

1. Create 5 EC2 instances by clicking Launch Instance → Create RSA key (midtermKey.pem) → Launch Instance. Set Inbound / Outbound traffic to All traffic.

- a. Mongos:
- b. Configs:
- c. Shard1:
- d. Shard2:
- e. Shard3:

The screenshot shows the AWS EC2 Instances page with the following details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Avg.
mongos	i-02bd86a06ee543123	Running	t2.small	2/2 checks passed	No alarms	us-
configs	i-0f719806994d0d923	Running	t2.small	2/2 checks passed	No alarms	us-
shard1	i-0ae82915aacb93f22	Running	t2.small	2/2 checks passed	No alarms	us-
shard2	i-08c14502eaa1bc33f	Running	t2.small	2/2 checks passed	No alarms	us-
shard3	i-005ce4df31de9df77	Running	t2.small	2/2 checks passed	No alarms	us-

2. Give permissions to the key files created in step 1: chmod 600 midtermKey.pem

Access to servers: ssh -i midtermKey.pem ubuntu@<Public IPv4 address>

The screenshot shows five terminal windows on a desktop environment, each representing one of the five EC2 instances. The commands run in each window are:

```

mongos - Dung Dong x + ~
$ chmod 600 midtermKey.pem

config - Dung Dong x + ~
$ chmod 600 midtermKey.pem

shard1 - Dung Dong x + ~
$ chmod 600 midtermKey.pem

shard2 - Dung Dong x + ~
$ chmod 600 midtermKey.pem

shard3 - Dung Dong x + ~
$ chmod 600 midtermKey.pem

```

3. Follow steps on [Install MongoDB Community Edition on Ubuntu — MongoDB Manual](#) to install mongoDB

However, the first step gets an error saying that “gpg: WARNING: no command supplied. Trying to guess what you mean ...”. I checked the key-list and it signed for MongoDB 5.0 not 6.0.

Thus, I had to replace it with “curl -sS https://pgp.mongodb.com/server-6.0.asc | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/mongodb-6.0.gpg” for the first step.

```
ubuntu@ip-172-31-26-40: ~      + | -      X
Setting up mongodb-org-server (6.0.5) ...
Adding system user 'mongodb' (UID 115) ...
Adding new user 'mongodb' (UID 115) with group 'nogroup' ...
Not creating home directory '/home/mongodb'.
Adding group 'mongodb' (GID 122) ...
Done.
Adding user 'mongodb' to group 'mongodb' ...
Adding user mongodb to group mongodb
Done.
Setting up mongodb-org-shell (6.0.5) ...
Setting up mongodb-database-tools (100.7.0) ...
Setting up mongodb-org-mongos (6.0.5) ...
Setting up mongodb-org-database-tools-extra (6.0.5) ...
Setting up mongodb-org-database (6.0.5) ...
Setting up mongodb-org-tools (6.0.5) ...
Setting up mongodb-org (6.0.5) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

ubuntu@ip-172-31-26-40:~$
```

```
ubuntu@ip-172-31-31-11: ~      + | -      X
Setting up mongodb-org-server (6.0.5) ...
Adding system user 'mongodb' (UID 115) ...
Adding new user 'mongodb' (UID 115) with group 'nogroup' ...
Not creating home directory '/home/mongodb'.
Adding group 'mongodb' (GID 122) ...
Done.
Adding user 'mongodb' to group 'mongodb' ...
Adding user mongodb to group mongodb
Done.
Setting up mongodb-org-shell (6.0.5) ...
Setting up mongodb-database-tools (100.7.0) ...
Setting up mongodb-org-mongos (6.0.5) ...
Setting up mongodb-org-database-tools-extra (6.0.5) ...
Setting up mongodb-org-database (6.0.5) ...
Setting up mongodb-org-tools (6.0.5) ...
Setting up mongodb-org (6.0.5) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.
```

```
ubuntu@ip-172-31-17-52: ~      + | - X  
Setting up mongodb-org-server (6.0.5) ...  
Adding system user 'mongodb' (UID 115) ...  
Adding new user 'mongodb' (UID 115) with group 'nogroup' ...  
Not creating home directory '/home/mongodb'.  
Adding group 'mongodb' (GID 122) ...  
Done.  
Adding user 'mongodb' to group 'mongodb' ...  
Adding user mongodb to group mongodb  
Done.  
Setting up mongodb-org-shell (6.0.5) ...  
Setting up mongodb-database-tools (100.7.0) ...  
Setting up mongodb-org-mongos (6.0.5) ...  
Setting up mongodb-org-database-tools-extra (6.0.5) ...  
Setting up mongodb-org-database (6.0.5) ...  
Setting up mongodb-org-tools (6.0.5) ...  
Setting up mongodb-org (6.0.5) ...  
Processing triggers for man-db (2.10.2-1) ...  
Scanning processes...  
Scanning linux images...  
  
Running kernel seems to be up-to-date.  
  
No services need to be restarted.  
  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-17-52:~$
```

```
ubuntu@ip-172-31-22-123: ~      + | - X  
Setting up mongodb-org-server (6.0.5) ...  
Adding system user 'mongodb' (UID 115) ...  
Adding new user 'mongodb' (UID 115) with group 'nogroup' ...  
Not creating home directory '/home/mongodb'.  
Adding group 'mongodb' (GID 122) ...  
Done.  
Adding user 'mongodb' to group 'mongodb' ...  
Adding user mongodb to group mongodb  
Done.  
Setting up mongodb-org-shell (6.0.5) ...  
Setting up mongodb-database-tools (100.7.0) ...  
Setting up mongodb-org-mongos (6.0.5) ...  
Setting up mongodb-org-database-tools-extra (6.0.5) ...  
Setting up mongodb-org-database (6.0.5) ...  
Setting up mongodb-org-tools (6.0.5) ...  
Setting up mongodb-org (6.0.5) ...  
Processing triggers for man-db (2.10.2-1) ...  
Scanning processes...  
Scanning linux images...  
  
Running kernel seems to be up-to-date.  
  
No services need to be restarted.  
  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.
```

```
ubuntu@ip-172-31-19-219: ~ + | - | X

Setting up mongodb-org-server (6.0.5) ...
Adding system user `mongodb` (UID 115) ...
Adding new user `mongodb` (UID 115) with group `nogroup` ...
Not creating home directory `/home/mongodb'.
Adding group `mongodb` (GID 122) ...
Done.
Adding user `mongodb` to group `mongodb` ...
Adding user mongodb to group mongodb
Done.
Setting up mongodb-org-shell (6.0.5) ...
Setting up mongodb-database-tools (100.7.0) ...
Setting up mongodb-org-mongos (6.0.5) ...
Setting up mongodb-org-database-tools-extra (6.0.5) ...
Setting up mongodb-org-database (6.0.5) ...
Setting up mongodb-org-tools (6.0.5) ...
Setting up mongodb-org (6.0.5) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-19-219:~$
```

4.

- Config: sudo mkdir -p /db/config1 /db/config2 /db/config3

```
ubuntu@ip-172-31-17-52:~$ sudo mkdir -p /db/config1 /db/config2 /db/config3
ubuntu@ip-172-31-17-52:~$ cd /db/
ubuntu@ip-172-31-17-52:/db$ ls
config1 config2 config3
ubuntu@ip-172-31-17-52:/db$
```

- b.** Shard1: mkdir -p db/shard1/mem1 db/shard1/mem2 db/shard1/mem3

```
ubuntu@ip-172-31-19-219:~$ mkdir -p db/shard1/mem1 db/shard1/mem2 db/shard1/mem3
ubuntu@ip-172-31-19-219:~$ ls
db
ubuntu@ip-172-31-19-219:~$ cd db
ubuntu@ip-172-31-19-219:~/db$ ls
shard1
ubuntu@ip-172-31-19-219:~/db$ cd shard1
ubuntu@ip-172-31-19-219:~/db/shard1$ ls
mem1 mem2 mem3
ubuntu@ip-172-31-19-219:~/db/shard1$ █
```

- c.** Shard2: mkdir -p db/shard2/mem1 db/shard2/mem2 db/shard2/mem3

```
ubuntu@ip-172-31-31-11:~$ mkdir -p db/shard2/mem1 db/shard2/mem2 db/shard2/mem3
ubuntu@ip-172-31-31-11:~$ ls
db
ubuntu@ip-172-31-31-11:~$ cd db
ubuntu@ip-172-31-31-11:~/db$ ls
shard2
ubuntu@ip-172-31-31-11:~/db$ cd shard2
ubuntu@ip-172-31-31-11:~/db/shard2$ ls
mem1 mem2 mem3
ubuntu@ip-172-31-31-11:~/db/shard2$ █
```

- d.** Shard3: mkdir -p db/shard3/mem1 db/shard3/mem2 db/shard3/mem3

```
ubuntu@ip-172-31-22-123:~$ mkdir -p db/shard3/mem1 db/shard3/mem2 db/shard3/mem3
ubuntu@ip-172-31-22-123:~$ ls
db
ubuntu@ip-172-31-22-123:~$ cd db
ubuntu@ip-172-31-22-123:~/db$ ls
shard3
ubuntu@ip-172-31-22-123:~/db$ cd shard3
ubuntu@ip-172-31-22-123:~/db/shard3$ ls
mem1 mem2 mem3
ubuntu@ip-172-31-22-123:~/db/shard3$ █
```

5.

- a.** Mongos:

- i. Public IPv4 address: 54.226.173.1
- ii. Private IPv4 address: 172.31.26.40
- iii. DNS: ec2-54-226-173-1.compute-1.amazonaws.com

- b.** Configs:

- i. Public IPv4 address: 3.94.196.162
- ii. Private IPv4 address: 172.31.17.52
- iii. DNS: ec2-3-94-196-162.compute-1.amazonaws.com

- c.** Shard1:

- i. Public IPv4 address: 34.230.29.212
- ii. Private IPv4 address: 172.31.19.219

- iii. DNS: ec2-34-230-29-212.compute-1.amazonaws.com
  - d. Shard2:
    - i. Public IPv4 address: 34.226.213.111
    - ii. Private IPv4 address: 172.31.31.11
    - iii. DNS: ec2-34-226-213-111.compute-1.amazonaws.com
  - e. Shard3:
    - i. Public IPv4 address: 54.226.251.91
    - ii. Private IPv4 address: 172.31.22.123
    - iii. DNS: ec2-54-226-251-91.compute-1.amazonaws.com

6.

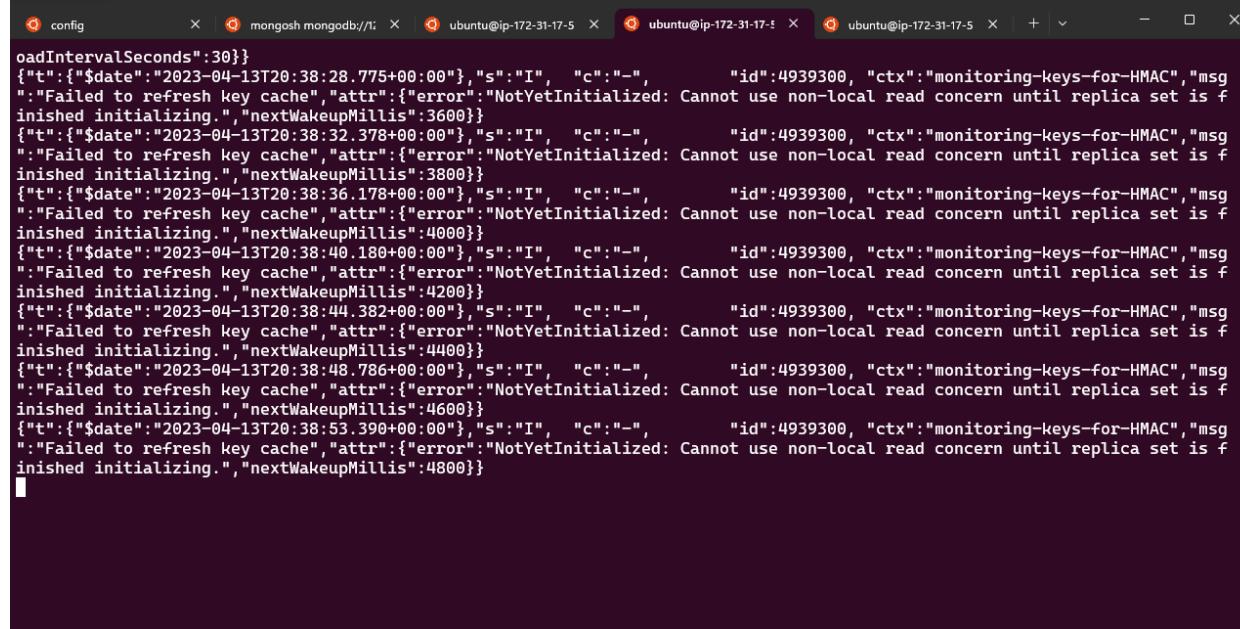
- a. Config 1:**

```
sudo mongod --port 27031 --configsvr --replicaSet config --dbpath /db/config1 --bind_ip localhost,ec2-107-22-129-133.compute-1.amazonaws.com
```

- b. Config 2:**

```
sudo mongod --port 27032 --configsvr --replicaSet config --dbpath /db/config2 --bind_ip
```

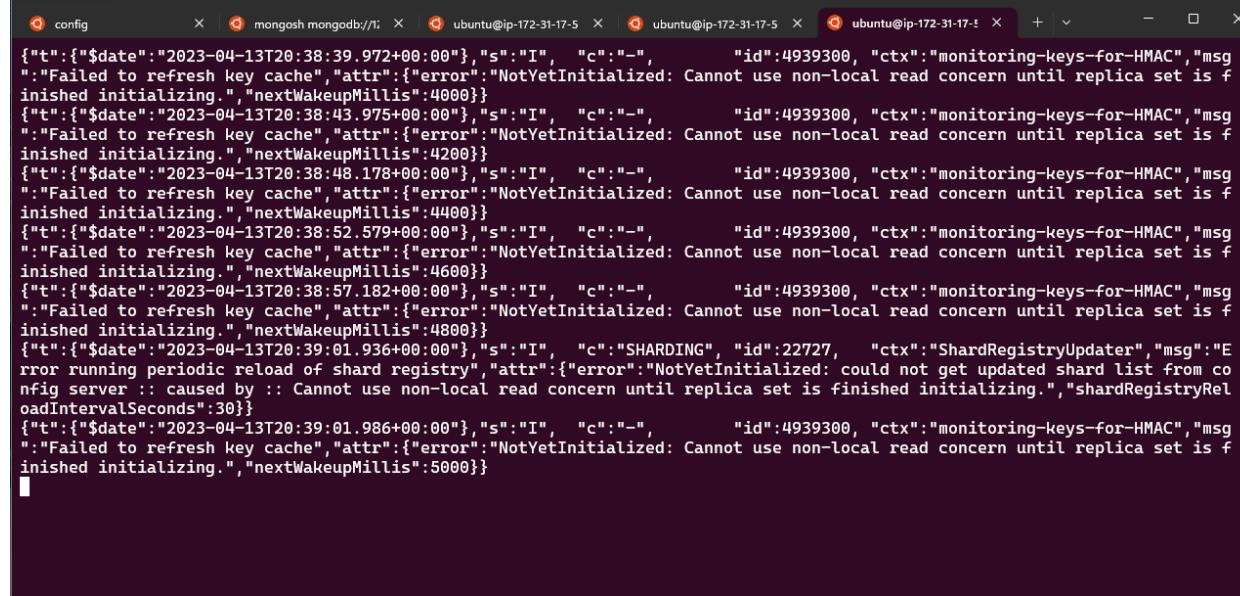
localhost,ec2-107-22-129-133.compute-1.amazonaws.com



```
oadIntervalSeconds":30}]}
{"t":{"$date":"2023-04-13T20:38:28.775+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":3600}}
{"t":{"$date":"2023-04-13T20:38:32.378+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":3800}}
{"t":{"$date":"2023-04-13T20:38:36.178+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4000}}
{"t":{"$date":"2023-04-13T20:38:40.180+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4200}}
{"t":{"$date":"2023-04-13T20:38:44.382+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4400}}
{"t":{"$date":"2023-04-13T20:38:48.786+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4600}}
{"t":{"$date":"2023-04-13T20:38:53.390+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4800}}
```

c. Config 3:

