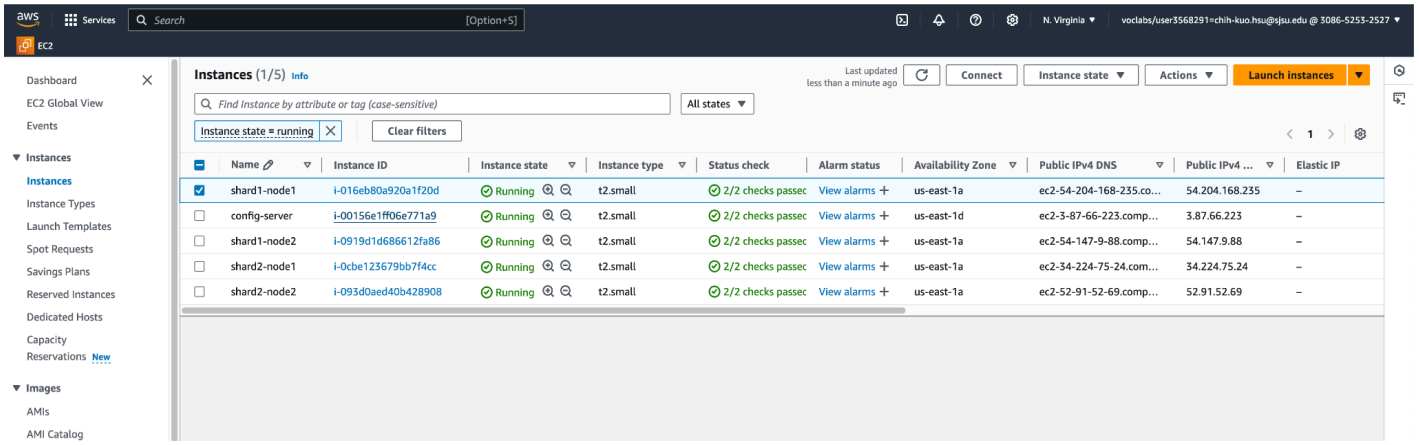
Cs157c

Chihkuo Hsu

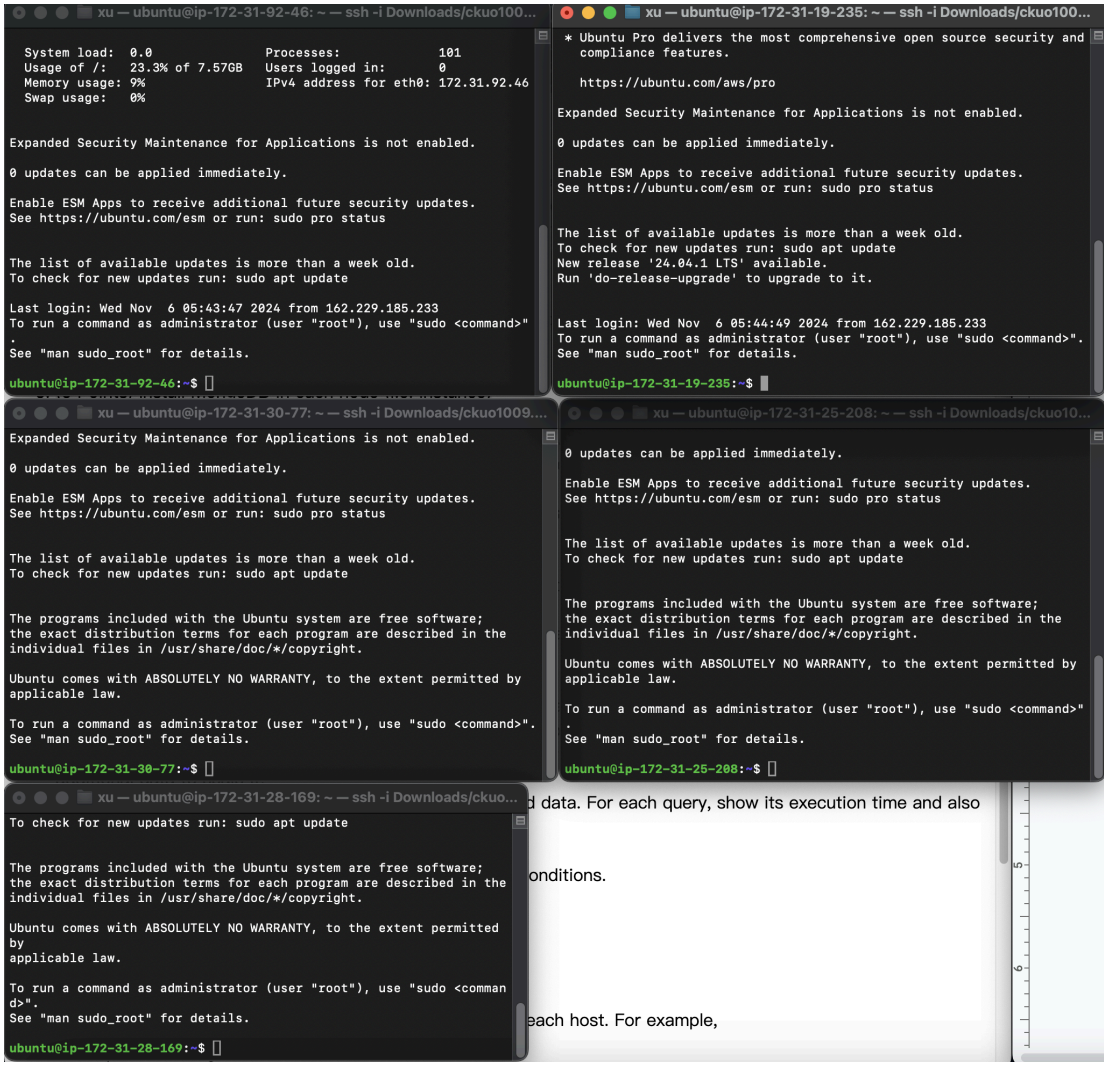
1.(5 Points) Set up nodes in AWS. Determine the number of nodes based on your deployment plan. (There should be at least 3 nodes.)

One node will be configured as a replica set for the config servers.

