> rs.status()
{
  set: 'b',
  date: ISODate("2022-11-05T06:38:31.641Z"),
  myState: 1,
  term: Long("13"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
    lastCommittedWallTime: ISODate("2022-11-05T06:38:22.879Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
    appliedOpTime: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
    durableOpTime: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
    lastAppliedWallTime: ISODate("2022-11-05T06:38:22.879Z"),
    lastDurableWallTime: ISODate("2022-11-05T06:38:22.879Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1667630262, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'priorityTakeover',
    lastElectionDate: ISODate("2022-11-05T06:18:02.783Z"),
    electionTerm: Long("13"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1667629081, i: 1 }), t: Long("12") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1667629081, i: 1 }), t: Long("12") },
    numVotesNeeded: 2,
    priorityAtElection: 4,
    electionTimeoutMillis: Long("10000"),
    priorPrimaryMemberId: 0,
    numCatchUpOps: Long("0"),
    newTermStartDate: ISODate("2022-11-05T06:18:02.819Z"),
    wMajorityWriteAvailabilityDate: ISODate("2022-11-05T06:18:03.873Z")
  },
  electionParticipantMetrics: {
    votedForCandidate: true,
    electionTerm: Long("12"),
    lastVoteDate: ISODate("2022-11-05T06:17:51.649Z"),
    electionCandidateMemberId: 0,
    voteReason: '',
    lastAppliedOpTimeAtElection: { ts: Timestamp({ t: 1667540932, i: 1 }), t: Long("10") },
    maxAppliedOpTimeInSet: { ts: Timestamp({ t: 1667540932, i: 1 }), t: Long("10") },
    priorityAtElection: 4
  },
  members: [
    {
      _id: 0,
      name: '172.31.93.248:27022',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 1245,
      optime: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
      optimeDurable: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
      optimeDate: ISODate("2022-11-05T06:38:22.000Z"),
      optimeDurableDate: ISODate("2022-11-05T06:38:22.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:22.879Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:22.879Z"),
      lastHeartbeat: ISODate("2022-11-05T06:38:31.370Z"),
      lastHeartbeatRecv: ISODate("2022-11-05T06:38:30.623Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.84.139:27022',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 2,
      configTerm: 13
    },
    {
      _id: 1,
      name: '172.31.84.139:27022',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 1247,
      optime: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
      optimeDate: ISODate("2022-11-05T06:38:22.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:22.879Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:22.879Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1667629082, i: 1 }),
      electionDate: ISODate("2022-11-05T06:18:02.000Z"),
      configVersion: 2,
      configTerm: 13,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 2,
      name: '172.31.90.164:27022',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 1245,
      optime: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
      optimeDurable: { ts: Timestamp({ t: 1667630302, i: 1 }), t: Long("13") },
      optimeDate: ISODate("2022-11-05T06:38:22.000Z"),
      optimeDurableDate: ISODate("2022-11-05T06:38:22.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:22.879Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:22.879Z"),
      lastHeartbeat: ISODate("2022-11-05T06:38:31.369Z"),
      lastHeartbeatRecv: ISODate("2022-11-05T06:38:31.413Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.93.248:27022',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 2,
      configTerm: 13
    }
  ],
  ok: 1,
  lastCommittedOpTime: Timestamp({ t: 1667630302, i: 1 }),
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1667630309, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
```

shardNode3

```
c [direct: primary] test> rs.status()
{
  set: 'c',
  date: ISODate("2022-11-05T06:38:40.783Z"),
  myState: 1,
  term: Long("17"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
    lastCommittedWallTime: ISODate("2022-11-05T06:38:35.060Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
    appliedOpTime: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
    durableOpTime: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
    lastAppliedWallTime: ISODate("2022-11-05T06:38:35.060Z"),
    lastDurableWallTime: ISODate("2022-11-05T06:38:35.060Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1667630266, i: 2 }),