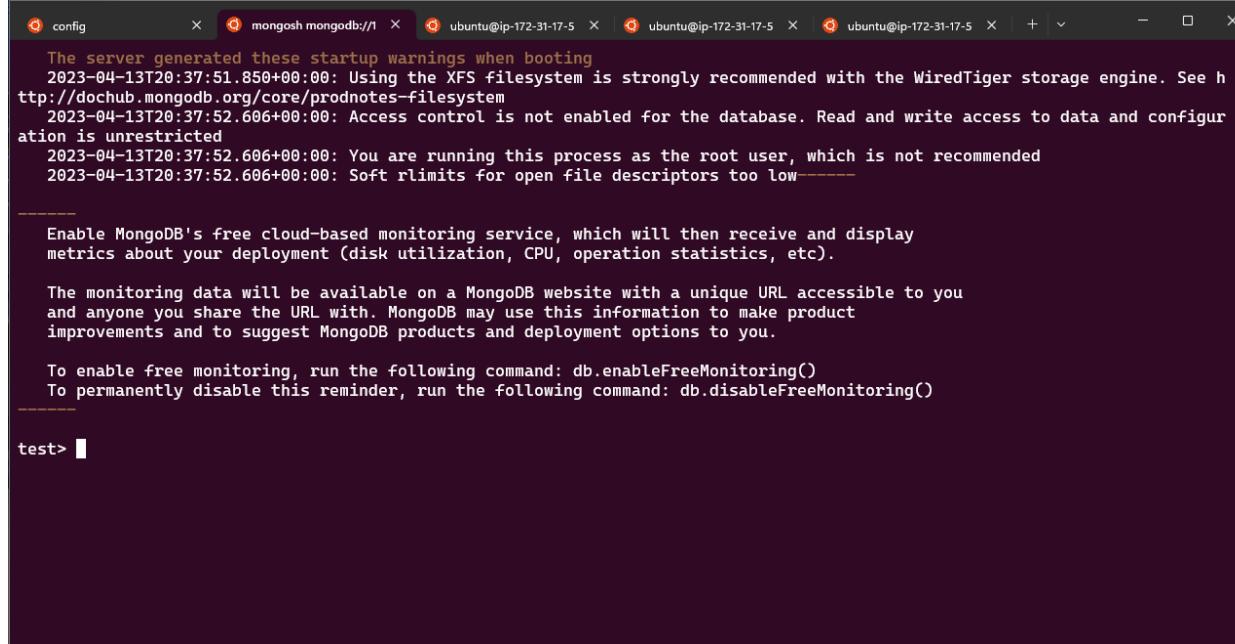
`sudo mongod --port 27033 --configsvr --repSet config --dbpath /db/config3 --bind_ip localhost,ec2-107-22-129-133.compute-1.amazonaws.com`



```
oadIntervalSeconds":30}]}
 {"t":{"$date":"2023-04-13T20:38:39.972+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4000}}
 {"t":{"$date":"2023-04-13T20:38:43.975+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4200}}
 {"t":{"$date":"2023-04-13T20:38:48.178+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4400}}
 {"t":{"$date":"2023-04-13T20:38:52.579+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4600}}
 {"t":{"$date":"2023-04-13T20:38:57.182+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":4800}}
 {"t":{"$date":"2023-04-13T20:39:01.936+00:00"},"s":"I", "c":"SHARDING", "id":22727, "ctx":"ShardRegistryUpdater","msg":"Error running periodic reload of shard registry","attr":{"error":"NotYetInitialized: could not get updated shard list from config server :: caused by :: Cannot use non-local read concern until replica set is finished initializing.", "shardRegistryReloadIntervalSeconds":30}}
 {"t":{"$date":"2023-04-13T20:39:01.986+00:00"},"s":"I", "c":"-", "id":4939300, "ctx":"monitoring-keys-for-HMAC","msg":"Failed to refresh key cache","attr":{"error":"NotYetInitialized: Cannot use non-local read concern until replica set is finished initializing.", "nextWakeupsMillis":5000}}
```

- d. Connect to one of the config servers:

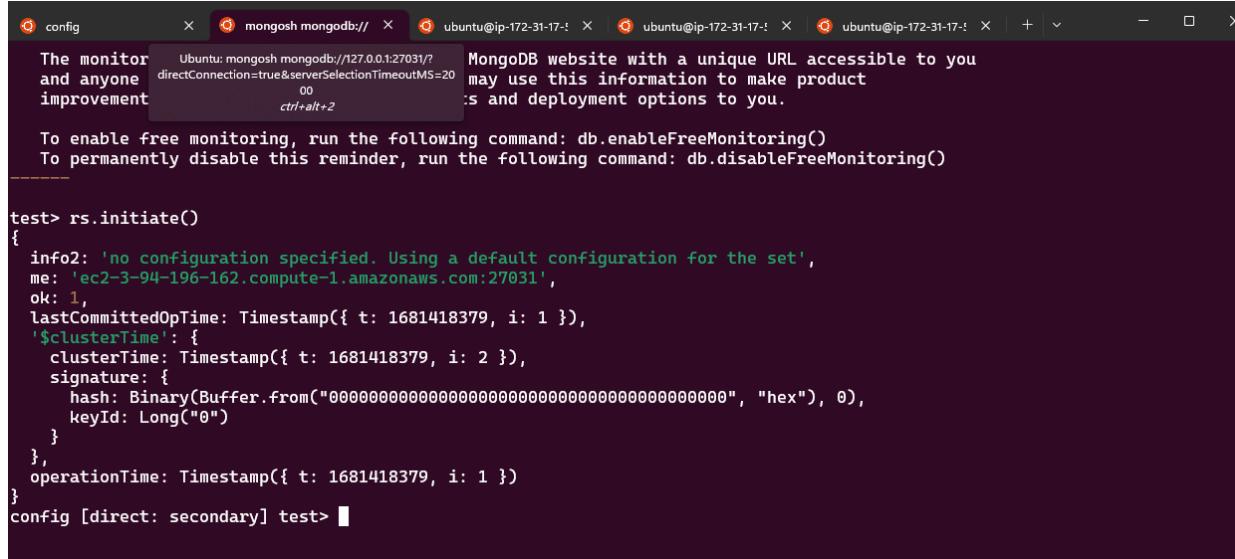
```
mongosh --port 27031
```



The server generated these startup warnings when booting  
2023-04-13T20:37:51.850+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See <http://dochub.mongodb.org/core/prodnotes-filesystem>  
2023-04-13T20:37:52.606+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted  
2023-04-13T20:37:52.606+00:00: You are running this process as the root user, which is not recommended  
2023-04-13T20:37:52.606+00:00: Soft rlimits for open file descriptors too low-----  
-----  
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).  
The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.  
To enable free monitoring, run the following command: db.enableFreeMonitoring()  
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()  
-----  
test> █

- e. Initiate the replica set:

```
rs.initiate()
```



The monitor and anyone improvement Ubuntu: mongosh mongodb://127.0.0.1:27031/? directConnection=true&serverSelectionTimeoutMS=20 MongoDB website with a unique URL accessible to you may use this information to make product improvements and deployment options to you.  
To enable free monitoring, run the following command: db.enableFreeMonitoring()  
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()  
-----  
test> rs.initiate()  
{  
 info2: 'no configuration specified. Using a default configuration for the set',  
 me: 'ec2-3-94-196-162.compute-1.amazonaws.com:27031',  
 ok: 1,  
 lastCommittedOpTime: Timestamp({ t: 1681418379, i: 1 }),  
 '\$clusterTime': {  
 clusterTime: Timestamp({ t: 1681418379, i: 2 }),  
 signature: {  
 hash: Binary(Buffer.from("00000000000000000000000000000000", "hex"), 0),  
 keyId: Long("0")  
 }  
 },  
 operationTime: Timestamp({ t: 1681418379, i: 1 })  
}  
config [direct: secondary] test> █

- f. Check status:

```
rs.status()
```

```
config [direct: Ubuntu: mongosh mongodb://127.0.0.1:27031/? directConnection=true&serverSelectionTimeoutMS=20
{
  set: 'config',
  date: ISODate('2023-04-13T20:40:35.682Z'),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  configsvr: true,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 1,
  writeMajorityCount: 1,
  votingMembersCount: 1,
  writableVotingMembersCount: 1,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1681418435, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-04-13T20:40:35.682Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1681418435, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1681418435, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1681418435, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-04-13T20:40:35.682Z"),
    lastDurableWallTime: ISODate("2023-04-13T20:40:35.682Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1681418379, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate("2023-04-13T20:39:39.404Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681418379, i: 1 }), t: Long("-1") },
    lastSeenOptimeAtElection: { ts: Timestamp({ t: 1681418379, i: 1 }), t: Long("-1") },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    newTermStartDate: ISODate("2023-04-13T20:39:39.458Z"),
    wMajorityWriteAvailabilityDate: ISODate("2023-04-13T20:39:39.664Z")
  },
  members: [
    {
      _id: 0,
      name: 'ec2-3-94-196-162.compute-1.amazonaws.com:27031',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 165,
      optime: { ts: Timestamp({ t: 1681418435, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-04-13T20:40:35.000Z"),
      lastAppliedWallTime: ISODate("2023-04-13T20:40:35.682Z"),
      lastDurableWallTime: ISODate("2023-04-13T20:40:35.682Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1681418379, i: 2 }),
      electionDate: ISODate("2023-04-13T20:39:39.000Z"),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ],
  ok: 1,
  lastCommittedOpTime: Timestamp({ t: 1681418435, i: 1 }),
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1681418435, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000000000000000000000000000", "hex")
    },
    keyId: Long("0")
  },
  operationTime: Timestamp({ t: 1681418435, i: 1 })
}
config [direct: primary] test> 
```

- g.** Add other members to the set and check for status again:

```
rs.add("ec2-3-86-247-48.compute-1.amazonaws.com:27032")
```

```
rs.add("ec2-3-86-247-48.compute-1.amazonaws.com:27033")
```

```
rs.status()
```

```
config X mongosh mongodb:// X ubuntu@ip-172-31-17-: X ubuntu@ip-172-31-17-: X ubuntu@ip-172-31-17-: X + v

        operationTime   Ubuntu: mongosh mongodb://127.0.0.1:27031/? directConnection=true&serverSelectionTimeoutMS=20
    }
    config [direct:
    {
        set: 'config',
        date: ISODate("2023-04-13T20:41:58.914Z"),
        myState: 1,
        term: Long("1"),
        syncSourceHost: '',
        syncSourceId: -1,
        configsvr: true,
        heartbeatIntervalMillis: Long("2000"),
        majorityVoteCount: 2,
        writeMajorityCount: 2,
        votingMembersCount: 3,
        writableVotingMembersCount: 3,
        optimes: {
            lastCommittedOpTime: { ts: Timestamp({ t: 1681418518, i: 1 }), t: Long("1") },
            lastCommittedWallTime: ISODate("2023-04-13T20:41:58.706Z"),
            readConcernMajorityOpTime: { ts: Timestamp({ t: 1681418518, i: 1 }), t: Long("1") },
            appliedOpTime: { ts: Timestamp({ t: 1681418518, i: 1 }), t: Long("1") },
            durableOpTime: { ts: Timestamp({ t: 1681418518, i: 1 }), t: Long("1") },
            lastAppliedWallTime: ISODate("2023-04-13T20:41:58.706Z"),
            lastDurableWallTime: ISODate("2023-04-13T20:41:58.706Z")
        },
        lastStableRecoveryTimestamp: Timestamp({ t: 1681418497, i: 2 }),
        electionCandidateMetrics: {
            lastElectionReason: 'electionTimeout',
            lastElectionDate: ISODate("2023-04-13T20:39:39.404Z"),
            electionTerm: Long("1"),
            lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681418379, i: 1 }), t: Long("-1") },
            lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1681418379, i: 1 }), t: Long("-1") },
            numVotesNeeded: 1,
            priorityAtElection: 1,
            electionTimeoutMillis: Long("10000"),
            newTermStartDate: ISODate("2023-04-13T20:39:39.458Z"),
            wMajorityWriteAvailabilityDate: ISODate("2023-04-13T20:39:39.664Z")
        },
        members: [
            {
                _id: 0,
                name: 'ec2-3-94-196-162.compute-1.amazonaws.com:27031',
                health: 1,
                state: 1,
                stateStr: 'PRIMARY',
                uptime: 247,
                optime: { ts: Timestamp({ t: 1681418518, i: 1 }), t: Long("1") },
                optimeDate: ISODate("2023-04-13T20:41:58.000Z"),
                lastAppliedWallTime: ISODate("2023-04-13T20:41:58.706Z"),
                lastDurableWallTime: ISODate("2023-04-13T20:41:58.706Z"),
                syncSourceHost: '',
                syncSourceId: -1,
                infoMessage: '',
                electionTime: Timestamp({ t: 1681418379, i: 2 }),
                electionDate: ISODate("2023-04-13T20:39:39.000Z"),
                configVersion: 5,
                configTerm: 1,
                self: true,
                lastHeartbeatMessage: ''
            }
        ]
    }
}
```

The screenshot shows a terminal window with multiple tabs. The active tab is titled 'mongosh mongodb://'. It displays the output of a command to test the connection to a MongoDB configuration server. The output is a JSON object representing the configuration state:

```

{
  "self": {
    "lastHeartbeat": "2023-04-13T20:41:56.000Z",
    "ok": 1,
    "name": "Ubuntu: mongosh mongodb://127.0.0.1:27031/?directConnection=true&serverSelectionTimeoutMS=2000",
    "state": 2,
    "stateStr": "SECONDARY",
    "uptime": 24,
    "optime": { "ts": Timestamp({ t: 1681418516, i: 1 }), "t": Long("1") },
    "optimeDurable": { "ts": Timestamp({ t: 1681418516, i: 1 }), "t": Long("1") },
    "optimeDate": ISODate("2023-04-13T20:41:56.000Z"),
    "optimeDurableDate": ISODate("2023-04-13T20:41:56.000Z"),
    "lastAppliedWallTime": ISODate("2023-04-13T20:41:58.706Z"),
    "lastDurableWallTime": ISODate("2023-04-13T20:41:58.706Z"),
    "lastHeartbeat": ISODate("2023-04-13T20:41:57.243Z"),
    "lastHeartbeatRecv": ISODate("2023-04-13T20:41:57.251Z"),
    "pingMs": Long("0"),
    "lastHeartbeatMessage": '',
    "syncSourceHost": "ec2-3-94-196-162.compute-1.amazonaws.com:27031",
    "syncSourceId": 0,
    "infoMessage": '',
    "configVersion": 5,
    "configTerm": 1
  },
  {
    "_id": 2,
    "name": "ec2-3-94-196-162.compute-1.amazonaws.com:27033",
    "health": 1,
    "state": 2,
    "stateStr": "SECONDARY",
    "uptime": 17,
    "optime": { "ts": Timestamp({ t: 1681418516, i: 1 }), "t": Long("1") },
    "optimeDurable": { "ts": Timestamp({ t: 1681418516, i: 1 }), "t": Long("1") },
    "optimeDate": ISODate("2023-04-13T20:41:56.000Z"),
    "optimeDurableDate": ISODate("2023-04-13T20:41:56.000Z"),
    "lastAppliedWallTime": ISODate("2023-04-13T20:41:58.706Z"),
    "lastDurableWallTime": ISODate("2023-04-13T20:41:58.706Z"),
    "lastHeartbeat": ISODate("2023-04-13T20:41:57.242Z"),
    "lastHeartbeatRecv": ISODate("2023-04-13T20:41:58.255Z"),
    "pingMs": Long("0"),
    "lastHeartbeatMessage": '',
    "syncSourceHost": "ec2-3-94-196-162.compute-1.amazonaws.com:27032",
    "syncSourceId": 1,
    "infoMessage": '',
    "configVersion": 5,
    "configTerm": 1
  }
],
"ok": 1,
"lastCommittedOpTime": Timestamp({ t: 1681418518, i: 1 }),
"$clusterTime": {
  "clusterTime": Timestamp({ t: 1681418518, i: 1 }),
  "signature": {
    "hash": Binary(Buffer.from("0000000000000000000000000000000000000000000000000000000000000000", "hex"), 0),
    "keyId": Long("0")
  }
},
"operationTime": Timestamp({ t: 1681418518, i: 1 })
}
config [direct: primary] test>

```

## 7. Start mongos & connect to config servers from mongos:

`sudo mongos --configdb`

`config/ec2-3-86-247-48.compute-1.amazonaws.com:27031,ec2-3-86-247-48.compute-1.amazonaws.com:27032,ec2-3-86-247-48.compute-1.amazonaws.com:27033 --bind_ip`

localhost,ec2-54-152-90-237.compute-1.amazonaws.com

```
mongos      x  ubuntu@ip-172-31-26-40: ~  +  -  X
{
  "t": {"$date": "2023-04-13T20:44:01.155+00:00"}, "s": "I", "c": "HEALTH", "id": 5
  "msg": "No active health observers are configured."}
  "t": {"$date": "2023-04-13T20:44:01.155+00:00"}, "s": "I", "c": "HEALTH", "id": 5
  "msg": "The fault manager initial health checks have completed", "attr": {"state": "Ok"}}
  "t": {"$date": "2023-04-13T20:44:01.155+00:00"}, "s": "I", "c": "HEALTH", "id": 5
  "msg": "Fault manager changed state ", "attr": {"state": "Ok"}}
  "t": {"$date": "2023-04-13T20:44:01.157+00:00"}, "s": "I", "c": "NETWORK", "id": 2
  "ctx": "listener", "msg": "Listening on", "attr": {"address": "/tmp/mongodb-2
  7017.sock"}}
  "t": {"$date": "2023-04-13T20:44:01.157+00:00"}, "s": "I", "c": "NETWORK", "id": 2
  "ctx": "listener", "msg": "Listening on", "attr": {"address": "127.0.0.1"}}
  "t": {"$date": "2023-04-13T20:44:01.158+00:00"}, "s": "I", "c": "NETWORK", "id": 2
  "ctx": "listener", "msg": "Listening on", "attr": {"address": "172.31.26.40"}}
  "t": {"$date": "2023-04-13T20:44:01.158+00:00"}, "s": "I", "c": "NETWORK", "id": 2
  "ctx": "listener", "msg": "Waiting for connections", "attr": {"port": 27017, "ssl": "off"}}
  "t": {"$date": "2023-04-13T20:44:01.163+00:00"}, "s": "I", "c": "SH_REFR", "id": 4
  "ctx": "CatalogCache-0", "msg": "Collection has found to be unsharded afte
  r refresh", "attr": {"namespace": "config.system.sessions", "durationMillis": 4}}
  "t": {"$date": "2023-04-13T20:44:01.163+00:00"}, "s": "I", "c": "CONTROL", "id": 2
  "ctx": "LogicalSessionCacheRefresh", "msg": "Failed to refresh session cac
  he, will try again at the next refresh interval", "attr": {"error": "NamespaceNotS
  harded: Collection config.system.sessions is not sharded."}}
  "t": {"$date": "2023-04-13T20:44:01.163+00:00"}, "s": "I", "c": "CONTROL", "id": 2
  "ctx": "LogicalSessionCacheReap", "msg": "Sessions collection is not set u
  p; waiting until next sessions reap interval", "attr": {"error": "NamespaceNotShar
  ded: Collection config.system.sessions is not sharded."}}
```

- a. Connect to config server:

```
mongosh --host ec2-54-152-90-237.compute-1.amazonaws.com --port 27017
```

The screenshot shows a terminal window with three tabs: "mongos", "ubuntu@ip-172-31-21", and "mongosh mongodb:~". The "mongosh" tab is active and displays the following output:

```
ubuntu@ip-172-31-26-40:~$ mongosh --host ec2-54-226-173-1.compute-1.amazonaws.com --port 27017
Current Mongosh Log ID: 643869d5e6d2a6dd04809b98
Connecting to:      mongodb://ec2-54-226-173-1.compute-1.amazonaws.com:27017/?directConnection=true&appName=mongosh+1.8.0
Using MongoDB:      6.0.5
Using Mongosh:      1.8.0

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

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2023-04-13T20:43:59.119+00:00: Access control is not enabled for the database
e. Read and write access to data and configuration is unrestricted
2023-04-13T20:43:59.119+00:00: You are running this process as the root user
, which is not recommended
-----
[direct: mongos] test> █
```

8.