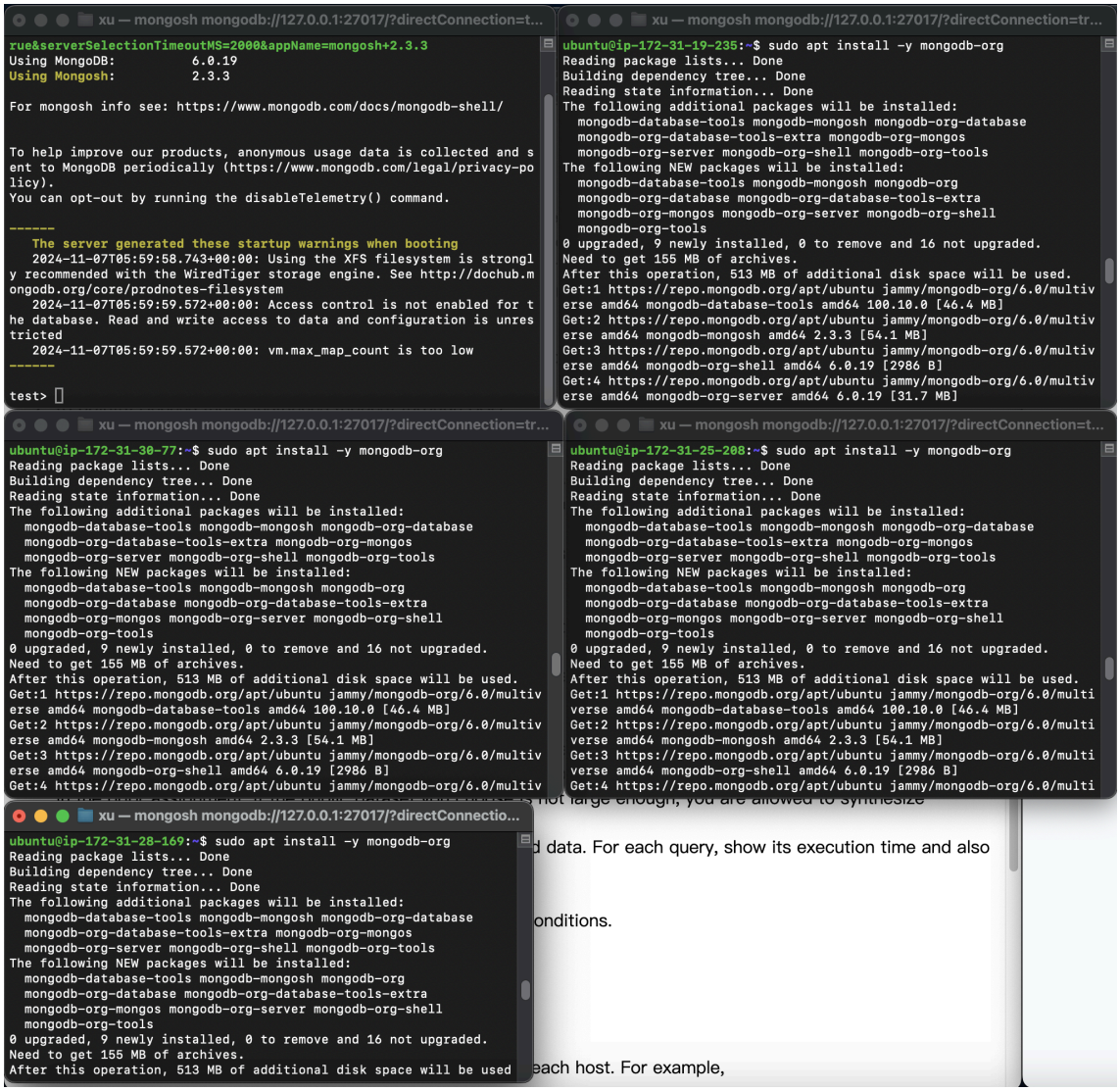
The remaining four nodes will be set up as two shards, each configured with a replica set of two nodes.



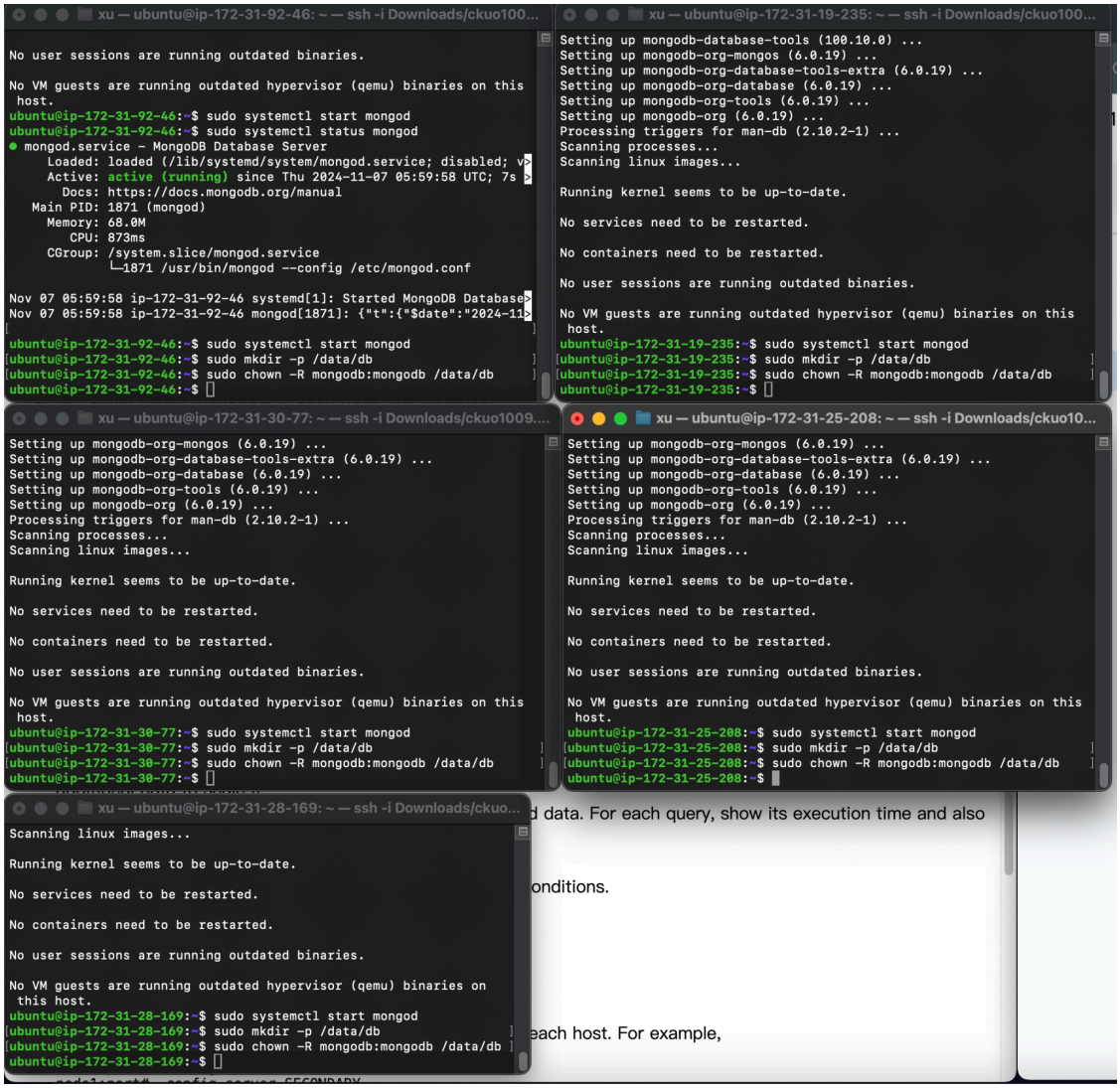
2. (5 points) Access these instances (nodes) through SSH