  electionCandidateMetrics: {
    lastElectionReason: 'priorityTakeover',
    lastElectionDate: ISODate("2022-11-05T06:18:24.977Z"),
    electionTerm: Long("17"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1667629103, i: 1 }), t: Long("16") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1667629103, i: 1 }), t: Long("16") },
    numVotesNeeded: 2,
    priorityAtElection: 4,
    electionTimeoutMillis: Long("10000"),
    priorPrimaryMemberId: 0,
    numCatchUpOps: Long("0"),
    newTermStartDate: ISODate("2022-11-05T06:18:24.990Z"),
    wMajorityWriteAvailabilityDate: ISODate("2022-11-05T06:18:25.992Z")
  },
  electionParticipantMetrics: {
    votedForCandidate: true,
    electionTerm: Long("16"),
    lastVoteDate: ISODate("2022-11-05T06:18:13.542Z"),
    electionCandidateMemberId: 0,
    voteReason: '',
    lastAppliedOpTimeAtElection: { ts: Timestamp({ t: 1667540927, i: 1 }), t: Long("14") },
    maxAppliedOpTimeInSet: { ts: Timestamp({ t: 1667540927, i: 1 }), t: Long("14") },
    priorityAtElection: 4
  },
  members: [
    {
      _id: 0,
      name: '172.31.93.248:27023',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 1237,
      optime: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
      optimeDurable: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
      optimeDate: ISODate("2022-11-05T06:38:35.000Z"),
      optimeDurableDate: ISODate("2022-11-05T06:38:35.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:35.060Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:35.060Z"),
      lastHeartbeat: ISODate("2022-11-05T06:38:39.559Z"),
      lastHeartbeatRecv: ISODate("2022-11-05T06:38:40.662Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.90.164:27023',
      syncSourceId: 2,
      infoMessage: '',
      configVersion: 2,
      configTerm: 17
    },
    {
      _id: 1,
      name: '172.31.84.139:27023',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 1234,
      optime: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
      optimeDurable: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
      optimeDate: ISODate("2022-11-05T06:38:35.000Z"),
      optimeDurableDate: ISODate("2022-11-05T06:38:35.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:35.060Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:35.060Z"),
      lastHeartbeat: ISODate("2022-11-05T06:38:39.559Z"),
      lastHeartbeatRecv: ISODate("2022-11-05T06:38:39.558Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.90.164:27023',
      syncSourceId: 2,
      infoMessage: '',
      configVersion: 2,
      configTerm: 17
    },
    {
      _id: 2,
      name: '172.31.90.164:27023',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 1245,
      optime: { ts: Timestamp({ t: 1667630315, i: 1 }), t: Long("17") },
      optimeDate: ISODate("2022-11-05T06:38:35.000Z"),
      lastAppliedWallTime: ISODate("2022-11-05T06:38:35.060Z"),
      lastDurableWallTime: ISODate("2022-11-05T06:38:35.060Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1667629104, i: 1 }),
      electionDate: ISODate("2022-11-05T06:18:24.000Z"),
      configVersion: 2,
      configTerm: 17,
      self: true,
      lastHeartbeatMessage: ''
    }
  ],
  ok: 1,
  lastCommittedOpTime: Timestamp({ t: 1667630315, i: 1 }),
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1667630319, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
```

## 14. List of Hosts

Please see the Deployment Summary in part 1 for a detailed explanation.

| Node Number | Instance/ Node Name | Port | Service | IP Address (Private) |
|---|---|---|---|---|
| Node 1 | mongos | 27020 | Shard Controller | 172.31.80.79 |
| Node 2 | config1 | 27019 | Config Server Primary | 172.31.90.97 |
| Node 3 | config2 | 27019 | Config Server Secondary 1 | 172.31.89.25 |
| Node 4 | config3 | 27019 | Config Server Secondary 2 | 172.31.86.144 |
| Node 5 | shardNode1 | 27021 | Shard a Primary | 172.31.93.248 |
| Node 5 | shardNode1 | 27022 | Shard b Secondary 1 | 172.31.93.248 |
| Node 5 | shardNode1 | 27023 | Shard c Secondary 1 | 172.31.93.248 |
| Node 6 | shardNode2 | 27021 | Shard a Secondary 1 | 172.31.84.139 |
| Node 6 | shardNode2 | 27022 | Shard b Primary | 172.31.84.139 |
| Node 6 | shardNode2 | 27023 | Shard c Secondary 2 | 172.31.84.139 |
| Node 7 | shardNode3 | 27021 | Shard a Secondary 2 | 172.31.90.164 |
| Node 7 | shardNode3 | 27022 | Shard b Secondary 2 | 172.31.90.164 |
| Node 7 | shardNode3 | 27023 | Shard c Primary | 172.31.90.164 |