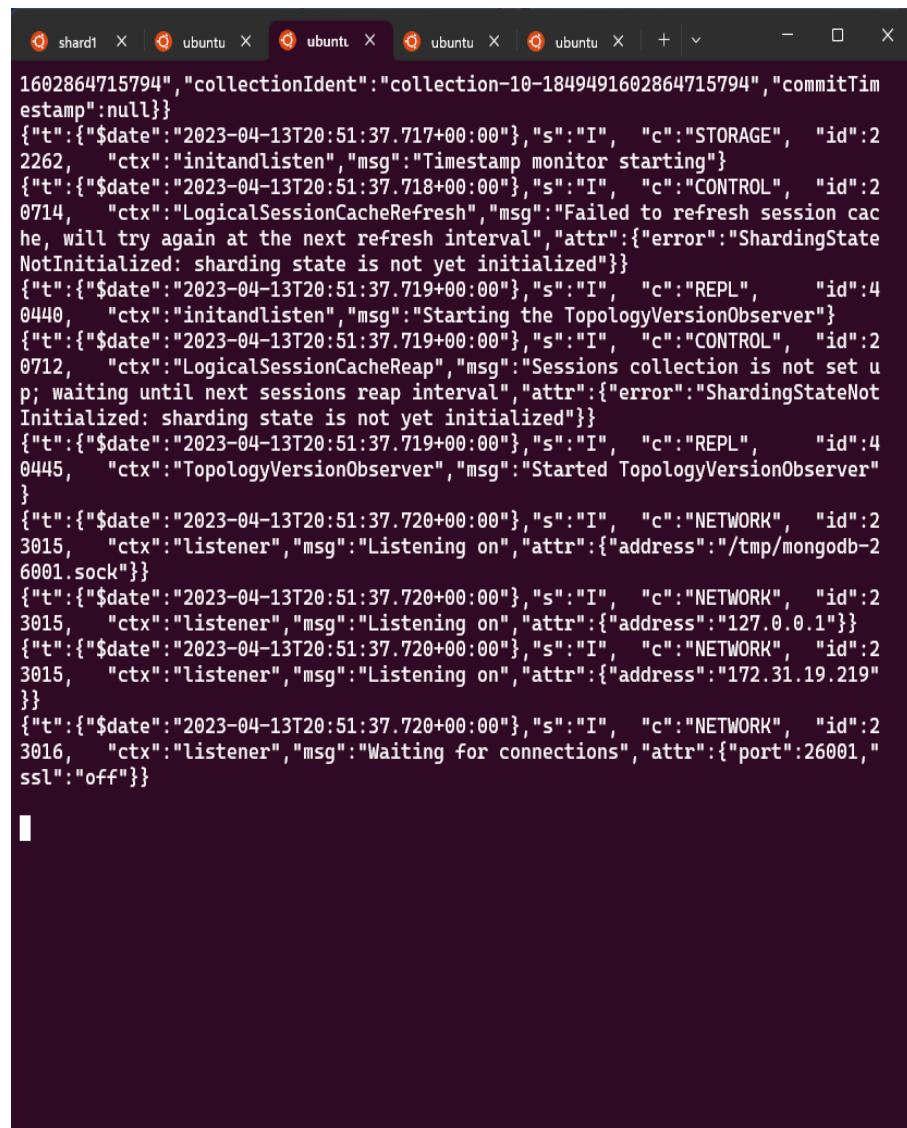
- a. Shard1:

- i. Start shard1 servers:

Shard 1.1:

```
sudo mongod --port 26001 --shardsvr --bind_ip
localhost,ec2-34-230-29-212.compute-1.amazonaws.com --repSet shard1
```

```
--dbpath db/shard1/mem1
```



The screenshot shows a terminal window with several tabs open, but only one tab is active, displaying MongoDB log entries. The log entries are timestamped at 2023-04-13T20:51:37.719+00:00 and detail various startup and monitoring events for a shard. Key messages include:

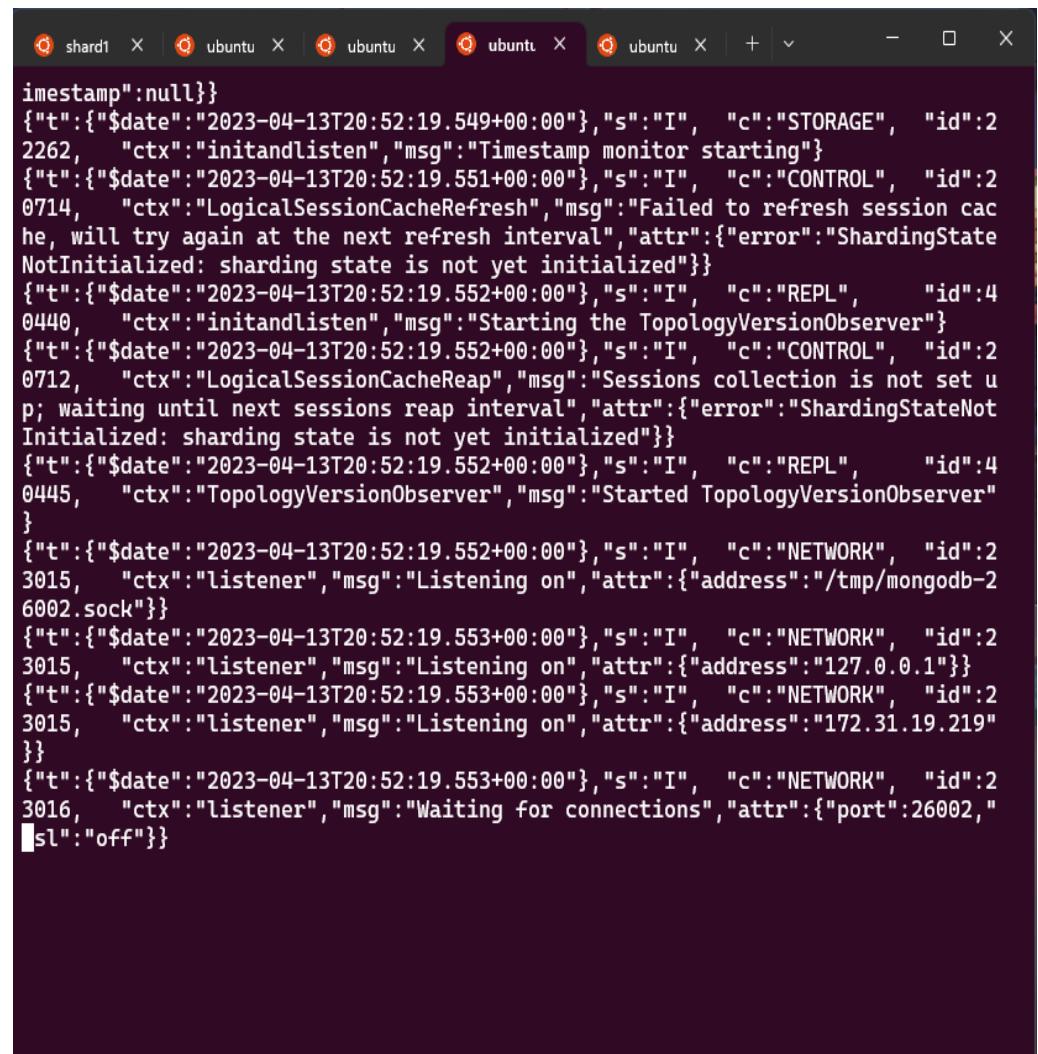
- "Timestamp monitor starting"
- "Failed to refresh session cache, will try again at the next refresh interval"
- "Starting the TopologyVersionObserver"
- "Sessions collection is not set up; waiting until next sessions reap interval"
- "Started TopologyVersionObserver"
- "Listening on" (multiple entries for different ports: 127.0.0.1, 172.31.19.219, and 26001)
- "Waiting for connections"

The terminal window has a dark background and light-colored text. The title bar of the active tab says "shard1".

### Shard 1.2:

```
sudo mongod --port 26002 --shardsvr --bind_ip  
localhost,ec2-34-230-29-212.compute-1.amazonaws.com --repSet shard1
```

```
--dbpath db/shard1/mem2
```



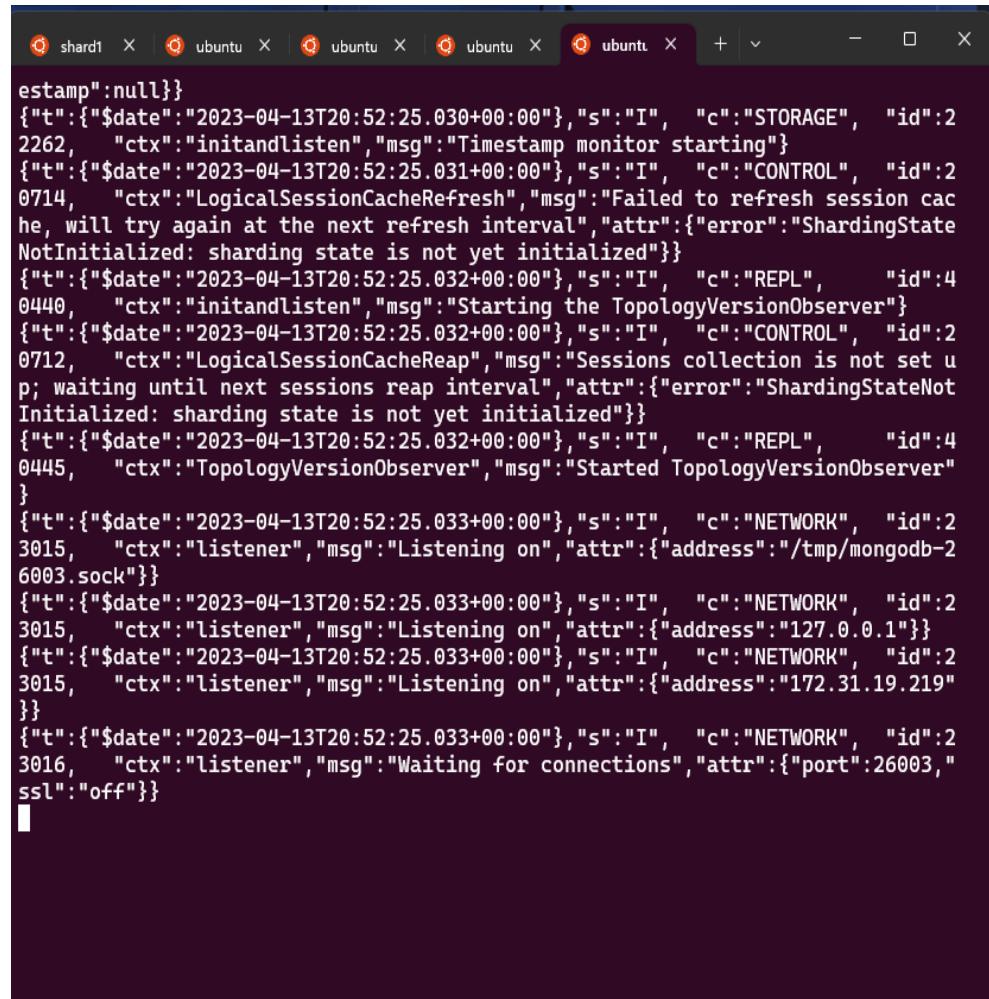
The screenshot shows a terminal window with multiple tabs. The active tab displays MongoDB log entries for shard1. The logs detail various startup and initialization processes, including timestamp monitors, logical session cache refreshes, topology version observers, and network listeners. Key messages include "Timestamp monitor starting", "LogicalSessionCacheRefresh", "Sessions collection is not set up; waiting until next sessions reap interval", and "Started TopologyVersionObserver". The logs also mention "Waiting for connections" on port 26002.

```
imestamp":null}}
{"t":{"$date":"2023-04-13T20:52:19.549+00:00"}, "s":"I", "c":"STORAGE", "id":2262, "ctx":"initandlisten", "msg":"Timestamp monitor starting"}
{"t":{"$date":"2023-04-13T20:52:19.551+00:00"}, "s":"I", "c":"CONTROL", "id":20714, "ctx":"LogicalSessionCacheRefresh", "msg":"Failed to refresh session cache, will try again at the next refresh interval", "attr":{"error":"ShardingStateNotInitialized: sharding state is not yet initialized"}}
{"t":{"$date":"2023-04-13T20:52:19.552+00:00"}, "s":"I", "c":"REPL", "id":40440, "ctx":"initandlisten", "msg":"Starting the TopologyVersionObserver"}
{"t":{"$date":"2023-04-13T20:52:19.552+00:00"}, "s":"I", "c":"CONTROL", "id":20712, "ctx":"LogicalSessionCacheReap", "msg":"Sessions collection is not set up; waiting until next sessions reap interval", "attr":{"error":"ShardingStateNotInitialized: sharding state is not yet initialized"}}
{"t":{"$date":"2023-04-13T20:52:19.552+00:00"}, "s":"I", "c":"REPL", "id":40445, "ctx":"TopologyVersionObserver", "msg":"Started TopologyVersionObserver"}
{
{"t":{"$date":"2023-04-13T20:52:19.552+00:00"}, "s":"I", "c":"NETWORK", "id":23015, "ctx":"listener", "msg":"Listening on", "attr":{"address":"/tmp/mongodb-26002.sock"}}
{"t":{"$date":"2023-04-13T20:52:19.553+00:00"}, "s":"I", "c":"NETWORK", "id":23015, "ctx":"listener", "msg":"Listening on", "attr":{"address":"127.0.0.1"}}
{"t":{"$date":"2023-04-13T20:52:19.553+00:00"}, "s":"I", "c":"NETWORK", "id":23015, "ctx":"listener", "msg":"Listening on", "attr":{"address":"172.31.19.219"}}
{
{"t":{"$date":"2023-04-13T20:52:19.553+00:00"}, "s":"I", "c":"NETWORK", "id":23016, "ctx":"listener", "msg":"Waiting for connections", "attr":{"port":26002, "ssl": "off"}}
```

### Shard 1.3:

```
sudo mongod --port 26003 --shardsvr --bind_ip
localhost,ec2-34-230-29-212.compute-1.amazonaws.com --replicaset shard1
```

```
--dbpath db/shard1/mem3
```



The screenshot shows a terminal window with several tabs open, but only one tab is active, displaying MongoDB log messages. The log output is as follows:

```
estamp":null}}
{"t":{"$date":"2023-04-13T20:52:25.030+00:00"}, "s":"I", "c":"STORAGE", "id":2262, "ctx":"initandlisten", "msg":"Timestamp monitor starting"}
{"t":{"$date":"2023-04-13T20:52:25.031+00:00"}, "s":"I", "c":"CONTROL", "id":20714, "ctx":"LogicalSessionCacheRefresh", "msg":"Failed to refresh session cache, will try again at the next refresh interval", "attr":{"error":"ShardingStateNotInitialized: sharding state is not yet initialized"}}
{"t":{"$date":"2023-04-13T20:52:25.032+00:00"}, "s":"I", "c":"REPL", "id":40440, "ctx":"initandlisten", "msg":"Starting the TopologyVersionObserver"}
{"t":{"$date":"2023-04-13T20:52:25.032+00:00"}, "s":"I", "c":"CONTROL", "id":20712, "ctx":"LogicalSessionCacheReap", "msg":"Sessions collection is not set up; waiting until next sessions reap interval", "attr":{"error":"ShardingStateNotInitialized: sharding state is not yet initialized"}}
{"t":{"$date":"2023-04-13T20:52:25.032+00:00"}, "s":"I", "c":"REPL", "id":40445, "ctx":"TopologyVersionObserver", "msg":"Started TopologyVersionObserver"
}
{"t":{"$date":"2023-04-13T20:52:25.033+00:00"}, "s":"I", "c":"NETWORK", "id":23015, "ctx":"listener", "msg":"Listening on", "attr":{"address":"/tmp/mongodb-26003.sock"}}
{"t":{"$date":"2023-04-13T20:52:25.033+00:00"}, "s":"I", "c":"NETWORK", "id":23015, "ctx":"listener", "msg":"Listening on", "attr":{"address":"127.0.0.1"}}
 {"t":{"$date":"2023-04-13T20:52:25.033+00:00"}, "s":"I", "c":"NETWORK", "id":23015, "ctx":"listener", "msg":"Listening on", "attr":{"address":"172.31.19.219"}}
 {"t":{"$date":"2023-04-13T20:52:25.033+00:00"}, "s":"I", "c":"NETWORK", "id":23016, "ctx":"listener", "msg":"Waiting for connections", "attr":{"port":26003, "ssl":"off"}}
|
```