xu — ubuntu@ip-172-31-92-46: ~ — ssh -i Downloads/ckuo100...
System load: 0.0 Processes: 101
Usage of /: 23.3% of 7.57GB Users logged in: 0
Memory usage: 9% IPv4 address for eth0: 172.31.92.46
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Wed Nov 6 05:43:47 2024 from 162.229.185.233
To run a command as administrator (user "root"), use "sudo <command>"
.
See "man sudo_root" for details.
ubuntu@ip-172-31-92-46:~$
xu — ubuntu@ip-172-31-19-235: ~ — ssh -i Downloads/ckuo100...
* Ubuntu Pro delivers the most comprehensive open source security and
compliance features.
https://ubuntu.com/aws/pro
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Wed Nov 6 05:44:49 2024 from 162.229.185.233
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-19-235:~$
xu — ubuntu@ip-172-31-30-77: ~ — ssh -i Downloads/ckuo1009....
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-30-77:~$
xu — ubuntu@ip-172-31-25-208: ~ — ssh -i Downloads/ckuo10...
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>"
.
See "man sudo_root" for details.
ubuntu@ip-172-31-25-208:~$
xu — ubuntu@ip-172-31-28-169: ~ — ssh -i Downloads/ckuo...
To check for new updates run: sudo apt update
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted
by
applicable law.
To run a command as administrator (user "root"), use "sudo <comman
d>".
See "man sudo_root" for details.
ubuntu@ip-172-31-28-169:~$
d data. For each query, show its execution time and also
onditions.
each host. For example,

3. (5 Points) Install MongoDB in each node (i.e. instance)



4. (3 points) Create a directory to store database in each node



5. (5 points) Specify Public and Private IP Addresses of AWS instances used in your solution.

config-server: Public IP - 3.88.181.36, Private IP -172.31.92.46

shard1-node1: Public IP - 34.235.87.158, Private IP - 172.31.19.235

shard1-node2: Public IP - 54.157.22.128, Private IP - 172.31.30.77

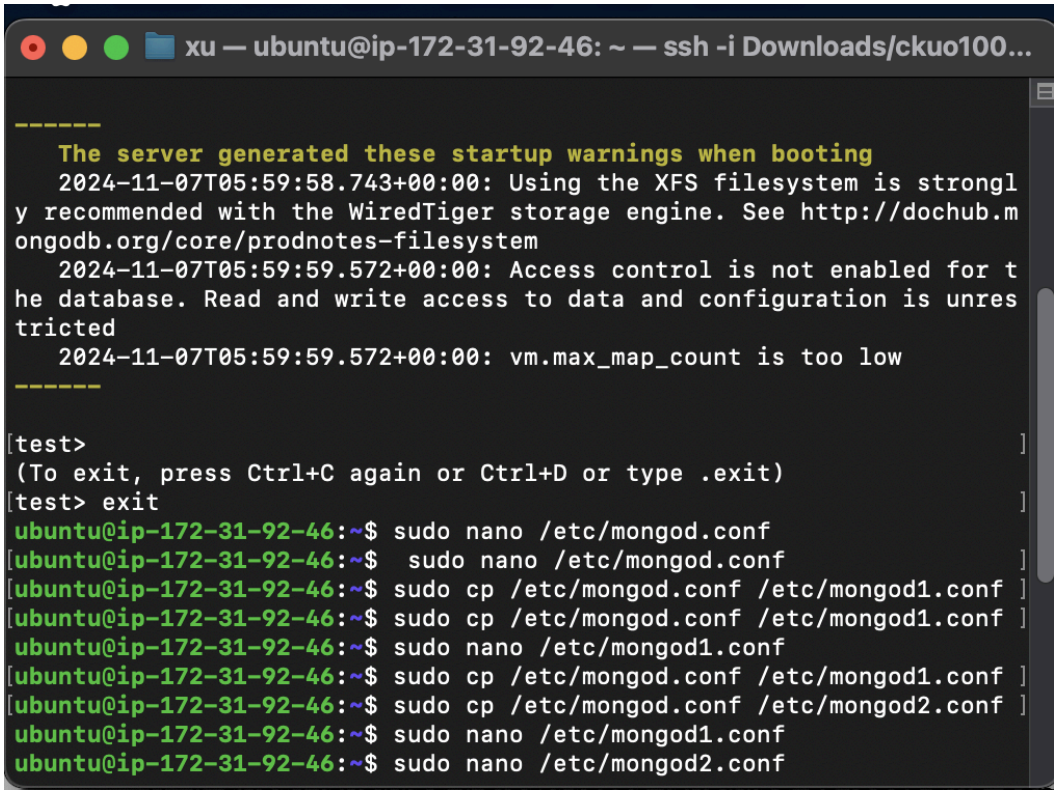
shard2-node1: Public IP - 54.152.172.105, Private IP - 172.31.25.208

shard2-node2: Public IP - 34.227.226.218, Private IP - 172.31.28.169

6. (6 points) Set up and launch three config servers in a replica set.