ii. **Connect to one of those servers and set up replica set:**

```
mongosh --port 26001
rs.initiate()
rs.status()
rs.add("ec2-34-230-29-212.compute-1.amazonaws.com:26002")
rs.add("ec2-34-230-29-212.compute-1.amazonaws.com:26003")

rs.status()
```

```
shard1 [direct: primary] test> rs.status()
{
  set: 'shard1',
  date: ISODate("2023-04-13T21:04:14.501Z"),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-04-13T21:04:11.319Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-04-13T21:04:11.319Z"),
    lastDurableWallTime: ISODate("2023-04-13T21:04:11.319Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1681419826, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate("2023-04-13T21:03:46.636Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681419826, i: 1 }), t: Long("-1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1681419826, i: 1 }), t: Long("-1") },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    newTermStartDate: ISODate("2023-04-13T21:03:46.706Z"),
    wMajorityWriteAvailabilityDate: ISODate("2023-04-13T21:03:46.828Z")
  },
  members: [
    {
      _id: 0,
      name: 'ec2-34-230-29-212.compute-1.amazonaws.com:26001',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 758,
      optime: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-04-13T21:04:11.000Z"),
      lastAppliedWallTime: ISODate("2023-04-13T21:04:11.319Z"),
      lastDurableWallTime: ISODate("2023-04-13T21:04:11.319Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1681419826, i: 2 }),
      electionDate: ISODate("2023-04-13T21:03:46.000Z"),
      configVersion: 5,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ]
}
```

```
shard1 X mongc X ubuntu X | ubuntu X | ubuntu X | + | - | □ | ×
    electionDate: ISODate("2023-04-13T21:03:46.000Z"),
    configVersion: 5,
    configTerm: 1,
    self: true,
    lastHeartbeatMessage: ''
},
{
    _id: 1,
    name: 'ec2-34-230-29-212.compute-1.amazonaws.com:26002',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 11,
    optime: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
    optimeDurable: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
    optimeDate: ISODate("2023-04-13T21:04:11.000Z"),
    optimeDurableDate: ISODate("2023-04-13T21:04:11.000Z"),
    lastAppliedWallTime: ISODate("2023-04-13T21:04:11.319Z"),
    lastDurableWallTime: ISODate("2023-04-13T21:04:11.319Z"),
    lastHeartbeat: ISODate("2023-04-13T21:04:13.331Z"),
    lastHeartbeatRecv: ISODate("2023-04-13T21:04:13.340Z"),
    pingMs: Long("0"),
    lastHeartbeatMessage: '',
    syncSourceHost: 'ec2-34-230-29-212.compute-1.amazonaws.com:26001',
    syncSourceId: 0,
    infoMessage: '',
    configVersion: 5,
    configTerm: 1
},
{
    _id: 2,
    name: 'ec2-34-230-29-212.compute-1.amazonaws.com:26003',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 5,
    optime: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
    optimeDurable: { ts: Timestamp({ t: 1681419851, i: 1 }), t: Long("1") },
    optimeDate: ISODate("2023-04-13T21:04:11.000Z"),
    optimeDurableDate: ISODate("2023-04-13T21:04:11.000Z"),
    lastAppliedWallTime: ISODate("2023-04-13T21:04:11.319Z"),
    lastDurableWallTime: ISODate("2023-04-13T21:04:11.319Z"),
    lastHeartbeat: ISODate("2023-04-13T21:04:13.331Z"),
    lastHeartbeatRecv: ISODate("2023-04-13T21:04:14.344Z"),
    pingMs: Long("2"),
    lastHeartbeatMessage: '',
    syncSourceHost: 'ec2-34-230-29-212.compute-1.amazonaws.com:26002',
    syncSourceId: 1,
    infoMessage: '',
    configVersion: 5,
    configTerm: 1
},
],
ok: 1,
'$clusterTime': {
    clusterTime: Timestamp({ t: 1681419851, i: 1 }),
    signature: {
        hash: Binary(Buffer.from("0000000000000000000000000000000000000000000000000000000000000000", "hex")
    },
    keyId: Long("0")
},
operationTime: Timestamp({ t: 1681419851, i: 1 })
}
shard1 [direct: primary] test>
```

**b. Shard 2:**

```
sudo mongod --port 26101 --shardsvr --bind_ip  
localhost,ec2-34-226-213-111.compute-1.amazonaws.com --replicaSet  
shard2 --dbpath db/shard2/mem1
```

```
sudo mongod --port 26102 --shardsvr --bind_ip  
localhost,ec2-34-226-213-111.compute-1.amazonaws.com --replicaSet  
shard2 --dbpath db/shard2/mem2
```

```
sudo mongod --port 26103 --shardsvr --bind_ip  
localhost,ec2-34-226-213-111.compute-1.amazonaws.com --replicaSet  
shard2 --dbpath db/shard2/mem3
```

```
mongosh --port 26101  
rs.initiate()  
rs.status()  
rs.add("ec2-34-226-213-111.compute-1.amazonaws.com:26102")  
rs.add("ec2-34-226-213-111.compute-1.amazonaws.com:26103")  
rs.status()
```

```
shard2 [direct: pr] s()
{
  set: 'shard2',
  date: ISODate("2023-04-13T21:11:37.822Z"),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-04-13T21:11:34.430Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-04-13T21:11:34.430Z"),
    lastDurableWallTime: ISODate("2023-04-13T21:11:34.430Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1681420262, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate("2023-04-13T21:11:02.702Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681420262, i: 1 }), t: Long("-1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1681420262, i: 1 }), t: Long("-1") },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    newTermStartDate: ISODate("2023-04-13T21:11:02.790Z"),
    wMajorityWriteAvailabilityDate: ISODate("2023-04-13T21:11:02.843Z")
  },
  members: [
    {
      _id: 0,
      name: 'ec2-34-226-213-111.compute-1.amazonaws.com:26101',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 63,
      optime: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-04-13T21:11:34.000Z"),
      lastAppliedWallTime: ISODate("2023-04-13T21:11:34.430Z"),
      lastDurableWallTime: ISODate("2023-04-13T21:11:34.430Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1681420262, i: 2 }),
      electionDate: ISODate("2023-04-13T21:11:02.000Z"),
      configVersion: 5,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ]
}
```

```
shard2 X mongc X ubuntu X ubuntu X ubuntu X + v - □ ×
electionDate: "Ubuntu:ubuntu@ip-172-31-31-11:~$T21:11:02.000Z"),
configVersion: 5,
configTerm: 1,
self: true,
lastHeartbeatMessage: '',
},
{
_id: 1,
name: 'ec2-34-226-213-111.compute-1.amazonaws.com:26102',
health: 1,
state: 2,
stateStr: 'SECONDARY',
uptime: 9,
optime: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
optimeDurable: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
optimeDate: ISODate("2023-04-13T21:11:34.000Z"),
optimeDurableDate: ISODate("2023-04-13T21:11:34.000Z"),
lastAppliedWallTime: ISODate("2023-04-13T21:11:34.430Z"),
lastDurableWallTime: ISODate("2023-04-13T21:11:34.430Z"),
lastHeartbeat: ISODate("2023-04-13T21:11:36.436Z"),
lastHeartbeatRecv: ISODate("2023-04-13T21:11:36.447Z"),
pingMs: Long("0"),
lastHeartbeatMessage: '',
syncSourceHost: 'ec2-34-226-213-111.compute-1.amazonaws.com:26101',
syncSourceId: 0,
infoMessage: '',
configVersion: 5,
configTerm: 1
},
{
_id: 2,
name: 'ec2-34-226-213-111.compute-1.amazonaws.com:26103',
health: 1,
state: 2,
stateStr: 'SECONDARY',
uptime: 5,
optime: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
optimeDurable: { ts: Timestamp({ t: 1681420294, i: 1 }), t: Long("1") },
optimeDate: ISODate("2023-04-13T21:11:34.000Z"),
optimeDurableDate: ISODate("2023-04-13T21:11:34.000Z"),
lastAppliedWallTime: ISODate("2023-04-13T21:11:34.430Z"),
lastDurableWallTime: ISODate("2023-04-13T21:11:34.430Z"),
lastHeartbeat: ISODate("2023-04-13T21:11:36.438Z"),
lastHeartbeatRecv: ISODate("2023-04-13T21:11:36.948Z"),
pingMs: Long("2"),
lastHeartbeatMessage: '',
syncSourceHost: 'ec2-34-226-213-111.compute-1.amazonaws.com:26102',
syncSourceId: 1,
infoMessage: '',
configVersion: 5,
configTerm: 1
}
],
ok: 1,
'$clusterTime': {
clusterTime: Timestamp({ t: 1681420294, i: 1 }),
signature: {
hash: Binary(Buffer.from("0000000000000000000000000000000000000000000000000000000000000000", "hex")),
keyId: Long("0")
},
operationTime: Timestamp({ t: 1681420294, i: 1 })
}
shard2 [direct: primary] test>
```

c. **Shard 3:**

```
sudo mongod --port 26201 --shardsvr --bind_ip  
localhost,ec2-54-226-251-91.compute-1.amazonaws.com --repSet shard3  
--dbpath db/shard3/mem1
```

```
sudo mongod --port 26202 --shardsvr --bind_ip  
localhost,ec2-54-226-251-91.compute-1.amazonaws.com --repSet shard3  
--dbpath db/shard3/mem2
```

```
sudo mongod --port 26203 --shardsvr --bind_ip  
localhost,ec2-54-226-251-91.compute-1.amazonaws.com --repSet shard3  
--dbpath db/shard3/mem3
```

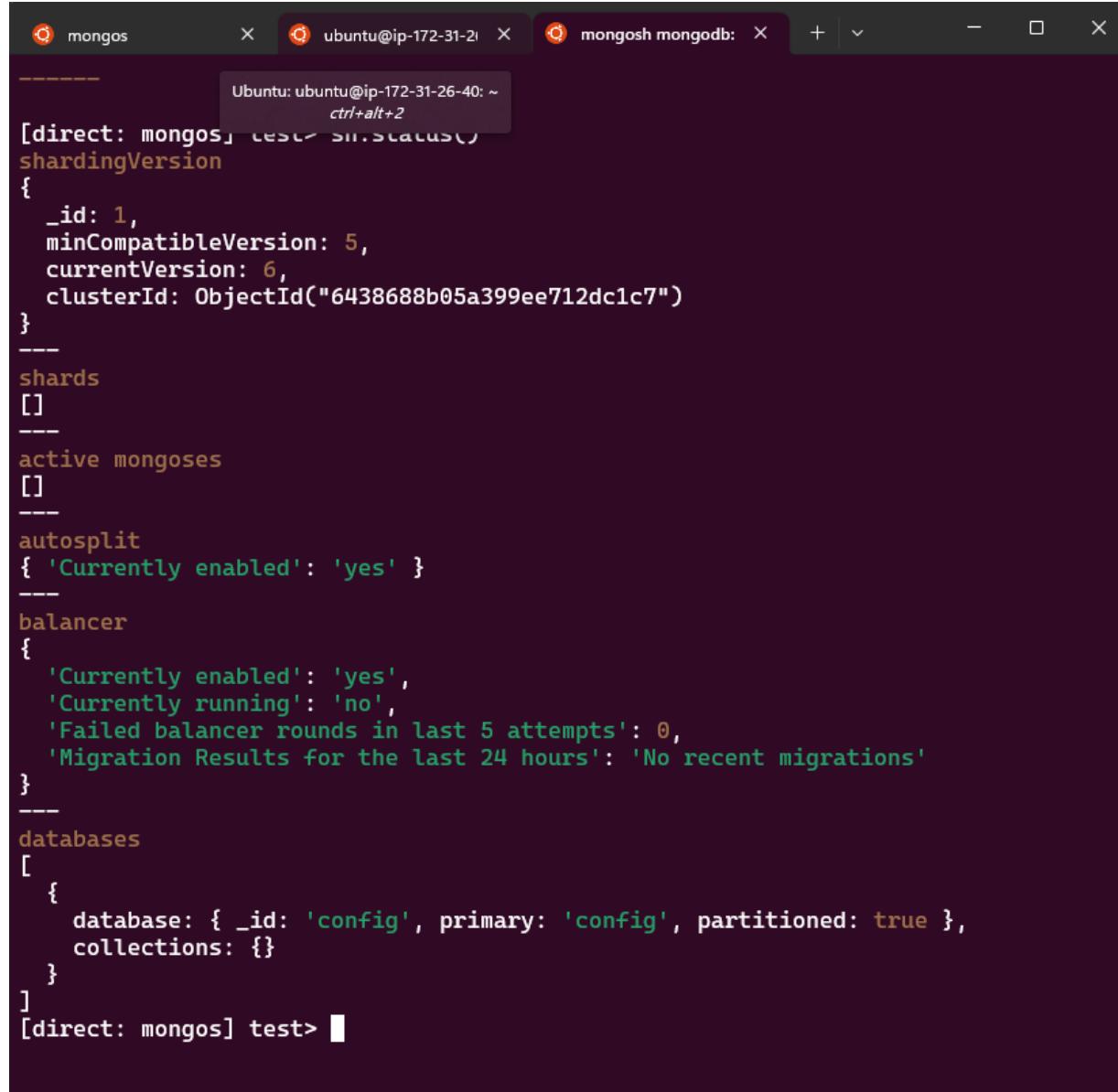
```
mongosh --port 26201  
rs.initiate()  
rs.status()  
rs.add("ec2-54-226-251-91.compute-1.amazonaws.com:26202")  
rs.add("ec2-54-226-251-91.compute-1.amazonaws.com:26203")  
rs.status()
```

```
shard3 [direct: primary] test> rs.status()
{
  set: 'shard3',
  date: ISODate("2023-04-13T21:17:36.334Z"),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-04-13T21:17:33.942Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-04-13T21:17:33.942Z"),
    lastDurableWallTime: ISODate("2023-04-13T21:17:33.942Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1681420619, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectioNDate: ISODate("2023-04-13T21:16:59.775Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681420619, i: 1 }), t: Long("-1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1681420619, i: 1 }), t: Long("-1") },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    newTermStartDate: ISODate("2023-04-13T21:16:59.817Z"),
    wMajorityWriteAvailabilityDate: ISODate("2023-04-13T21:16:59.857Z")
  },
  members: [
    {
      _id: 0,
      name: 'ec2-54-226-251-91.compute-1.amazonaws.com:26201',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 84,
      optime: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-04-13T21:17:33.000Z"),
      lastAppliedWallTime: ISODate("2023-04-13T21:17:33.942Z"),
      lastDurableWallTime: ISODate("2023-04-13T21:17:33.942Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1681420619, i: 2 }),
      electionDate: ISODate("2023-04-13T21:16:59.000Z"),
      configVersion: 5,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ]
}
```

```
shard3  mongc  ubuntu  ubuntu  ubuntu  +  -  ×
}, {
  _id: 1,
  name: 'ec2-54-226-251-91.compute-1.amazonaws.com:26202',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 8,
  optime: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
  optimeDurable: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
  optimeDate: ISODate("2023-04-13T21:17:33.000Z"),
  optimeDurableDate: ISODate("2023-04-13T21:17:33.000Z"),
  lastAppliedWallTime: ISODate("2023-04-13T21:17:33.942Z"),
  lastDurableWallTime: ISODate("2023-04-13T21:17:33.942Z"),
  lastHeartbeat: ISODate("2023-04-13T21:17:35.948Z"),
  lastHeartbeatRecv: ISODate("2023-04-13T21:17:35.964Z"),
  pingMs: Long("0"),
  lastHeartbeatMessage: '',
  syncSourceHost: 'ec2-54-226-251-91.compute-1.amazonaws.com:26201',
  syncSourceId: 0,
  infoMessage: '',
  configVersion: 5,
  configTerm: 1
},
{
  _id: 2,
  name: 'ec2-54-226-251-91.compute-1.amazonaws.com:26203',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 4,
  optime: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
  optimeDurable: { ts: Timestamp({ t: 1681420653, i: 1 }), t: Long("1") },
  optimeDate: ISODate("2023-04-13T21:17:33.000Z"),
  optimeDurableDate: ISODate("2023-04-13T21:17:33.000Z"),
  lastAppliedWallTime: ISODate("2023-04-13T21:17:33.942Z"),
  lastDurableWallTime: ISODate("2023-04-13T21:17:33.942Z"),
  lastHeartbeat: ISODate("2023-04-13T21:17:35.951Z"),
  lastHeartbeatRecv: ISODate("2023-04-13T21:17:34.459Z"),
  pingMs: Long("2"),
  lastHeartbeatMessage: '',
  syncSourceHost: 'ec2-54-226-251-91.compute-1.amazonaws.com:26202',
  syncSourceId: 1,
  infoMessage: '',
  configVersion: 5,
  configTerm: 1
}
],
ok: 1,
'$clusterTime': {
  clusterTime: Timestamp({ t: 1681420653, i: 1 }),
  signature: {
    hash: Binary(Buffer.from("000000000000000000000000000000000000000000000000000000000000000", "hex")
  },
  keyId: Long("0")
},
operationTime: Timestamp({ t: 1681420653, i: 1 })
}
shard3 [direct: primary] test>
```

d. Before adding shard:

```
sh.status()
```



The screenshot shows a terminal window with three tabs: 'mongos', 'ubuntu@ip-172-31-21 ~', and 'mongosh mongodb:'. The 'mongosh' tab is active and displays the output of the 'sh.status()' command. The output is a JSON-like document showing shard information, including shardVersion, shards, active mongoses, and balancer settings. The 'databases' section lists a single database entry for 'config'.

```
Ubuntu: ubuntu@ip-172-31-26-40: ~
ctrl+alt+2
[direct: mongos] test> sh.status()
{
  "shardingVersion": {
    "_id": 1,
    "minCompatibleVersion": 5,
    "currentVersion": 6,
    "clusterId": ObjectId("6438688b05a399ee712dc1c7")
  },
  "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:

Shard 1:

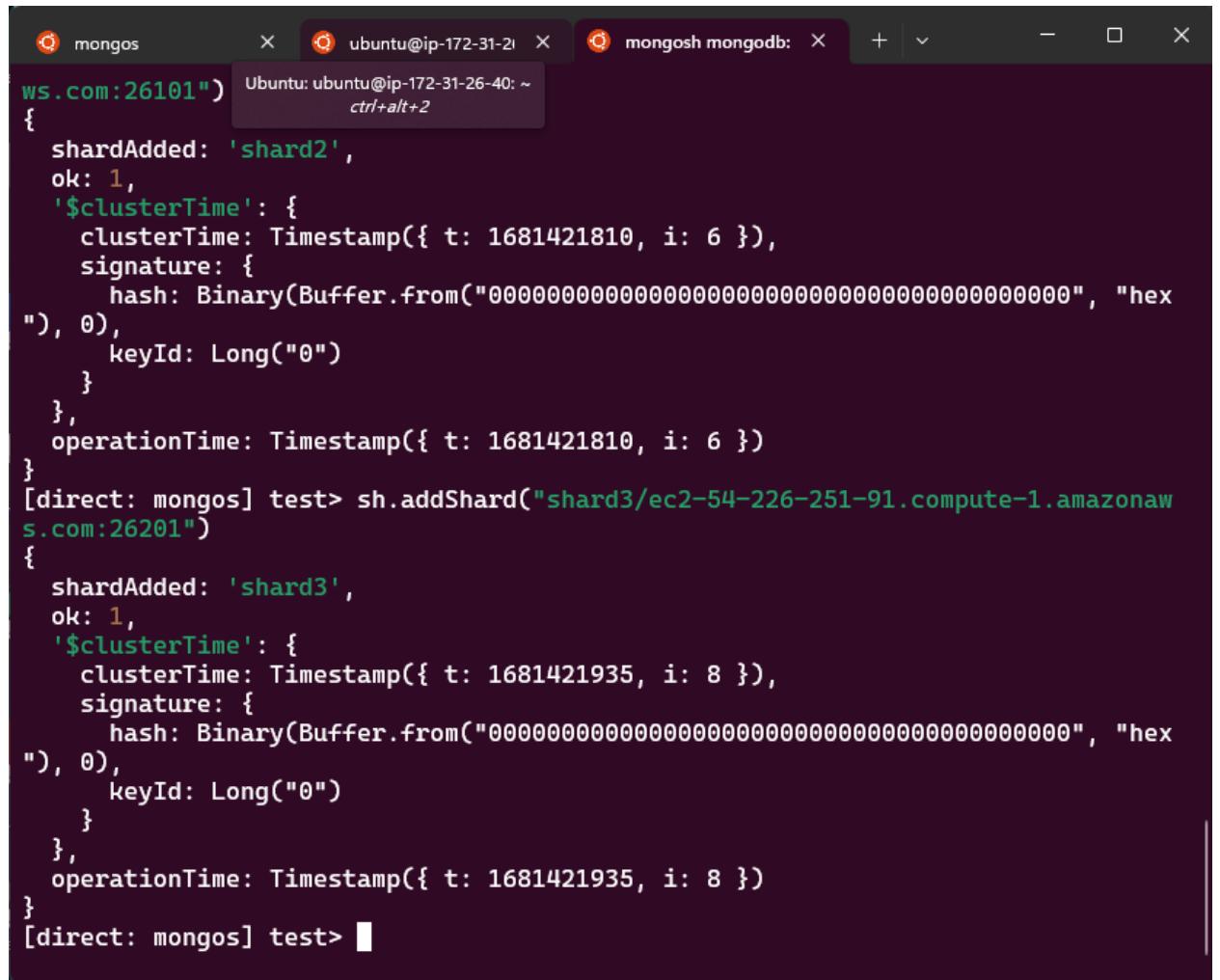
```
sh.addShard("shard1/ec2-34-230-29-212.compute-1.amazonaws.com:26001")
```

## Shard 2:

```
sh.addShard("shard2/ec2-34-226-213-111.compute-1.amazonaws.com:26101")
```

### Shard 3:

```
sh.addShard("shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201")
```



The screenshot shows a terminal window with three tabs: 'mongos', 'ubuntu@ip-172-31-21-2: ~', and 'mongosh mongodb: ~'. The 'mongos' tab contains the command 'sh.addShard("shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201")' and its response, which is a BSON object indicating the shard was added successfully. The 'ubuntu@ip-172-31-21-2: ~' tab shows the prompt 'ctrl+alt+2'. The 'mongosh mongodb: ~' tab shows the command '[direct: mongos] test> sh.addShard("shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201")' and its response, also a BSON object indicating success.

```
ws.com:26101") Ubuntu: ubuntu@ip-172-31-26-40: ~
{
  shardAdded: 'shard2',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1681421810, i: 6 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000000000000000000000000000", "hex"),
      0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1681421810, i: 6 })
}
[direct: mongos] test> sh.addShard("shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201")
{
  shardAdded: 'shard3',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1681421935, i: 8 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000000000000000000000000000", "hex"),
      0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1681421935, i: 8 })
}
[direct: mongos] test> █
```

### Shard status after adding shards :

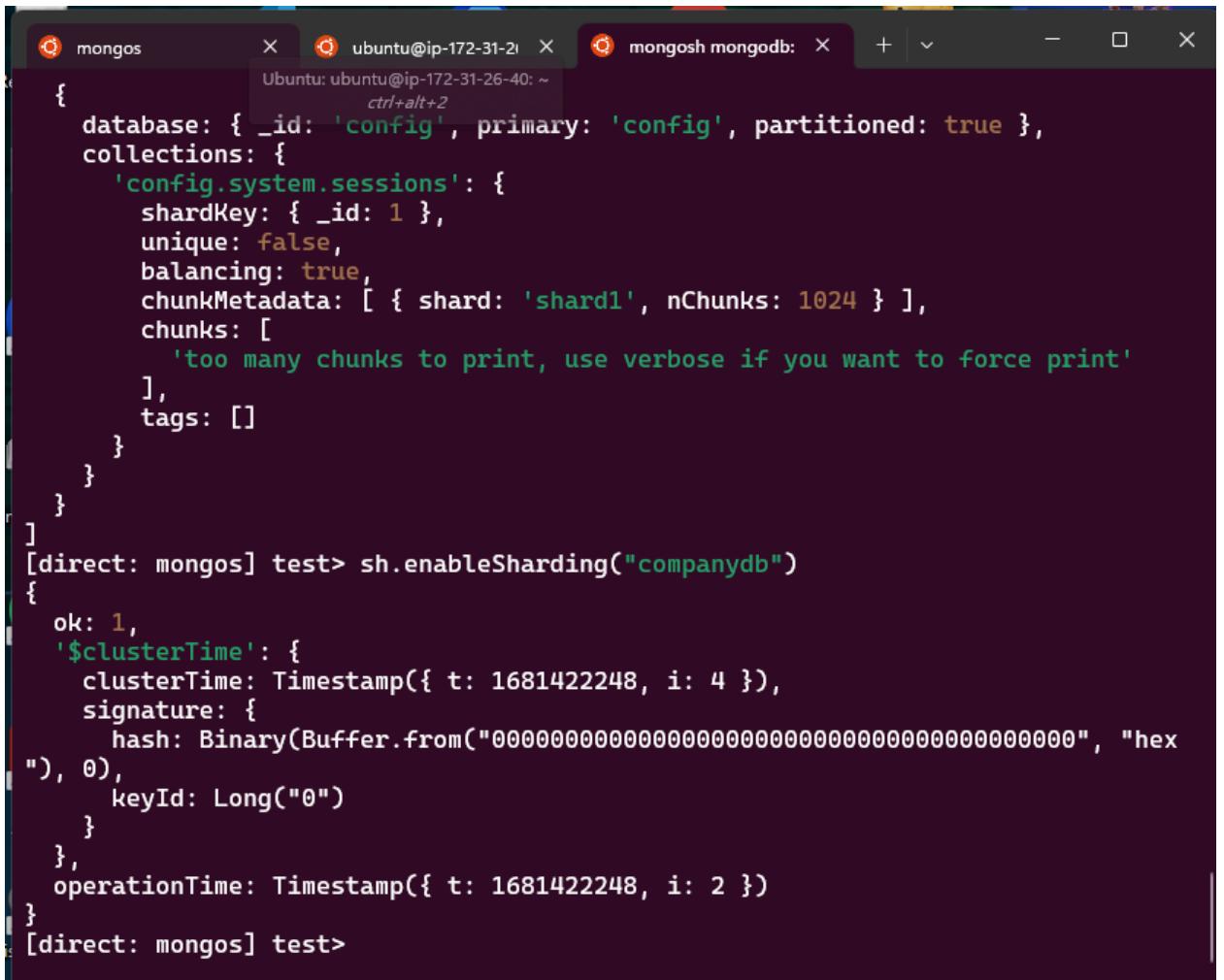
```
sh.status()
```

```
[direct: mongos] Ubuntu: ubuntu@ip-172-31-21: ~
shardingVersion          ctrl+alt+2
{
  _id: 1,
  minCompatibleVersion: 5,
  currentVersion: 6,
  clusterId: ObjectId("6438688b05a399ee712dc1c7")
}
---
shards
[
  {
    _id: 'shard1',
    host: 'shard1/ec2-34-230-29-212.compute-1.amazonaws.com:26001,ec2-34-230-29-212.compute-1.amazonaws.com:26002,ec2-34-230-29-212.compute-1.amazonaws.com:26003',
    state: 1,
    topologyTime: Timestamp({ t: 1681421605, i: 1 })
  },
  {
    _id: 'shard2',
    host: 'shard2/ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26102,ec2-34-226-213-111.compute-1.amazonaws.com:26103',
    state: 1,
    topologyTime: Timestamp({ t: 1681421810, i: 4 })
  },
  {
    _id: 'shard3',
    host: 'shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201,ec2-54-226-251-91.compute-1.amazonaws.com:26202,ec2-54-226-251-91.compute-1.amazonaws.com:26203',
    state: 1,
    topologyTime: Timestamp({ t: 1681421935, i: 6 })
  }
]
---
```

```
mongos      X | ubuntu@ip-172-31-21: ~ X | mongosh mongodb: X + | - | X
--- Ubuntu: ubuntu@ip-172-31-26-40: ~ ctrl+alt+2
active mongoses
[ { '6.0.5': 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: 'config', primary: 'config', partitioned: true },
    collections: {
      'config.system.sessions': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shard1', nChunks: 1024 } ],
        chunks: [
          'too many chunks to print, use verbose if you want to force print'
        ],
        tags: []
      }
    }
  }
]
[direct: mongos] test>
```

## 10. Enable sharding:

```
sh.enableSharding("companydb")
```



The screenshot shows a terminal window with three tabs: "mongos", "ubuntu@ip-172-31-21", and "mongosh mongodb: ~". The "mongosh" tab contains the following MongoDB shell session:

```
{  
    database: { _id: 'config', primary: 'config', partitioned: true },  
    collections: {  
        'config.system.sessions': {  
            shardKey: { _id: 1 },  
            unique: false,  
            balancing: true,  
            chunkMetadata: [ { shard: 'shard1', nChunks: 1024 } ],  
            chunks: [  
                'too many chunks to print, use verbose if you want to force print'  
            ],  
            tags: []  
        }  
    }  
}  
[direct: mongos] test> sh.enableSharding("companydb")  
{  
    ok: 1,  
    '$clusterTime': {  
        clusterTime: Timestamp({ t: 1681422248, i: 4 }),  
        signature: {  
            hash: Binary(Buffer.from("00000000000000000000000000000000", "hex"), 0),  
            keyId: Long("0")  
        },  
        operationTime: Timestamp({ t: 1681422248, i: 2 })  
    }  
}[direct: mongos] test>
```

## Create hashed-index using company's name and enable shard-key:

```
db.company.createIndex({"name": "hashed"})  
sh.shardCollection("companydb.company", {"name": "hashed"})
```

11. Dataset: [mongodb-sample-dataset/companies.json at main · neelabalan/mongodb-sample-dataset · GitHub](https://github.com/neelabalan/mongodb-sample-dataset)

## **Download and Import dataset:**

*wget*

[https://raw.githubusercontent.com/neelabalan/mongodb-sample-dataset/main/sample\\_training/companies.json](https://raw.githubusercontent.com/neelabalan/mongodb-sample-dataset/main/sample_training/companies.json)

```
mongoimport --port 27017 --db companydb --collection company --file companies.json
```

```
mongos      X  ubuntu@ip-  X  mongosh m  X  ubuntu@ip-  X + | - | X
Last login: Ubuntu:ubuntu@ip-172-31-26-40: ~ 2023 from 69.181.90.221
ctrl+alt+2
ubuntu@ip-172-31-26-40:~$ wget https://raw.githubusercontent.com/neelabalan/mongodb-sample-dataset/main/sample_training/companies.json
--2023-04-13 22:05:43-- https://raw.githubusercontent.com/neelabalan/mongodb-sample-dataset/main/sample_training/companies.json
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38563354 (37M) [text/plain]
Saving to: 'companies.json'

companies.json      100%[=====] 36.78M  110MB/s   in 0.3s

2023-04-13 22:05:43 (110 MB/s) - 'companies.json' saved [38563354/38563354]

ubuntu@ip-172-31-26-40:~$ ls
companies.json
ubuntu@ip-172-31-26-40:~$ mongoimport --port 27017 --db companydb --collection company --file companies.json
2023-04-13T22:06:04.477+0000    connected to: mongodb://localhost:27017/
2023-04-13T22:06:07.478+0000    [#####] companydb.company 2
3.0MB/36.8MB (62.6%)
2023-04-13T22:06:09.423+0000    [#####] companydb.company 3
6.8MB/36.8MB (100.0%)
2023-04-13T22:06:09.423+0000    9500 document(s) imported successfully. 0 document(s) failed to import.
ubuntu@ip-172-31-26-40:~$ █
```

## 12.1: Find companies founded between 2010 and 2022

`db.company.find({"founded_year":{$gt: 2010, $lte:2022}})`

Result:

```
mongos  X  ubuntu@ip-172-31-26-40: ~  mongosh m  X  ubuntu@ip-  +  -  □  X
partne Ubuntu: ubuntu@ip-172-31-26-40: ~
{
  _id: ObjectId("52cdef7c4bab8bd675297fb1"),
  name: 'headr',
  permalink: 'headr',
  crunchbase_url: 'http://www.crunchbase.com/company/headr',
  homepage_url: 'http://www.headr.com',
  blog_url: 'http://tripodsocial.tumblr.com/',
  blog_feed_url: '',
  twitter_username: 'tripodsocial',
  category_code: 'web',
  number_of_employees: 8,
  founded_year: 2012,
  founded_month: 1,
  founded_day: null,
  deadpooled_year: null,
  deadpooled_month: null,
  deadpooled_day: null,
  deadpooled_url: '',
  tag_list: '',
  alias_list: null,
  email_address: 'info@headr.com',
  phone_number: '+49-511-336337-0',
  description: 'Free Facebook video chat app.',
  created_at: 'Sun Sep 02 22:13:39 UTC 2007',
  updated_at: 'Tue Jan 22 06:31:44 UTC 2013',
  overview: '<p><a href="http://vive.me/" title="vive" rel="nofollow">vive</a> lets you video chat with your friends and meet amazing people on Facebook! Sh  
are your awesome, ask for advice, laugh and learn, be inspired!</p>',
  image: {
    available_sizes: [
      [
        [ 147, 73 ],
        'assets/images/resized/0019/3143/193143v1-max-150x150.png'
      ],
      [
        [ 147, 73 ],
        'assets/images/resized/0019/3143/193143v1-max-250x250.png'
      ],
      [
        [ 147, 73 ],
        'assets/images/resized/0019/3143/193143v1-max-450x450.png'
      ]
    ],
    attribution: null
  },
  products: [],
  relationships: [
    {
      is_past: false,
      title: 'CEO',
      person: {
        first_name: 'Arnd',
        last_name: 'Aschentrup',
        permalink: 'arnd-aschentrup'
      }
    },
    {
      is_past: false,
      title: 'CPO',
      person: {
        first_name: 'Matthias',
        last_name: 'Kleimann',
        permalink: 'matthias-kleimann'
      }
    }
  ]
}
```

```

mongos   X  ubuntu@ip-  X  mongosh m  X  ubuntu@ip-  X  +  -  □  ×
Ubuntu:ubuntu@ip-172-31-26-40: ~  12-14', 2
[ctrl+alt+2
[direct: mongos] CompanyUpdate db.company.find({$gt: 2010, $lte:2022}).limit(2)
[
{
  _id: ObjectId("52cdef7c4bab8bd675297f78"),
  name: 'CircleUp',
  permalink: 'circleup',
  crunchbase_url: 'http://www.crunchbase.com/company/circleup',
  homepage_url: 'http://circleup.com',
  blog_url: 'https://circleup.com/blog/',
  blog_feed_url: '',
  twitter_username: 'CircleUp',
  category_code: 'finance',
  number_of_employees: 11,
  founded_year: 2011,
  founded_month: 10,
  founded_day: 1,
  deadpooled_year: null,
  deadpooled_month: null,
  deadpooled_day: null,
  deadpooled_url: null,
  tag_list: 'crowdfunding, angel-investing, private-equity, venture-capital, marketplace',
  alias_list: null,
  email_address: 'info@circleup.com',
  phone_number: '',
  description: 'Equity-based Crowdfunding',
  created_at: 'Tue Aug 28 17:27:38 UTC 2007',
  updated_at: 'Thu Oct 24 17:40:05 UTC 2013',
  overview: '<p>CircleUp (www.circleup.com) is an online private company investment platform. We provide accredited investors free access to direct investments in high-growth consumer product and retail private companies that were previously difficult to identify and access. For retail and consumer product entrepreneurs, we offer an efficient way to access a network of sophisticated investors as well as value added partners. </p>\n' +
  '\n' +
  '<p>Typical investments on CircleUp are food, personal care, pet product, apparel or retail/restaurant companies with $1-$10 million in revenue and are looking to raise $100,000 to $2.0 million in growth equity. Less than 2% of companies that apply are listed on our site. </p>\n' +
  '\n' +
  '<p>As the largest equity based crowdfunding site, CircleUp provides not only access to interesting consumer and retail private company investments, but also a wide ranging, sophisticated investor network. Our investor base includes retail and consumer product industry experts, venture capital, private equity and other financial professionals, business leaders, angel investors and others interested in expanding their investment portfolios with private company investments.</p>\n' +
  '\n' +
  '<p>Investors can review a curated list of private company investments, ask questions to the management team, request product samples and complete their private company investment online via CircleUp. </p>\n' +
  '\n' +
  '<p>For more information, visit the CircleUp Press Room (https://circleup.com/press/), like us on Facebook (http://www.facebook.com/CircleUp), follow us on Twitter (https://twitter.com/CircleUp), or visit our blog (https://circleup.com/blog/).</p>',
  image: {
    available_sizes: [

```

**Shards served the query: shard 1, shard 2, shard 3**

`db.company.find({"founded_year":{$gt: 2010, $lte:2022}}).explain()`