On the config-server instance:

- Configure and start three MongoDB config server instances, each using a different configuration file and port 27019, 27020, 27021.
- Initialize the replica set so that these three config server instances act as members of the same replica set.




```
ubuntu@ip-172-31-92-46:~$ sudo mongod --config /etc/mongod.conf --for
k --logpath /var/log/mongodb/mongod.log
about to fork child process, waiting until server is ready for connec
tions.
forked process: 13164
child process started successfully, parent exiting
ubuntu@ip-172-31-92-46:~$ sudo mongod --config /etc/mongod1.conf --fo
rk --logpath /var/log/mongodb/mongod1.log
about to fork child process, waiting until server is ready for connec
tions.
forked process: 13230
child process started successfully, parent exiting
ubuntu@ip-172-31-92-46:~$ sudo mongod --config /etc/mongod2.conf --fo
rk --logpath /var/log/mongodb/mongod2.log
about to fork child process, waiting until server is ready for connec
tions.
forked process: 13292
child process started successfully, parent exiting
ubuntu@ip-172-31-92-46:~$
```

```
test> rs.initiate({
...   _id: "configReplSet",
...   configsvr: true,
...   members: [
...     { _id: 0, host: "172.31.92.46:27019" },
...     { _id: 1, host: "172.31.92.46:27020" },
...     { _id: 2, host: "172.31.92.46:27021" }
...   ]
... })
{ ok: 1, lastCommittedOpTime: Timestamp({ t: 1730978788, i: 1 }) }
configReplSet [direct: other] test>
```


xu — mongosh mongodb://127.0.0.1:27019/?directConnection=tr...
configReplSet [direct: other] test> rs.status()
{
set: 'configReplSet',
date: ISODate('2024-11-07T11:27:30.288Z'),
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: 1730978849, i: 1 }), t:
Long('1') },
lastCommittedWallTime: ISODate('2024-11-07T11:27:29.995Z'),
readConcernMajorityOpTime: { ts: Timestamp({ t: 1730978849, i: 1 }
), t: Long('1') },
appliedOpTime: { ts: Timestamp({ t: 1730978849, i: 1 }), t: Long('
1') },
durableOpTime: { ts: Timestamp({ t: 1730978849, i: 1 }), t: Long('
1') },
lastAppliedWallTime: ISODate('2024-11-07T11:27:29.995Z'),
lastDurableWallTime: ISODate('2024-11-07T11:27:29.995Z')
},
lastStableRecoveryTimestamp: Timestamp({ t: 1730978847, i: 1 }),
electionCandidateMetrics: {
lastElectionReason: 'electionTimeout',
lastElectionDate: ISODate('2024-11-07T11:26:38.629Z'),
electionTerm: Long('1'),
lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1730978788, i:
1 }), t: Long('-1') },
lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1730978788, i: 1 })
, t: Long('-1') },
numVotesNeeded: 2,
priorityAtElection: 1,
electionTimeoutMillis: Long('10000'),
numCatchUpOps: Long('0'),
newTermStartDate: ISODate('2024-11-07T11:26:38.685Z'),
wMajorityWriteAvailabilityDate: ISODate('2024-11-07T11:26:39.227Z'
)
},
members: [
{
_id: 0,
name: '172.31.92.46:27019',
health: 1,
state: 1,
stateStr: 'PRIMARY',
uptime: 83,
optime: { ts: Timestamp({ t: 1730978849, i: 1 }), t: Long('1') }
,
optimeDate: ISODate('2024-11-07T11:27:29.000Z'),
lastAppliedWallTime: ISODate('2024-11-07T11:27:29.995Z'),
lastDurableWallTime: ISODate('2024-11-07T11:27:29.995Z'),
syncSourceHost: '',
syncSourceId: -1,
infoMessage: 'Could not find member to sync from',
electionTime: Timestamp({ t: 1730978798, i: 1 }),
electionDate: ISODate('2024-11-07T11:26:38.000Z'),
configVersion: 1,
configTerm: 1,
self: true,
lastHeartbeatMessage: ''
},
{

```
    },
    {
      _id: 1,
      name: '172.31.92.46:27020',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 61,
      optime: { ts: Timestamp({ t: 1730978847, i: 1 }), t: Long('1') }
,
      optimeDurable: { ts: Timestamp({ t: 1730978847, i: 1 }), t: Long
('1') },
      optimeDate: ISODate('2024-11-07T11:27:27.000Z'),
      optimeDurableDate: ISODate('2024-11-07T11:27:27.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T11:27:29.995Z'),
      lastDurableWallTime: ISODate('2024-11-07T11:27:29.995Z'),
      lastHeartbeat: ISODate('2024-11-07T11:27:28.645Z'),
      lastHeartbeatRecv: ISODate('2024-11-07T11:27:29.657Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.92.46:27019',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 1
    },
    {
      _id: 2,
      name: '172.31.92.46:27021',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 61,
      optime: { ts: Timestamp({ t: 1730978847, i: 1 }), t: Long('1') }
,
      optimeDurable: { ts: Timestamp({ t: 1730978847, i: 1 }), t: Long
('1') },
      optimeDate: ISODate('2024-11-07T11:27:27.000Z'),
      optimeDurableDate: ISODate('2024-11-07T11:27:27.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T11:27:29.995Z'),
      lastDurableWallTime: ISODate('2024-11-07T11:27:29.995Z'),
      lastHeartbeat: ISODate('2024-11-07T11:27:28.645Z'),
      lastHeartbeatRecv: ISODate('2024-11-07T11:27:29.651Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.92.46:27019',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 1
    }
  ],
  ok: 1,
  lastCommittedOpTime: Timestamp({ t: 1730978849, i: 1 }),
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730978849, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730978849, i: 1 })
}
configReplSet [direct: primary] test>
```