```
mongos      × | ubuntu@ip-172-31-26-40: ~ × | mongosh mongodb://ec2-54- × | ubuntu@ip-172-31-26-40: ~ × + | Ubuntu: ubuntu@ip-172-31-26-40: ~
ctrl+alt+4
stage: 'SHARD_MERGE',
shards: [
{
  shardName: 'shard2',
  connectionString: 'shard2/ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26101',
  serverInfo: {
    host: 'ip-172-31-31-11',
    port: 26101,
    version: '6.0.5',
    gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
  },
  namespace: 'companydb.company',
  indexFilterSet: false,
  parsedQuery: {
    '$and': [
      { founded_year: { '$lte': 2022 } },
      { founded_year: { '$gt': 2010 } }
    ]
  },
  queryHash: 'D7E45BD9',
  planCacheKey: 'D7E45BD9',
  maxIndexedOrSolutionsReached: false,
  maxIndexedAndSolutionsReached: false,
  maxScansToExplodeReached: false,
  winningPlan: {
    stage: 'SHARDING_FILTER',
    inputStage: {
      stage: 'COLLSCAN',
      filter: {
        '$and': [
          { founded_year: { '$lte': 2022 } },
          { founded_year: { '$gt': 2010 } }
        ]
      },
      direction: 'forward'
    }
  },
  rejectedPlans: []
},
{
  shardName: 'shard3',
  connectionString: 'shard3/ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26101',
  serverInfo: {
    host: 'ip-172-31-31-12',
    port: 26101,
    version: '6.0.5',
    gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
  },
  namespace: 'companydb.company',
  indexFilterSet: false,
  parsedQuery: {
    '$and': [
      { founded_year: { '$lte': 2022 } },
      { founded_year: { '$gt': 2010 } }
    ]
  },
  queryHash: 'D7E45BD9',
  planCacheKey: 'D7E45BD9',
  maxIndexedOrSolutionsReached: false,
  maxIndexedAndSolutionsReached: false,
  maxScansToExplodeReached: false,
  winningPlan: {
    stage: 'SHARDING_FILTER',
    inputStage: {
      stage: 'COLLSCAN',
      filter: {
        '$and': [
          { founded_year: { '$lte': 2022 } },
          { founded_year: { '$gt': 2010 } }
        ]
      },
      direction: 'forward'
    }
  },
  rejectedPlans: []
}
]
```

```

mongos                               X  ubuntu@ip-172-31-26-40: ~      X  mongosh mongodb://ec2-54-  X  ubuntu@ip-172-31-26-40: ~      X
                                         Ubuntu: ubuntu@ip-172-31-26-40: ~
                                         ctrl+alt+2

},
rejectedPlans: []
},
{
  shardName: 'shard1',
  connectionString: 'shard1/ec2-34-230-29-212.compute-1.amazonaws.com:26001,ec2-34-230-29-
serverInfo: {
  host: 'ip-172-31-19-219',
  port: 26001,
  version: '6.0.5',
  gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
},
namespace: 'companydb.company',
indexFilterSet: false,
parsedQuery: {
  '$and': [
    { founded_year: { '$lte': 2022 } },
    { founded_year: { '$gt': 2010 } }
  ]
},
queryHash: 'D7E45BD9',
planCacheKey: 'D7E45BD9',
maxIndexedOrSolutionsReached: false,
maxIndexedAndSolutionsReached: false,
maxScansToExplodeReached: false,
winningPlan: {
  stage: 'SHARDING_FILTER',
  inputStage: {
    stage: 'COLLSCAN',
    filter: {
      '$and': [
        { founded_year: { '$lte': 2022 } },
        { founded_year: { '$gt': 2010 } }
      ]
    },
    direction: 'forward'
  }
},
rejectedPlans: []
},
{
  shardName: 'shard3',
  connectionString: 'shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201,ec2-54-226-251-
serverInfo: {
  host: 'ip-172-31-22-123',
  port: 26201,
  version: '6.0.5',
  gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
},
namespace: 'companydb.company',
indexFilterSet: false,
parsedQuery: {
  '$and': [
    { founded_year: { '$lte': 2022 } },
    { founded_year: { '$gt': 2010 } }
  ]
},

```

**Execution time:** 12

```

db.company.find({"founded_year":{$gt: 2010,
$lte:2022}}).explain("executionStats").executionStats.executionTimeMillis

```

```
[direct: mongos] companydb> db.company.find({"founded_year":{$gt: 2010, $lte:2022}}).explain("executionStats").executionStats.executionTimeMillis  
12  
[direct: mongos] companydb> █
```

## 12.2: Find companies that offices in San Jose founded after 2000

*db.company.find({"offices": {\$elemMatch: {"city": "San Jose"}}, founded\_year: {\$gt: 2000}}, {\_id: 1, name: 1, founded\_year: 1, offices: 1})*

```
mongos      X  ubuntu@ip-  X  mongosh m  X  ubuntu@ip-  X  +  v  -  □  ×
{ _id: ObjectId("52cdef7c4bab8bd67529846c"),
  name: 'Asterix',
  founded_year: 2006,
  offices: [
    {
      description: null,
      address1: '4340 Stevens Creek Blvd #191',
      address2: '',
      zip_code: '95129',
      city: 'San Jose',
      state_code: 'CA',
      country_code: 'USA',
      latitude: 37.321798,
      longitude: -121.979453
    }
  ]
},
{
  _id: ObjectId("52cdef7c4bab8bd67529854c"),
  name: 'Wrike',
  founded_year: 2006,
  offices: [
    {
      description: '',
      address1: '',
      address2: '',
      zip_code: '95113',
      city: 'San Jose',
      state_code: 'CA',
      country_code: 'USA',
      latitude: 37.3973787,
      longitude: -122.0879281
    }
  ]
},
{
  _id: ObjectId("52cdef7d4bab8bd675298d7a"),
  name: 'Codefast',
  founded_year: 2004,
  offices: [
    {
      description: null,
      address1: '',
      address2: '',
      zip_code: '',
      city: 'San Jose',
      state_code: 'CA',
      country_code: 'USA',
      latitude: 37.316466,
      longitude: -121.873881
    }
  ]
},
{
  _id: ObjectId("52cdef7d4bab8bd6752993eb"),
  name: 'Roamware',
  founded_year: 2002,
  offices: [
    {
      description: 'Corporate Headquarters',
      address1: '3031 Tisch Way',
      address2: 'Suite 1000',
      zip_code: '95128',
      city: 'San Jose',
      state_code: 'CA'
    }
  ]
}
```

## Shards served the query: shard 1, shard 2, shard 3:

```
db.company.find({"offices": {$elemMatch: {"city": "San Jose"}}, founded_year: {$gt: 2000}}, {_id: 1, name: 1, founded_year: 1, offices: 1}).explain()
```

The screenshot shows a terminal window with four tabs. The active tab is titled 'mongosh' and contains the MongoDB command and its execution results. The command is:

```
[direct: mongos] companydb> db.company.find({$offices: {$elemMatch: {"city": "San Jose"}}, $founded_year: {$gt: 2000}}, {_id: 1, name: 1, $founded_year: 1, offices: 1}).explain()
```

The output shows the query plan, which includes the following stages:

- queryPlanner:** A detailed object containing the planner's version, winning plan, and various shard details.
- winningPlan:** An object describing the final execution plan, which includes:
  - stage:** 'SHARD\_MERGE'
  - shards:** An array of shard configurations, each with a shard name ('shard1'), connection string, server info (host ip-172-31-19-219, port 26001), and namespace ('companydb.company').
  - indexFilterSet:** false
  - parsedQuery:** An object representing the query structure, including '\$and' and '\$elemMatch' clauses.
  - queryHash:** 'B8E7A422'
  - planCacheKey:** 'B8E7A422'
  - maxIndexedOrSolutionsReached:** false
  - maxIndexedAndSolutionsReached:** false
  - maxScansToExplodeReached:** false

```
mongos      X  ubuntu@ip-  X  mongosh m  X  ubuntu@ip-  X  +  -  □  ×
Ubuntu:ubuntu@ip-172-31-26-40: ~
ctrl+alt+2
...ard2/ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26102,ec2-34-226-213-111.compute-1.amazonaws.com:26103',
serverInfo: {
  host: 'ip-172-31-31-11',
  port: 26101,
  version: '6.0.5',
  gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
},
namespace: 'companydb.company',
indexFilterSet: false,
parsedQuery: {
  '$and': [
    {
      'offices': { '$elemMatch': { city: { '$eq': 'San Jose' } } }
    },
    { 'founded_year': { '$gt': 2000 } }
  ]
},
queryHash: 'B8E7A422',
planCacheKey: 'B8E7A422',
maxIndexedOrSolutionsReached: false,
maxIndexedAndSolutionsReached: false,
maxScansToExplodeReached: false,
winningPlan: {
  stage: 'PROJECTION_SIMPLE',
  transformBy: { _id: 1, name: 1, founded_year: 1, offices: 1 },
  inputStage: {
    stage: 'SHARDING_FILTER',
    inputStage: {
      stage: 'COLLSCAN',
      filter: {
        '$and': [
          {
            'offices': { '$elemMatch': { city: { '$eq': 'San Jose' } } }
          },
          { 'founded_year': { '$gt': 2000 } }
        ]
      },
      direction: 'forward'
    }
  },
  rejectedPlans: []
},
{
  shardName: 'shard3',
  connectionString: 'shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201,ec2-54-226-251-91.compute-1.amazonaws.com:26203',
  serverInfo: {
    host: 'ip-172-31-22-123',
    port: 26201,
    version: '6.0.5',
    gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
  },
  namespace: 'companydb.company',
  indexFilterSet: false,
  parsedQuery: {
    '$and': [
      {
        'offices': { '$elemMatch': { city: { '$eq': 'San Jose' } } }
      },
      { 'founded_year': { '$gt': 2000 } }
    ]
  }
}
```

**Execution Time: 7**

```
db.company.find({"offices": {$elemMatch: {"city": "San Jose"}}, founded_year: {$gt: 2000}}, {"_id": 1, name: 1, founded_year: 1, offices: 1}).explain("executionStats").executionStats.executionTimeMillis
```

```
[direct: mongos] companydb> db.company.find({"offices": {$elemMatch: {"city": "San Jose"}}, founded_year: {$gt: 2000}}, {"_id": 1, name: 1, founded_year: 1, offices: 1}).explain("executionStats").executionStats.executionTimeMillis
7
[direct: mongos] companydb>
```

### 12.3: Find all companies that has founded year in: 2012 and 2013:

```
db.company.find({founded_year: {$in: [2013, 2012]}}, {"_id": 1, name: 1, founded_year: 1})
```

**Result:**

```
mongos      X  ubuntu@ip-  X  mongosh m  X  ubuntu@ip-  X  +  -  □  ×
mongos      X  ubuntu@ip-  X  mongosh m  X  Ubuntu: mongosh mongodb://
                                                    ec2-54-226-173-1.compute-1.amazonaws.com:27017/?_
                                                    directConnection=true
                                                    ctrl+alt+3
]
] Type "it" for more
[direct: mongos] companydb> db.company.find({$and: [{"$in": [2013, 2012]}], "_id": 1, "name": 1, "founded_year": 1})
[
  {
    "_id": ObjectId("52cdef7c4bab8bd6752983b6"),
    "name": "Mobiluck",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7c4bab8bd675298509"),
    "name": "Skydeck",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7d4bab8bd6752992f7"),
    "name": "Bling Easy",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7d4bab8bd675299609"),
    "name": "Non-Member Films",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7d4bab8bd675299ed8"),
    "name": "PriceHub",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7e4bab8bd67529b254"),
    "name": "Pikk",
    "founded_year": 2013
  },
  {
    "_id": ObjectId("52cdef7e4bab8bd67529b58d"),
    "name": "Pict",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7e4bab8bd67529b681"),
    "name": "Plug in SEO",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7e4bab8bd67529b8ab"),
    "name": "Veewow",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7e4bab8bd67529bd37"),
    "name": "Red Ambiental",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7c4bab8bd675297fb1"),
    "name": "headr",
    "founded_year": 2012
  },
  {
    "_id": ObjectId("52cdef7c4bab8bd6752981a0"),
    "name": "Pinger",
  }
]
```

**Shards served the query: shard 1, shard 2, shard 3**

```
db.company.find({founded_year: {$in: [2013,2012]}},{_id: 1, name: 1, founded_year: 1}).explain()
```

```
mongos      X  ubuntu@ip-  X  mongosh m  X  ubuntu@ip-  X  +  v  -  □  ×
Ubuntu: ubuntu@ip-172-31-26-40: ~
ctrl+alt+2

stage: 'SHARDING_FILTER'
shard: 'shard1'
{
  shardName: 'shard1',
  connectionString: 'shard1/ec2-34-230-29-212.compute-1.amazonaws.com:26001,ec2-34-230-29-212.compute-1.amazonaws.com:26002,ec2-34-230-29-212.compute-1.amazonaws.com:26003',
  serverInfo: {
    host: 'ip-172-31-19-219',
    port: 26001,
    version: '6.0.5',
    gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
  },
  namespace: 'companydb.company',
  indexFilterSet: false,
  parsedQuery: { founded_year: { '$in': [ 2012, 2013 ] } },
  queryHash: '82AC7EE9',
  planCacheKey: '82AC7EE9',
  maxIndexedOrSolutionsReached: false,
  maxIndexedAndSolutionsReached: false,
  maxScansToExplodeReached: false,
  winningPlan: {
    stage: 'PROJECTION_SIMPLE',
    transformBy: { _id: 1, name: 1, founded_year: 1 },
    inputStage: {
      stage: 'SHARDING_FILTER',
      inputStage: {
        stage: 'COLLSCAN',
        filter: { founded_year: { '$in': [ 2012, 2013 ] } },
        direction: 'forward'
      }
    }
  },
  rejectedPlans: []
},
{
  shardName: 'shard2',
  connectionString: 'shard2/ec2-34-226-213-111.compute-1.amazonaws.com:26101,ec2-34-226-213-111.compute-1.amazonaws.com:26102,ec2-34-226-213-111.compute-1.amazonaws.com:26103',
  serverInfo: {
    host: 'ip-172-31-31-11',
    port: 26101,
    version: '6.0.5',
    gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
  },
  ...
}
```

```
mongos      x  |  ubuntu@ip-  x  |  mongosh m  x  |  ubuntu@ip-  x  | +  | -  |  X
              |  |  |  |  |  |  |  |  |
        rejectedPlans: []
    },
{
  shardName: 'shard3',
  connectionString: 'shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201,ec2-54-226-251-91.compute-1.amazonaws.com:26202,ec2-54-226-251-91.compute-1.amazonaws.com:26203',
  serverInfo: {
    host: 'ip-172-31-22-123',
    port: 26201,
    version: '6.0.5',
    gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
  },
  namespace: 'companydb.company',
  indexFilterSet: false,
  parsedQuery: { founded_year: { '$in': [ 2012, 2013 ] } },
  queryHash: '82AC7EE9',
  planCacheKey: '82AC7EE9',
  maxIndexedOrSolutionsReached: false,
  maxIndexedAndSolutionsReached: false,
  maxScansToExplodeReached: false,
  winningPlan: {
    stage: 'PROJECTION_SIMPLE',
    transformBy: { _id: 1, name: 1, founded_year: 1 },
    inputStage: {
      stage: 'SHARDING_FILTER',
      inputStage: {
        stage: 'COLLSCAN',
        filter: { founded_year: { '$in': [ 2012, 2013 ] } },
        direction: 'forward'
      }
    },
    rejectedPlans: []
  }
}
]
```

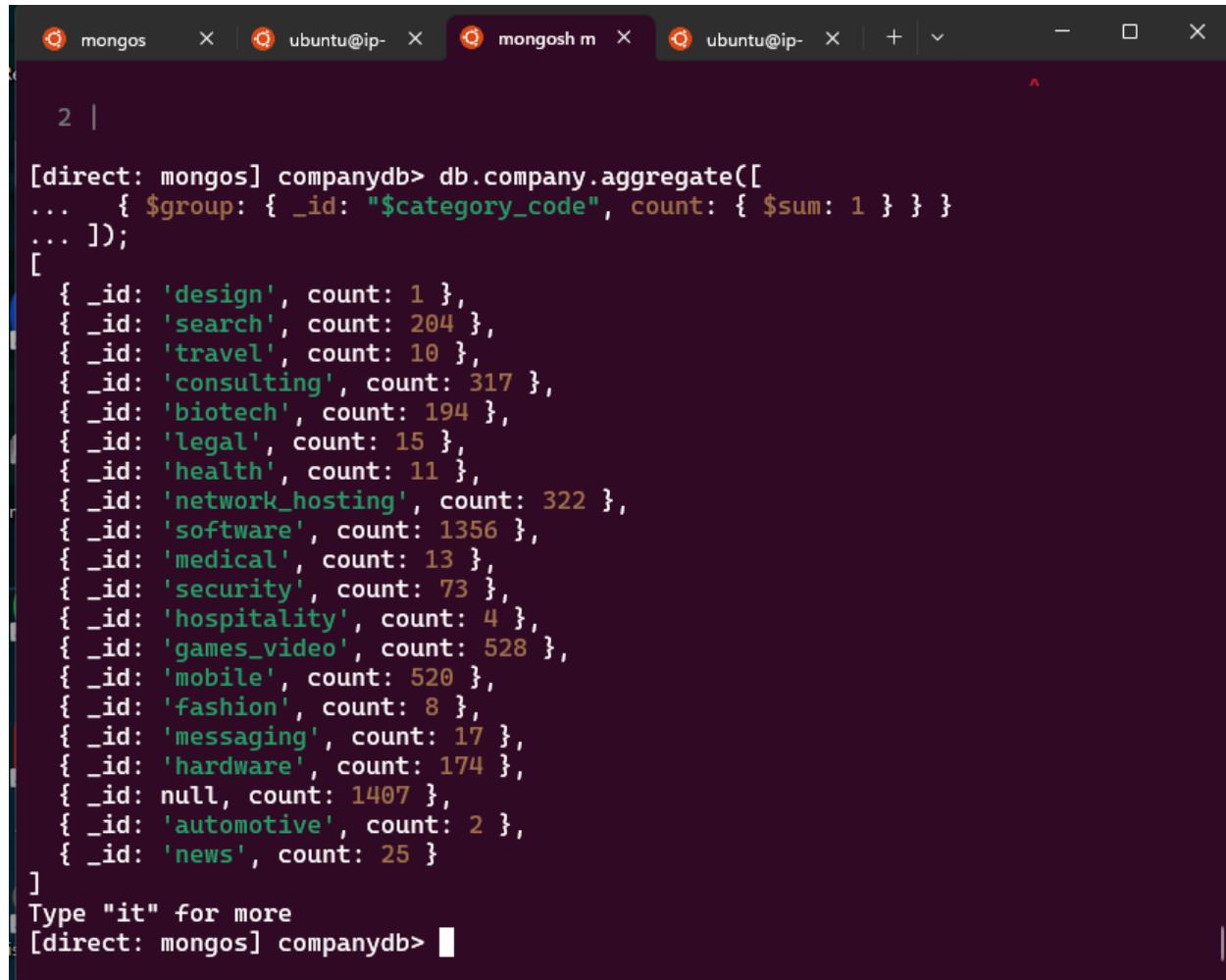
## Execution Time: 5

```
db.company.find({founded_year: {$in: [2013,2012]}}, {_id: 1, name: 1, founded_year: 1}).explain("executionStats").executionStats.executionTimeMillis
}
[direct: mongos] companydb> db.company.find({founded_year: {$in: [2013,2012]}},
{_id: 1, name: 1, founded_year: 1}).explain("executionStats").executionStats.ex
ecutionTimeMillis
5
[direct: mongos] companydb>
```

## 12.4: Find number of companies in each category\_code:

**Result:**

```
db.company.aggregate([ { $group: { _id: "$category_code", count: { $sum: 1 } } }]);
```

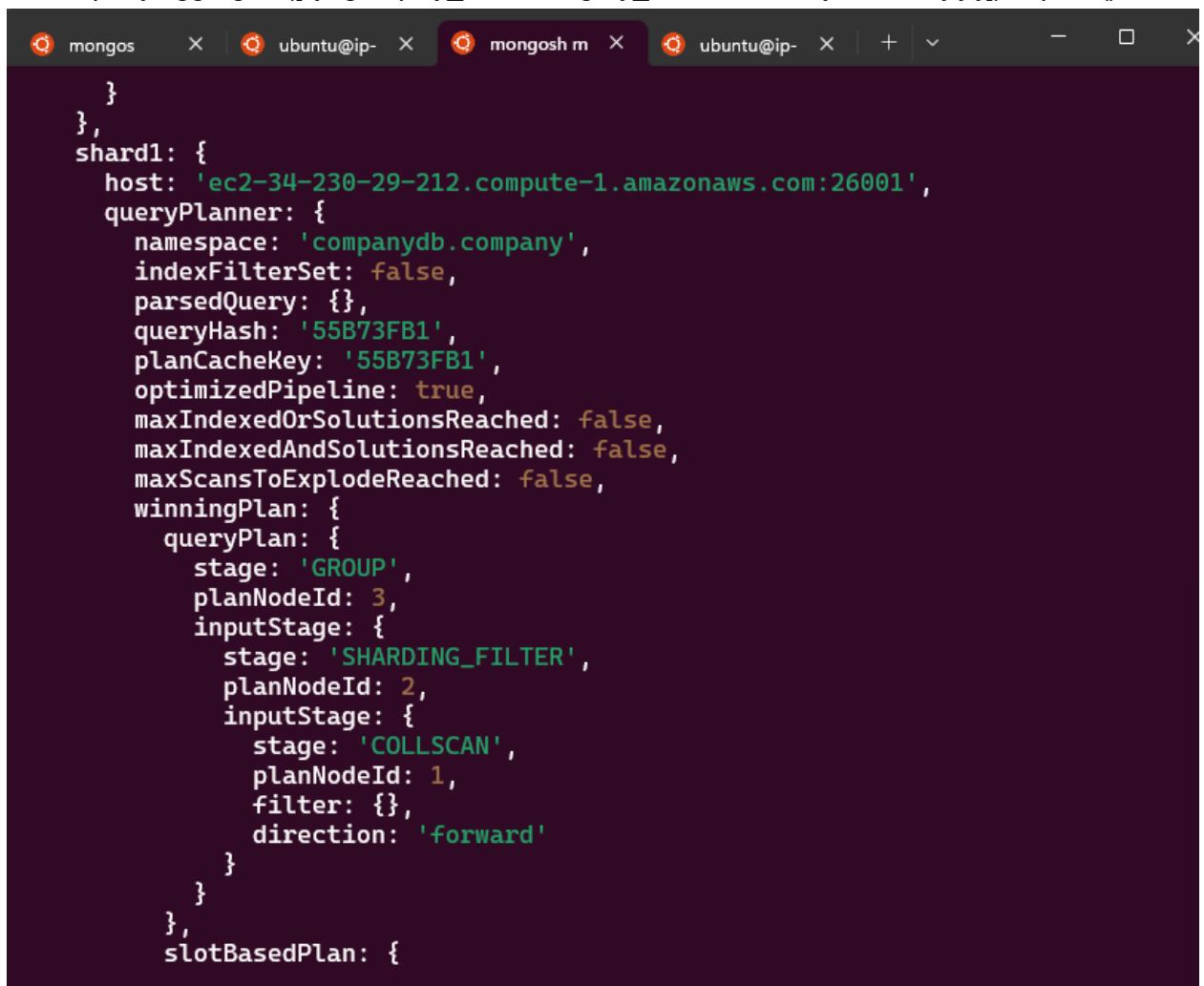


The screenshot shows a terminal window with four tabs at the top: 'mongos' (active), 'ubuntu@ip...', 'mongosh m', and 'ubuntu@ip...'. The main pane displays the results of a MongoDB aggregate query. The output starts with a cursor symbol '2 |', followed by the query itself, and then a list of documents. Each document contains a '\_id' field (the category code) and a 'count' field (the number of companies). The categories listed include 'design', 'search', 'travel', 'consulting', 'biotech', 'legal', 'health', 'network\_hosting', 'software', 'medical', 'security', 'hospitality', 'games\_video', 'mobile', 'fashion', 'messaging', 'hardware', 'null', 'automotive', and 'news'. The counts range from 1 to 1407.

```
2 |  
[  
[{"_id": "design", "count": 1},  
 {"_id": "search", "count": 204},  
 {"_id": "travel", "count": 10},  
 {"_id": "consulting", "count": 317},  
 {"_id": "biotech", "count": 194},  
 {"_id": "legal", "count": 15},  
 {"_id": "health", "count": 11},  
 {"_id": "network_hosting", "count": 322},  
 {"_id": "software", "count": 1356},  
 {"_id": "medical", "count": 13},  
 {"_id": "security", "count": 73},  
 {"_id": "hospitality", "count": 4},  
 {"_id": "games_video", "count": 528},  
 {"_id": "mobile", "count": 520},  
 {"_id": "fashion", "count": 8},  
 {"_id": "messaging", "count": 17},  
 {"_id": "hardware", "count": 174},  
 {"_id": null, "count": 1407},  
 {"_id": "automotive", "count": 2},  
 {"_id": "news", "count": 25}  
]  
Type "it" for more  
[direct: mongos] companydb> █
```

**Shards served the query:** shard 1, shard 2, shard 3

```
db.company.aggregate([ { $group: { _id: "$category_code", count: { $sum: 1 } } } ]).explain()
```



A screenshot of a terminal window showing the output of a MongoDB explain command. The terminal has four tabs open: 'mongos', 'ubuntu@ip...', 'mongosh m', and 'ubuntu@ip...'. The 'mongosh m' tab is active and contains the following JSON output:

```
{  
  "query": {  
    "shard1": {  
      "host": "ec2-34-230-29-212.compute-1.amazonaws.com:26001",  
      "queryPlanner": {  
        "namespace": "companydb.company",  
        "indexFilterSet": false,  
        "parsedQuery": {},  
        "queryHash": "55B73FB1",  
        "planCacheKey": "55B73FB1",  
        "optimizedPipeline": true,  
        "maxIndexedOrSolutionsReached": false,  
        "maxIndexedAndSolutionsReached": false,  
        "maxScansToExplodeReached": false,  
        "winningPlan": {  
          "queryPlan": {  
            "stage": "GROUP",  
            "planNodeId": 3,  
            "inputStage": {  
              "stage": "SHARDING_FILTER",  
              "planNodeId": 2,  
              "inputStage": {  
                "stage": "COLLSCAN",  
                "planNodeId": 1,  
                "filter": {},  
                "direction": "Forward"  
              }  
            }  
          },  
          "slotBasedPlan": {}  
        }  
      }  
    }  
  }  
}
```

```
mongos      X | ubuntu@ip-  X | mongosh m  X | ubuntu@ip-  X | + | v | - | □ | X
}
]
},
shards: {
  shard2: {
    host: 'ec2-34-226-213-111.compute-1.amazonaws.com:26101',
    queryPlanner: {
      namespace: 'companydb.company',
      indexFilterSet: false,
      parsedQuery: {},
      queryHash: '55B73FB1',
      planCacheKey: '55B73FB1',
      optimizedPipeline: true,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      winningPlan: {
        queryPlan: {
          stage: 'GROUP',
          planNodeId: 3,
          inputStage: {
            stage: 'SHARDING_FILTER',
            planNodeId: 2,
            inputStage: {
              stage: 'COLLSCAN',
              planNodeId: 1,
              filter: {},
              direction: 'forward'
            }
          }
        }
      }
    }
  }
}
```

### Execution Time: 176ms

```
const start = new Date();
const result = db.company.aggregate([ { $group: { _id: "$category_code", count: { $sum: 1 } } } ]);
const end = new Date();
const executionTime = end - start;
```

```
console.log(`Execution time: ${executionTime}ms`);
```

The screenshot shows a terminal window with several tabs open. The active tab is 'mongosh m' which contains the following MongoDB shell code:

```
ok: 1,
'$clusterTime': {
  clusterTime: Timestamp({ t: 1681429035, i: 1 }),
  signature: {
    hash: Binary(Buffer.from("0000000000000000000000000000000000000000000000000000000000000000", "hex"),
    0),
    keyId: Long("0")
  },
  operationTime: Timestamp({ t: 1681429033, i: 1 })
}
[direct: mongos] companydb> const start = new Date();

[direct: mongos] companydb> const result = db.company.aggregate([
  { $group: { _id: "$category_code", count: { $sum: 1 } } }
]);

[direct: mongos] companydb>

[direct: mongos] companydb> const end = new Date();

[direct: mongos] companydb>

[direct: mongos] companydb> const executionTime = end - start;

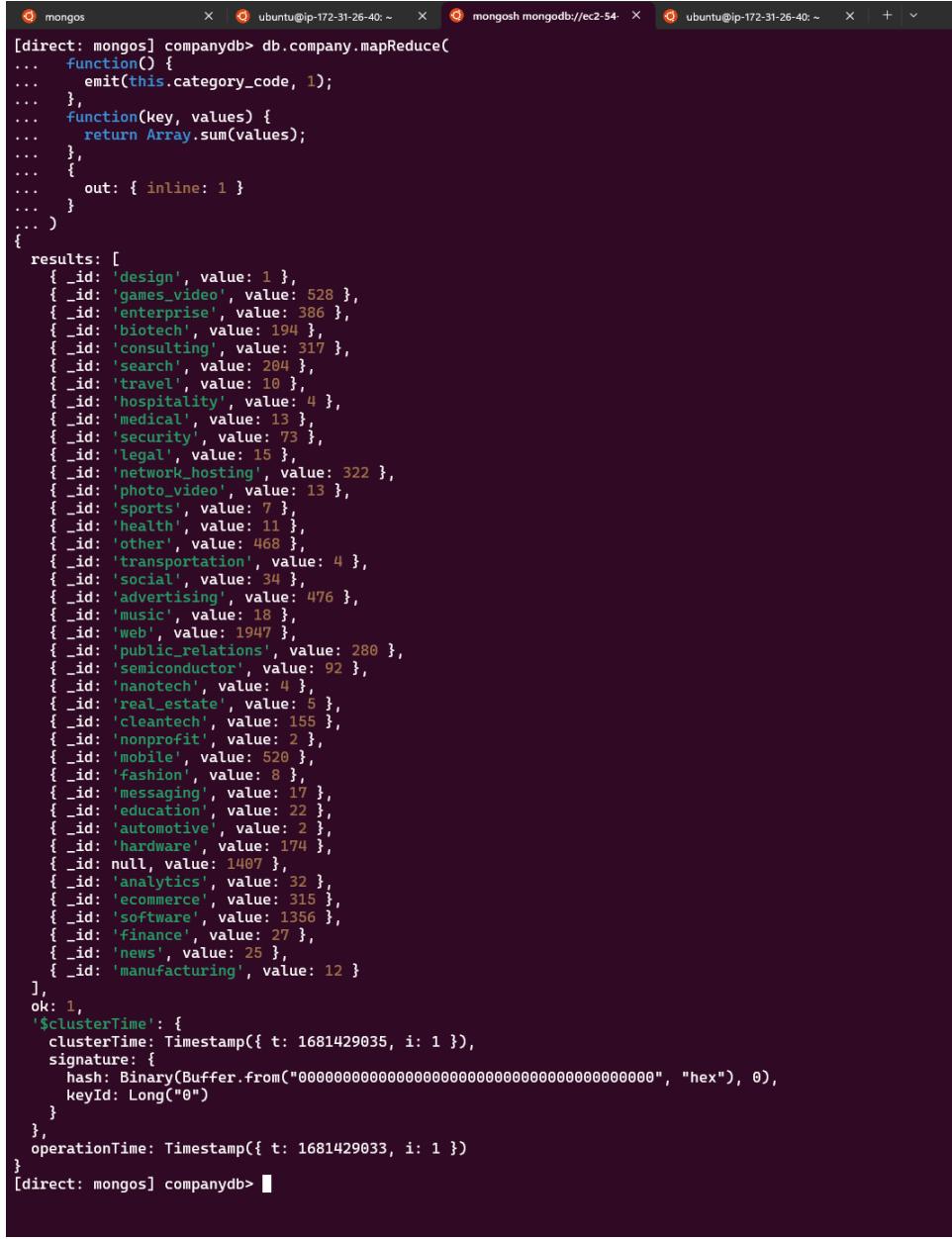
[direct: mongos] companydb>

[direct: mongos] companydb> console.log(`Execution time: ${executionTime}ms`);
Execution time: 176ms

[direct: mongos] companydb>
```

## 12.5: Map Reduce: Find numbers of companies in each category\_code:

```
db.company.mapReduce(
  function() {
    emit(this.category_code, 1);
  },
  function(key, values) {
    return Array.sum(values);
  },
  {
    out: { inline: 1 }
  });
});
```



The screenshot shows a terminal window with four tabs. The active tab displays the results of a MongoDB mapReduce operation on the 'company' collection. The command used was:

```
[direct: mongos] companydb> db.company.mapReduce(
...   function() {
...     emit(this.category_code, 1);
...   },
...   function(key, values) {
...     return Array.sum(values);
...   },
...   {
...     out: { inline: 1 }
...   }
... )
```

The results are listed as an array of documents, each containing a category code and its sum value. The top few results are:

```
results: [
  { _id: 'design', value: 1 },
  { _id: 'games_video', value: 528 },
  { _id: 'enterprise', value: 386 },
  { _id: 'biotech', value: 194 },
  { _id: 'consulting', value: 317 },
  { _id: 'search', value: 204 },
  { _id: 'travel', value: 10 },
  { _id: 'hospitality', value: 4 },
  { _id: 'medical', value: 13 },
  { _id: 'security', value: 73 },
  { _id: 'legal', value: 15 },
  { _id: 'network_hosting', value: 322 },
  { _id: 'photo_video', value: 13 },
  { _id: 'sports', value: 7 },
  { _id: 'health', value: 11 },
  { _id: 'other', value: 468 },
  { _id: 'transportation', value: 4 },
  { _id: 'social', value: 34 },
  { _id: 'advertising', value: 476 },
  { _id: 'music', value: 18 },
  { _id: 'web', value: 1947 },
  { _id: 'public_relations', value: 280 },
  { _id: 'semiconductor', value: 92 },
  { _id: 'nanotech', value: 4 },
  { _id: 'real_estate', value: 5 },
  { _id: 'cleantech', value: 155 },
  { _id: 'nonprofit', value: 2 },
  { _id: 'mobile', value: 520 },
  { _id: 'fashion', value: 8 },
  { _id: 'messaging', value: 17 },
  { _id: 'education', value: 22 },
  { _id: 'automotive', value: 2 },
  { _id: 'hardware', value: 174 },
  { _id: null, value: 1407 },
  { _id: 'analytics', value: 32 },
  { _id: 'ecommerce', value: 315 },
  { _id: 'software', value: 1356 },
  { _id: 'finance', value: 27 },
  { _id: 'news', value: 25 },
  { _id: 'manufacturing', value: 12 }
],
```

ok: 1,  
\$clusterTime: {  
 clusterTime: Timestamp({ t: 1681429035, i: 1 }),  
 signature: {  
 hash: Binary(Buffer.from("00000000000000000000000000000000", "hex"), 0),  
 keyId: Long("0")  
 }  
},  
operationTime: Timestamp({ t: 1681429033, i: 1 })

**Execution Time:** 468ms