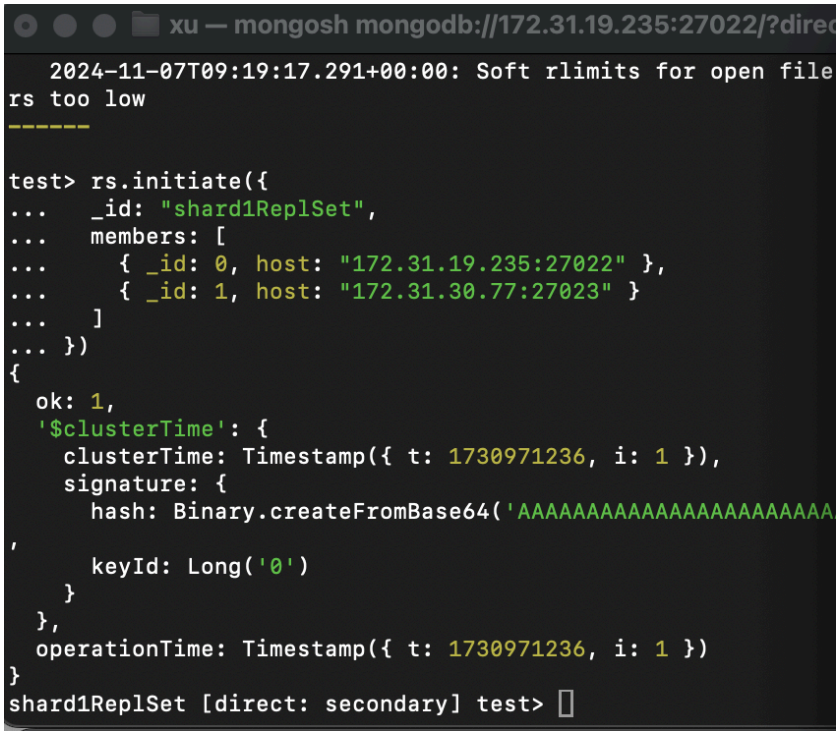
7. (5 points) Connect mongos to each config server.

```
ubuntu@ip-172-31-92-46:~$ sudo mongos --configdb configReplSet/172.31.
92.46:27019,172.31.92.46:27020,172.31.92.46:27021 --bind_ip 0.0.0.0 --
port 27017 --fork --logpath /var/log/mongodb/mongos.log
about to fork child process, waiting until server is ready for connect
ions.
forked process: 3903
child process started successfully, parent exiting
ubuntu@ip-172-31-92-46:~$ mongosh --port 27017
```

```
[direct: mongos] test> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[]
---
active mongoses
[]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently running': 'no',
  'Currently enabled': 'yes',
  'Failed balancer rounds in last 5 attempts': 0,
  'Migration Results for the last 24 hours': 'No recent migrations'
}
---
shardedDataDistribution
[]
---
databases
[
  {
    database: { _id: 'config', primary: 'config', partitioned: true },
    collections: {}
  }
]
[direct: mongos] test>
```

8. (10 points)Set up and launch each of the three shards. Make sure to include the result of sh.status() before adding the shards, which will be done in the next task.

shard1ReplSet (shard1-node1)



```
shard1ReplSet [direct: primary] test> rs.status();
{
  set: 'shard1ReplSet',
  date: ISODate('2024-11-07T09:25:46.522Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 2,
  writableVotingMembersCount: 2,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1730971537, i: 1 }), t:
Long('1') },
    lastCommittedWallTime: ISODate('2024-11-07T09:25:37.057Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1730971537, i: 1 }
), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1730971537, i: 1 }), t: Long('
1') },
    durableOpTime: { ts: Timestamp({ t: 1730971537, i: 1 }), t: Long('
1') },
    lastAppliedWallTime: ISODate('2024-11-07T09:25:37.057Z'),
    lastDurableWallTime: ISODate('2024-11-07T09:25:37.057Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1730971527, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-11-07T09:20:46.962Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1730971236, i:
 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1730971236, i: 1 })
, t: Long('-1') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2024-11-07T09:20:47.020Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-11-07T09:20:47.600Z'
)
  },
  members: [
    {
      _id: 0,
      name: '172.31.19.235:27022',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 390,
      optime: { ts: Timestamp({ t: 1730971537, i: 1 }), t: Long('1') }
,
      optimeDate: ISODate('2024-11-07T09:25:37.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T09:25:37.057Z'),
      lastDurableWallTime: ISODate('2024-11-07T09:25:37.057Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1730971246, i: 1 }),
      electionDate: ISODate('2024-11-07T09:20:46.000Z'),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
```

```
      _id: 1,
      name: '172.31.30.77:27023',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 309,
      optime: { ts: Timestamp({ t: 1730971537, i: 1 }), t: Long('1')
,
      optimeDurable: { ts: Timestamp({ t: 1730971537, i: 1 }), t: Lon
('1') },
      optimeDate: ISODate('2024-11-07T09:25:37.000Z'),
      optimeDurableDate: ISODate('2024-11-07T09:25:37.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T09:25:37.057Z'),
      lastDurableWallTime: ISODate('2024-11-07T09:25:37.057Z'),
      lastHeartbeat: ISODate('2024-11-07T09:25:45.163Z'),
      lastHeartbeatRecv: ISODate('2024-11-07T09:25:46.137Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.19.235:27022',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 1
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730971537, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730971537, i: 1 })
}
```

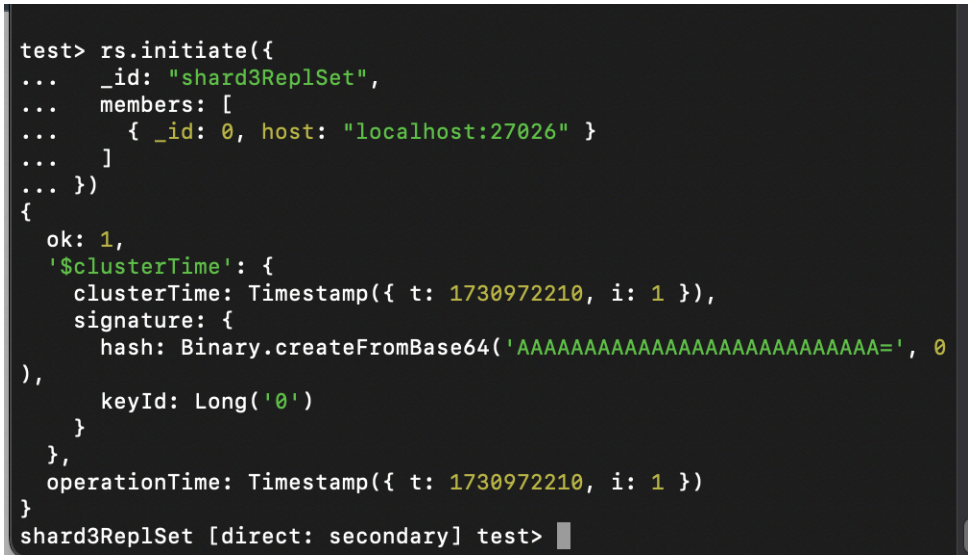
shard2ReplSet (shard2-node1)

```
test> rs.initiate({
...   _id: "shard2ReplSet",
...   members: [
...     { _id: 0, host: "172.31.25.208:27024" },
...     { _id: 1, host: "172.31.28.169:27025" }
...   ]
... })
{ ok: 1 }
shard2ReplSet [direct: other] test>
```

```
shard2ReplSet [direct: other] test> rs.status()
{
  set: 'shard2ReplSet',
  date: ISODate('2024-11-07T09:29:39.647Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 2,
  writableVotingMembersCount: 2,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1730971778, i: 1 }), t:
 Long('1') },
    lastCommittedWallTime: ISODate('2024-11-07T09:29:38.364Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1730971778, i: 1
}), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1730971778, i: 1 }), t: Long(
'1') },
    durableOpTime: { ts: Timestamp({ t: 1730971778, i: 1 }), t: Long(
'1') },
    lastAppliedWallTime: ISODate('2024-11-07T09:29:38.364Z'),
    lastDurableWallTime: ISODate('2024-11-07T09:29:38.364Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1730971748, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-11-07T09:28:28.280Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1730971698, i
: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1730971698, i: 1 }
), t: Long('-1') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2024-11-07T09:28:28.335Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-11-07T09:28:29.820Z
')
  },
  members: [
    {
      _id: 0,
      name: '172.31.25.208:27024',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 581,
      optime: { ts: Timestamp({ t: 1730971778, i: 1 }), t: Long('1')
},
      optimeDate: ISODate('2024-11-07T09:29:38.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T09:29:38.364Z'),
      lastDurableWallTime: ISODate('2024-11-07T09:29:38.364Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1730971708, i: 1 }),
      electionDate: ISODate('2024-11-07T09:28:28.000Z'),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
```

```
      _id: 1,
      name: '172.31.28.169:27025',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 81,
      optime: { ts: Timestamp({ t: 1730971768, i: 1 }), t: Long('1')
},
      optimeDurable: { ts: Timestamp({ t: 1730971768, i: 1 }), t: Lon
g('1') },
      optimeDate: ISODate('2024-11-07T09:29:28.000Z'),
      optimeDurableDate: ISODate('2024-11-07T09:29:28.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T09:29:38.364Z'),
      lastDurableWallTime: ISODate('2024-11-07T09:29:38.364Z'),
      lastHeartbeat: ISODate('2024-11-07T09:29:38.336Z'),
      lastHeartbeatRecv: ISODate('2024-11-07T09:29:37.840Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.25.208:27024',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 1
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730971778, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0
),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730971778, i: 1 })
}
shard2ReplSet [direct: primary] test>
```

shard3ReplSet (Config-server node)



```
shard3ReplSet [direct: secondary] test> rs.status()
{
  set: 'shard3ReplSet',
  date: ISODate('2024-11-07T09:39:10.785Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 1,
  writeMajorityCount: 1,
  votingMembersCount: 1,
  writableVotingMembersCount: 1,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1730972350, i: 1 }), t:
 Long('1') },
    lastCommittedWallTime: ISODate('2024-11-07T09:39:10.370Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1730972350, i: 1
}), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1730972350, i: 1 }), t: Long(
'1') },
    durableOpTime: { ts: Timestamp({ t: 1730972350, i: 1 }), t: Long(
'1') },
    lastAppliedWallTime: ISODate('2024-11-07T09:39:10.370Z'),
    lastDurableWallTime: ISODate('2024-11-07T09:39:10.370Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1730972320, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-11-07T09:36:50.303Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1730972210, i
: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1730972210, i: 1 }
), t: Long('-1') },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    newTermStartDate: ISODate('2024-11-07T09:36:50.346Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-11-07T09:36:50.368Z
')
  },
  members: [
    {
      _id: 0,
      name: 'localhost:27026',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 151,
      optime: { ts: Timestamp({ t: 1730972350, i: 1 }), t: Long('1')
},
      optimeDate: ISODate('2024-11-07T09:39:10.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T09:39:10.370Z'),
      lastDurableWallTime: ISODate('2024-11-07T09:39:10.370Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1730972210, i: 2 }),
      electionDate: ISODate('2024-11-07T09:36:50.000Z'),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    }
  ],
  ok: 1,
  '$clusterTime': {
```

```
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730972350, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0
),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730972350, i: 1 })
}
shard3ReplSet [direct: primary] test>
```

9. (5 points) Add the shards and ensure you include the result of sh.status() after adding them.

shard1ReplSet

```
[direct: mongos] test> sh.addShard("shard1ReplSet/172.31.19.235:27022,
172.31.30.77:27023")
{
  shardAdded: 'shard1ReplSet',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730980480, i: 9 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730980480, i: 9 })
}
[direct: mongos] test>
```

shard2ReplSet

```
}
[direct: mongos] test> sh.addShard("shard2ReplSet/172.31.25.208:27024,
172.31.28.169:27025")
{
  shardAdded: 'shard2ReplSet',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730980567, i: 7 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730980567, i: 7 })
}
[direct: mongos] test>
```