```
const start = new Date();
const result = db.company.mapReduce(
  function() {
    emit(this.category_code, 1);
  },
  function(key, values) {
    return Array.sum(values);
},
{
```

```

out: { inline: 1 }
});

const end = new Date();

const executionTime = end - start;

console.log(`Execution time: ${executionTime}ms`);




The terminal window shows the following session:



```

[mongos] companydb> const start = new Date();
[mongos] companydb> const result = db.company.mapReduce(
...   function() {
...     emit(this.category_code, 1);
...   },
...   function(key, values) {
...     return Array.sum(values);
...   },
...   {
...     out: { inline: 1 }
...   });
[mongos] companydb>
[mongos] companydb>
[mongos] companydb> const end = new Date();
[mongos] companydb>
[mongos] companydb> const executionTime = end - start;
[mongos] companydb>
[mongos] companydb> console.log(`Execution time: ${executionTime}ms`);
Execution time: 468ms
[mongos] companydb> 
```


```

## 12.6: Update a company named 'Wetpaint', founded\_year to become 2012

Before update:

```
mongos      x  |  ubuntu@ip-  x  |  mongosh m  x  |  ubuntu@ip-  x  | + | -  □  ×
}) " , (intermediate value)(intermediate value) , (intermediate value)(intermediate value)(intermediate value).explain is not a function
[direct: mongos] companydb> db.company.find({name: "Wetpaint"})
[
  {
    _id: ObjectId("52cdef7c4bab8bd675297d8a"),
    name: 'Wetpaint',
    permalink: 'abc2',
    crunchbase_url: 'http://www.crunchbase.com/company/wetpaint',
    homepage_url: 'http://wetpaint-inc.com',
    blog_url: 'http://digitalquarters.net/',
    blog_feed_url: 'http://digitalquarters.net/feed/',
    twitter_username: 'BachelrWetpaint',
    category_code: 'web',
    number_of_employees: 47,
    founded_year: 2005,
    founded_month: 10,
    founded_day: 17,
    deadpooled_year: 1,
    tag_list: 'wiki, seattle, elowitz, media-industry, media-platform, social-distribution-system',
    alias_list: '',
    email_address: 'info@wetpaint.com',
    phone_number: '206.859.6300',
    description: 'Technology Platform Company',
    created_at: ISODate("2007-05-25T06:51:27.000Z"),
    updated_at: 'Sun Dec 08 07:15:44 UTC 2013',
    overview: '<p>Wetpaint is a technology platform company that uses its proprietary state-of-the-art technology and expertise in social media to build and monetize audiences for digital publishers. Wetpaint's own online property, Wetpa
```

### After update:

```
db.company.updateOne({name: "Wetpaint"}, {$set: {founded_year: 2012}})
```

```
[direct: mongos] companydb> db.company.updateOne({name: "Wetpaint"}, {$set: {fo  
unded_year: 2012}})  
{  
    acknowledged: true,  
    insertedId: null,  
    matchedCount: 1,  
    modifiedCount: 1,  
    upsertedCount: 0  
}  
[direct: mongos] companydb> █
```

```
mongos      X  ubuntu@ip-  X  mongosh m  X  ubuntu@ip-  X  +  ▾  —  □  ×  
matchedCount: 1,  
modifiedCount: 1,  
upsertedCount: 0  
}  
[direct: mongos] companydb> db.company.find({name: "Wetpaint"})  
[  
  {  
    _id: ObjectId("52cdef7c4bab8bd675297d8a"),  
    name: 'Wetpaint',  
    permalink: 'abc2',  
    crunchbase_url: 'http://www.crunchbase.com/company/wetpaint',  
    homepage_url: 'http://wetpaint-inc.com',  
    blog_url: 'http://digitalquarters.net/',  
    blog_feed_url: 'http://digitalquarters.net/feed/',  
    twitter_username: 'BachelrWetpaint',  
    category_code: 'web',  
    number_of_employees: 47,  
    founded_year: 2012,  
    founded_month: 10,  
    founded_day: 17,  
    deadpooled_year: 1,  
    tag_list: 'wiki, seattle, elowitz, media-industry, media-platform, social-d  
istribution-system',  
    alias_list: '',  
    email_address: 'info@wetpaint.com',  
    phone_number: '206.859.6300',  
    description: 'Technology Platform Company',  
    created_at: ISODate("2007-05-25T06:51:27.000Z"),  
    updated_at: 'Sun Dec 08 07:15:44 UTC 2013',  
    overview: '<p>Wetpaint is a technology platform company that uses its propr
```

## Shard served the query: Shard 3

```
mongos      x  |  ubuntu@ip-  x  |  mongosh m  x  |  ubuntu@ip-  x  | + | v
name: "Wetpaint"}, {$set: {founded_year: 2012}})
TypeError: db.cityinfo.e ... tats").updateOne is not a function
[direct: mongos] companydb> db.company.find({name: "Wetpaint"}).explain()
{
  queryPlanner: {
    mongosPlannerVersion: 1,
    winningPlan: {
      stage: 'SINGLE_SHARD',
      shards: [
        {
          shardName: 'shard3',
          connectionString: 'shard3/ec2-54-226-251-91.compute-1.amazonaws.com:26201,ec2-54-226-251-91.compute-1.amazonaws.com:26202,ec2-54-226-251-91.compute-1.amazonaws.com:26203',
          serverInfo: {
            host: 'ip-172-31-22-123',
            port: 26201,
            version: '6.0.5',
            gitVersion: 'c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d'
          },
          namespace: 'companydb.company',
          indexFilterSet: false,
          parsedQuery: { name: { '$eq': 'Wetpaint' } },
          queryHash: '64908032',
          planCacheKey: 'A6C0273F',
          maxIndexedOrSolutionsReached: false,
          maxIndexedAndSolutionsReached: false,
          maxScansToExplodeReached: false,
          winningPlan: {
            stage: 'FETCH',
            keyPattern: null,
            limit: null,
            offset: null,
            sort: null
          }
        }
      ]
    }
  }
}
```

## 13. rs.status()

## a. Config:

```
config [direct: primary] test>
config [direct: primary] test> rs.status()
{
  set: 'config',
  date: ISODate("2023-04-13T23:52:33.595Z"),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  configsvr: true,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1681429953, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-04-13T23:52:33.057Z"),
    readConcernMajorityOptime: { ts: Timestamp({ t: 1681429953, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1681429953, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1681429953, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-04-13T23:52:33.057Z"),
    lastDurableWallTime: ISODate("2023-04-13T23:52:33.057Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1681429904, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate("2023-04-13T20:39:39.404Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681418379, i: 1 }), t: Long("-1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1681418379, i: 1 }), t: Long("-1") },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    newTermStartDate: ISODate("2023-04-13T20:39:39.458Z"),
    whMajorityWriteAvailabilityDate: ISODate("2023-04-13T20:39:39.664Z")
  },
  members: [
    {
      _id: 0,
      name: 'ec2-3-94-196-162.compute-1.amazonaws.com:27031',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 11682,
      optime: { ts: Timestamp({ t: 1681429953, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-04-13T23:52:33.000Z"),
      lastAppliedWallTime: ISODate("2023-04-13T23:52:33.057Z"),
      lastDurableWalltime: ISODate("2023-04-13T23:52:33.057Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1681418379, i: 2 }),
      electionDate: ISODate("2023-04-13T20:39:39.000Z"),
      configVersion: 5,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ]
}
```

```
config                                     mongosh mongodb://127.0.0.1:27017/test> 
                                         ubuntu@ip-172-31-17-52:~> 
                                         ubuntu@ip-172-31-17-52:~> 
                                         ubuntu@ip-172-31-17-52:~> 

    self: true,
    lastHeartbeatMessage: ''
},
{
  _id: 1,
  name: 'ec2-3-94-196-162.compute-1.amazonaws.com:27032',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 11459,
  optime: { ts: Timestamp({ t: 1681429953, i: 1 }), t: Long("1") },
  optimeDurable: { ts: Timestamp({ t: 1681429953, i: 1 }), t: Long("1") },
  optimeDate: ISODate("2023-04-13T23:52:33.000Z"),
  optimeDurableDate: ISODate("2023-04-13T23:52:33.000Z"),
  lastAppliedWallTime: ISODate("2023-04-13T23:52:33.057Z"),
  lastDurableWallTime: ISODate("2023-04-13T23:52:33.057Z"),
  lastHeartbeat: ISODate("2023-04-13T23:52:33.465Z"),
  lastHeartbeatRecv: ISODate("2023-04-13T23:52:33.480Z"),
  pingMs: Long("0"),
  lastHeartbeatMessage: '',
  syncSourceHost: 'ec2-3-94-196-162.compute-1.amazonaws.com:27031',
  syncSourceId: 0,
  infoMessage: '',
  configVersion: 5,
  configTerm: 1
},
{
  _id: 2,
  name: 'ec2-3-94-196-162.compute-1.amazonaws.com:27033',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 11452,
  optime: { ts: Timestamp({ t: 1681429951, i: 2 }), t: Long("1") },
  optimeDurable: { ts: Timestamp({ t: 1681429951, i: 2 }), t: Long("1") },
  optimeDate: ISODate("2023-04-13T23:52:31.000Z"),
  optimeDurableDate: ISODate("2023-04-13T23:52:31.000Z"),
  lastAppliedWallTime: ISODate("2023-04-13T23:52:33.057Z"),
  lastDurableWallTime: ISODate("2023-04-13T23:52:33.057Z"),
  lastHeartbeat: ISODate("2023-04-13T23:52:32.380Z"),
  lastHeartbeatRecv: ISODate("2023-04-13T23:52:33.013Z"),
  pingMs: Long("0"),
  lastHeartbeatMessage: '',
  syncSourceHost: 'ec2-3-94-196-162.compute-1.amazonaws.com:27032',
  syncSourceId: 1,
  infoMessage: '',
  configVersion: 5,
  configTerm: 1
},
],
ok: 1,
lastCommittedOpTime: Timestamp({ t: 1681429953, i: 1 }),
'$clusterTime': {
  clusterTime: Timestamp({ t: 1681429953, i: 1 }),
  signature: {
    hash: Binary(Buffer.from("0000000000000000000000000000000000000000000000000000000000000000", "hex"), 0),
    keyId: Long("0")
  }
},
operationTime: Timestamp({ t: 1681429953, i: 1 })
}
config [direct: primary] test> 
```

**b. Shard 1:**

```
shard1                               mongosh mongodb://127.0.0.1:27017/test> ubuntu@ip-172-31-19-219:~> ubuntu@ip-172-31-19-219:~>

{
  _id: 1,
  name: 'ec2-34-230-29-212.compute-1.amazonaws.com:26002',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 10170,
  optime: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
  optimeDurable: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
  optimeDate: ISODate("2023-04-13T23:53:27.000Z"),
  optimeDurableDate: ISODate("2023-04-13T23:53:27.000Z"),
  lastAppliedWallTime: ISODate("2023-04-13T23:53:27.153Z"),
  lastDurableWallTime: ISODate("2023-04-13T23:53:27.153Z"),
  lastHeartbeat: ISODate("2023-04-13T23:53:32.431Z"),
  lastHeartbeatRecv: ISODate("2023-04-13T23:53:32.462Z"),
  pingMs: Long("0"),
  lastHeartbeatMessage: '',
  syncSourceHost: 'ec2-34-230-29-212.compute-1.amazonaws.com:26001',
  syncSourceId: 0,
  infoMessage: '',
  configVersion: 5,
  configTerm: 1
},
{
  _id: 2,
  name: 'ec2-34-230-29-212.compute-1.amazonaws.com:26003',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 10163,
  optime: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
  optimeDurable: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
  optimeDate: ISODate("2023-04-13T23:53:27.000Z"),
  optimeDurableDate: ISODate("2023-04-13T23:53:27.000Z"),
  lastAppliedWallTime: ISODate("2023-04-13T23:53:27.153Z"),
  lastDurableWallTime: ISODate("2023-04-13T23:53:27.153Z"),
  lastHeartbeat: ISODate("2023-04-13T23:53:32.359Z"),
  lastHeartbeatRecv: ISODate("2023-04-13T23:53:32.907Z"),
  pingMs: Long("0"),
  lastHeartbeatMessage: '',
  syncSourceHost: 'ec2-34-230-29-212.compute-1.amazonaws.com:26002',
  syncSourceId: 1,
  infoMessage: '',
  configVersion: 5,
  configTerm: 1
},
],
ok: 1,
lastCommittedOpTime: Timestamp({ t: 1681430007, i: 1 }),
'$clusterTime': {
  clusterTime: Timestamp({ t: 1681430007, i: 2 }),
  signature: {
    hash: Binary(Buffer.from("00000000000000000000000000000000", "hex"), 0),
    keyId: Long("0")
  }
},
operationTime: Timestamp({ t: 1681430007, i: 1 })
}
shard1 [direct: primary] test>
```

```
shard1 mongosh mongodb://127.0.0.1:27017/ ubuntu@ip-172-31-19-219: ~ | ubuntu@ip-172-31-19-219: ~
},
operationTime: Timestamp({ t: 1681419851, i: 1 })
}
shard1 [direct: primary] test> rs.status()
{
  set: 'shard1',
  date: ISODate("2023-04-13T23:53:33.267Z"),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-04-13T23:53:27.153Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-04-13T23:53:27.153Z"),
    lastDurableWallTime: ISODate("2023-04-13T23:53:27.153Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1681429967, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate("2023-04-13T21:03:46.636Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681419826, i: 1 }), t: Long("-1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1681419826, i: 1 }), t: Long("-1") },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    newTermStartDate: ISODate("2023-04-13T21:03:46.706Z"),
    wMajorityWriteAvailabilityDate: ISODate("2023-04-13T21:03:46.828Z")
  },
  members: [
    {
      _id: 0,
      name: 'ec2-34-230-29-212.compute-1.amazonaws.com:26001',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 10917,
      optime: { ts: Timestamp({ t: 1681430007, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-04-13T23:53:27.000Z"),
      lastAppliedWallTime: ISODate("2023-04-13T23:53:27.153Z"),
      lastDurableWallTime: ISODate("2023-04-13T23:53:27.153Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1681419826, i: 2 }),
      electionDate: ISODate("2023-04-13T21:03:46.000Z"),
      configVersion: 5,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ]
}
```

c. Shard 2:

```
shard2 [direct: primary] test> rs.status()
{
  set: 'shard2',
  date: ISODate("2023-04-13T23:54:31.065Z"),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1681430063, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-04-13T23:54:23.212Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1681430063, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1681430063, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1681430063, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-04-13T23:54:23.212Z"),
    lastDurableWallTime: ISODate("2023-04-13T23:54:23.212Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1681430043, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate("2023-04-13T21:11:02.702Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681420262, i: 1 }), t: Long("-1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1681420262, i: 1 }), t: Long("-1") },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    newTermStartDate: ISODate("2023-04-13T21:11:02.790Z"),
    wMajorityWriteAvailabilityDate: ISODate("2023-04-13T21:11:02.843Z")
  },
  members: [
    {
      _id: 0,
      name: 'ec2-34-226-213-111.compute-1.amazonaws.com:26101',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 9837,
      optime: { ts: Timestamp({ t: 1681430063, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-04-13T23:54:23.000Z"),
      lastAppliedWallTime: ISODate("2023-04-13T23:54:23.212Z"),
      lastDurableWallTime: ISODate("2023-04-13T23:54:23.212Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1681420262, i: 2 }),
      electionDate: ISODate("2023-04-13T21:11:02.000Z"),
      configVersion: 5,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ]
}
```



d. Shard 3:

```
shard3 [direct: primary] test> rs.status()
{
  set: 'shard3',
  date: ISODate("2023-04-13T23:55:08.922Z"),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1681430100, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-04-13T23:55:00.142Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1681430100, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1681430100, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1681430100, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-04-13T23:55:00.142Z"),
    lastDurableWallTime: ISODate("2023-04-13T23:55:00.142Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1681430100, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate("2023-04-13T21:16:59.775Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1681420619, i: 1 }), t: Long("-1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1681420619, i: 1 }), t: Long("-1") },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    newTermStartDate: ISODate("2023-04-13T21:16:59.817Z"),
    wMajorityWriteAvailabilityDate: ISODate("2023-04-13T21:16:59.857Z")
  },
  members: [
    {
      _id: 0,
      name: 'ec2-54-226-251-91.compute-1.amazonaws.com:26201',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 9536,
      optime: { ts: Timestamp({ t: 1681430100, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-04-13T23:55:00.000Z"),
      lastAppliedWallTime: ISODate("2023-04-13T23:55:00.142Z"),
      lastDurableWallTime: ISODate("2023-04-13T23:55:00.142Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1681420619, i: 2 }),
      electionDate: ISODate("2023-04-13T21:16:59.000Z"),
      configVersion: 5,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ]
}
```



## **14.**

**node0: mongos: ec2-54-226-173-1.compute-1.amazonaws.com**  
node0:port#27017 mongos server

**node1: configs: ec2-3-94-196-162.compute-1.amazonaws.com**  
node1:port#27031 config server PRIMARY  
node1:port#27032 config server SECONDARY  
node1:port#27033 config server SECONDARY

**node2: shard1: ec2-34-230-29-212.compute-1.amazonaws.com**  
node2:port#26001 shard1 server PRIMARY  
node2:port#26002 shard1 server SECONDARY  
node2:port#26003 shard1 server SECONDARY

**node3: shard2: ec2-34-226-213-111.compute-1.amazonaws.com**  
node3:port#26101 shard2 server PRIMARY  
node3:port#26102 shard2 server SECONDARY  
node3:port#26103 shard2 server SECONDARY

**node4: shard3: ec2-54-226-251-91.compute-1.amazonaws.com**  
node4:port#26201 shard3 server PRIMARY  
node4:port#26202 shard3 server SECONDARY  
node4:port#26203 shard3 server SECONDARY