shard3ReplSet

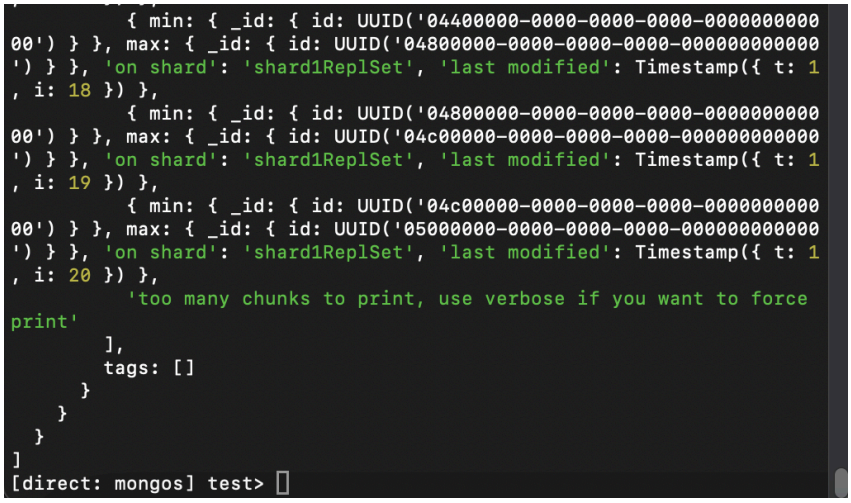
```
[direct: mongos] test> sh.addShard("shard3ReplSet/172.31.92.46:27026")

{
  shardAdded: 'shard3ReplSet',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730981004, i: 6 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730981004, i: 6 })
}
[direct: mongos] test>

[direct: mongos] test>
```

```
[direct: mongos] test> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
  {
    _id: 'shard1ReplSet',
    host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1730980480, i: 6 })
  },
  {
    _id: 'shard2ReplSet',
    host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025',
    state: 1,
    topologyTime: Timestamp({ t: 1730980567, i: 5 })
  },
  {
    _id: 'shard3ReplSet',
    host: 'shard3ReplSet/172.31.92.46:27026',
    state: 1,
    topologyTime: Timestamp({ t: 1730981004, i: 4 })
  }
]
---
active mongoses
[ { '6.0.19': 1 } ]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently running': 'no',
  'Currently enabled': 'yes',
  'Failed balancer rounds in last 5 attempts': 0,
  'Migration Results for the last 24 hours': 'No recent migrations'
}
---
shardedDataDistribution
[
  {
    ns: 'config.system.sessions',
    shards: [
      {
        shardName: 'shard1ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 3,
        ownedSizeBytes: 297,
        orphanedSizeBytes: 0
      }
    ]
  }
]
---
databases
[
  {
    database: { _id: 'config', primary: 'config', partitioned: true },
    collections: {
      'config.system.sessions': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shard1ReplSet', nChunks: 1024 } ],
        chunks: [
```

```
        chunks: [
          { min: { _id: MinKey() }, max: { _id: { id: UUID('00400000-0
000-0000-0000-000000000000') } }, 'on shard': 'shard1ReplSet', 'last m
odified': Timestamp({ t: 1, i: 1 }) },
          { min: { _id: { id: UUID('00400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('00800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 2 }) },
          { min: { _id: { id: UUID('00800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('00c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 3 }) },
          { min: { _id: { id: UUID('00c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 4 }) },
          { min: { _id: { id: UUID('01000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 5 }) },
          { min: { _id: { id: UUID('01400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 6 }) },
          { min: { _id: { id: UUID('01800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 7 }) },
          { min: { _id: { id: UUID('01c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 8 }) },
          { min: { _id: { id: UUID('02000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 9 }) },
          { min: { _id: { id: UUID('02400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 10 }) },
          { min: { _id: { id: UUID('02800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 11 }) },
          { min: { _id: { id: UUID('02c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 12 }) },
          { min: { _id: { id: UUID('03000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 13 }) },
          { min: { _id: { id: UUID('03400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 14 }) },
          { min: { _id: { id: UUID('03800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 15 }) },
          { min: { _id: { id: UUID('03c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('04000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 16 }) },
          { min: { _id: { id: UUID('04000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('04400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 17 }) },
```



10. (5points) Enable the shards and explain the nature of the shard key (ascending, random, or location-based) as well as the sharding strategy (range-based or hash-based) used in your deployment.

MongoDB has various types of shard keys, each influencing how data is distributed across shards:

1.Ascending: Shard keys with ascending values (like timestamps or IDs) will store documents in sequential order across shards. This can lead to uneven distribution if only recent or high values are frequently inserted, as it concentrates load on specific shards.

2. Random: Random shard keys distribute data more evenly, as the insertion pattern is unpredictable. This can help achieve balanced load across shards but may cause fragmentation.

3. Location-based: Location-based shard keys, often based on fields such as region or country, group related data within the same shard. This is useful for geographical data but may lead to imbalances if certain locations have more data than others

Sharding Strategies:

1. Range-based: In range-based sharding, data is distributed according to specified ranges of the shard key. This strategy is efficient for range queries and ordered data access but can lead to unbalanced loads if inserts are concentrated in specific ranges.

2. Hash-based: Hash-based sharding hashes the shard key value to determine the shard, creating a more random distribution. This ensures an even load across shards but may reduce range query efficiency.

.

```
[direct: mongos] test> use testDatabase
switched to db testDatabase
[direct: mongos] testDatabase> db.createCollection("testCollection")
{ ok: 1 }
[direct: mongos] testDatabase> for (let i = 1; i <= 1000; i++) {
...     db.testCollection.insert({ _id: i, name: "User" + i, age: Math
.floor(Math.random() * 50) + 20 })
... }
DeprecationWarning: Collection.insert() is deprecated. Use insertOne,
insertMany, or bulkWrite.
{ acknowledged: true, insertedIds: { '0': 1000 } }
[direct: mongos] testDatabase>
```

```
[direct: mongos] testDatabase> sh.enableSharding("testDatabase")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730981648, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730981648, i: 1 })
}
[direct: mongos] testDatabase>
```

```
[direct: mongos] testDatabase> sh.enableSharding("testDatabase")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730981648, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730981648, i: 1 })
}
[direct: mongos] testDatabase> sh.shardCollection("testDatabase.testCo
llection", { _id: 1 })
{
  collectionsharded: 'testDatabase.testCollection',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730981682, i: 33 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730981682, i: 29 })
}
[direct: mongos] testDatabase>
```


xu — mongosh mongodb://127.0.0.1:27017/?directConnection=tr...
[direct: mongos] test> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
{
_id: 'shard1ReplSet',
host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
state: 1,
topologyTime: Timestamp({ t: 1730980480, i: 6 })
},
{
_id: 'shard2ReplSet',
host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025',
state: 1,
topologyTime: Timestamp({ t: 1730980567, i: 5 })
},
{
_id: 'shard3ReplSet',
host: 'shard3ReplSet/172.31.92.46:27026',
state: 1,
topologyTime: Timestamp({ t: 1730981004, i: 4 })
}
]
---
active mongoses
[{ '6.0.19': 1 }]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
'Currently running': 'no',
'Currently enabled': 'yes',
'Failed balancer rounds in last 5 attempts': 0,
'Migration Results for the last 24 hours': 'No recent migrations'
}
---
shardedDataDistribution
[
{
ns: 'config.system.sessions',
shards: [
{
shardName: 'shard1ReplSet',
numOrphanedDocs: 0,
numOwnedDocuments: 3,
ownedSizeBytes: 297,
orphanedSizeBytes: 0
}
]
}
]
---
databases
[
{
database: { _id: 'config', primary: 'config', partitioned: true },
collections: {
'config.system.sessions': {
shardKey: { _id: 1 },
unique: false,
balancing: true,
chunkMetadata: [{ shard: 'shard1ReplSet', nChunks: 1024 }],

```
        chunks: [
          { min: { _id: MinKey() }, max: { _id: { id: UUID('00400000-0
000-0000-0000-000000000000') } }, 'on shard': 'shard1ReplSet', 'last m
odified': Timestamp({ t: 1, i: 1 }) },
          { min: { _id: { id: UUID('00400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('00800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 2 }) },
          { min: { _id: { id: UUID('00800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('00c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 3 }) },
          { min: { _id: { id: UUID('00c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 4 }) },
          { min: { _id: { id: UUID('01000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 5 }) },
          { min: { _id: { id: UUID('01400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 6 }) },
          { min: { _id: { id: UUID('01800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 7 }) },
          { min: { _id: { id: UUID('01c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 8 }) },
          { min: { _id: { id: UUID('02000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 9 }) },
          { min: { _id: { id: UUID('02400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 10 }) },
          { min: { _id: { id: UUID('02800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 11 }) },
          { min: { _id: { id: UUID('02c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 12 }) },
          { min: { _id: { id: UUID('03000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 13 }) },
          { min: { _id: { id: UUID('03400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 14 }) },
          { min: { _id: { id: UUID('03800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 15 }) },
          { min: { _id: { id: UUID('03c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('04000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 16 }) },
          { min: { _id: { id: UUID('04000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('04400000-0000-0000-0000-000000000000
```

```
        { min: { _id: { id: UUID('00800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('00c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 3 }) },
        { min: { _id: { id: UUID('00c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 4 }) },
        { min: { _id: { id: UUID('01000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 5 }) },
        { min: { _id: { id: UUID('01400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 6 }) },
        { min: { _id: { id: UUID('01800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 7 }) },
        { min: { _id: { id: UUID('01c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 8 }) },
        { min: { _id: { id: UUID('02000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 9 }) },
        { min: { _id: { id: UUID('02400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 10 }) },
        { min: { _id: { id: UUID('02800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('02c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 11 }) },
        { min: { _id: { id: UUID('02c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 12 }) },
        { min: { _id: { id: UUID('03000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 13 }) },
        { min: { _id: { id: UUID('03400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 14 }) },
        { min: { _id: { id: UUID('03800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('03c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 15 }) },
        { min: { _id: { id: UUID('03c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('04000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 16 }) },
        { min: { _id: { id: UUID('04000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('04400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 17 }) },
        { min: { _id: { id: UUID('04400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('04800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 18 }) },
        { min: { _id: { id: UUID('04800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('04c00000-0000-0000-0000-000000000000
```

```
          { min: { _id: { id: UUID('04c00000-0000-0000-0000-000000000000') } }, max: { _id: { id: UUID('05000000-0000-0000-0000-000000000000') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1, i: 20 }) },
          'too many chunks to print, use verbose if you want to force print'
        ],
        tags: []
      }
    }
  }
]
[direct: mongos] test> use testDatabase
switched to db testDatabase
[direct: mongos] testDatabase> db.createCollection("testCollection")
{ ok: 1 }
[direct: mongos] testDatabase> for (let i = 1; i <= 1000; i++) {
...     db.testCollection.insert({ _id: i, name: "User" + i, age: Math.floor(Math.random() * 50) + 20 })
... }
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{ acknowledged: true, insertedIds: { '0': 1000 } }
[direct: mongos] testDatabase> sh.enableSharding("testDatabase")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730981648, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730981648, i: 1 })
}
[direct: mongos] testDatabase> sh.shardCollection("testDatabase.testCollection", { _id: 1 })
{
  collectionsharded: 'testDatabase.testCollection',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730981682, i: 33 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730981682, i: 29 })
}
[direct: mongos] testDatabase> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
  {
    _id: 'shard1ReplSet',
    host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1730980480, i: 6 })
  },
  {
    _id: 'shard2ReplSet',
    host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025'
```

```
    topologyTime: Timestamp({ t: 1730980567, i: 5 })
  },
  {
    _id: 'shard3ReplSet',
    host: 'shard3ReplSet/172.31.92.46:27026',
    state: 1,
    topologyTime: Timestamp({ t: 1730981004, i: 4 })
  }
]
---
active mongoses
[ { '6.0.19': 1 } ]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently running': 'no',
  'Currently enabled': 'yes',
  'Failed balancer rounds in last 5 attempts': 0,
  'Migration Results for the last 24 hours': 'No recent migrations'
}
---
shardedDataDistribution
[
  {
    ns: 'config.system.sessions',
    shards: [
      {
        shardName: 'shard1ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 7,
        ownedSizeBytes: 693,
        orphanedSizeBytes: 0
      }
    ]
  },
  {
    ns: 'testDatabase.testCollection',
    shards: [
      {
        shardName: 'shard3ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 1000,
        ownedSizeBytes: 40000,
        orphanedSizeBytes: 0
      }
    ]
  }
]
---
databases
[
  {
    database: { _id: 'config', primary: 'config', partitioned: true },
    collections: {
      'config.system.sessions': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shard1ReplSet', nChunks: 1024 } ],
        chunks: [
          { min: { _id: MinKey() }, max: { _id: { id: UUID('00400000-0
000-0000-0000-000000000000') } }, 'on shard': 'shard1ReplSet', 'last m
odified': Timestamp({ t: 1, i: 1 }) },
```

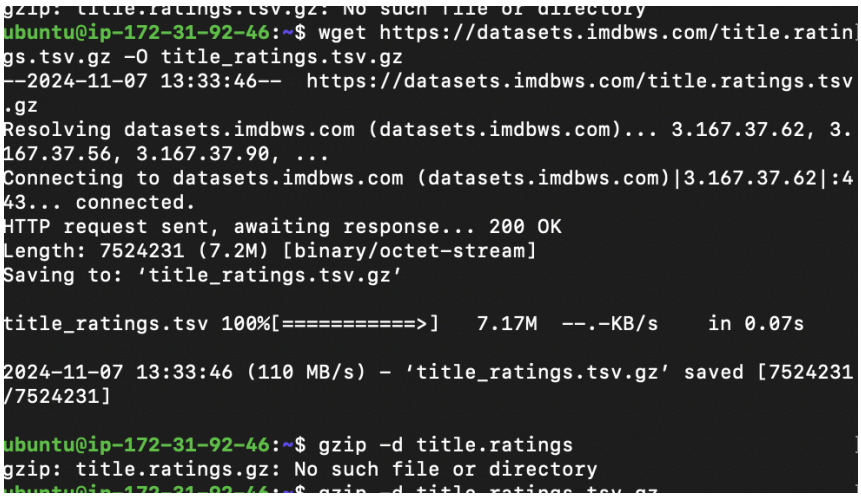
Shard Key and Sharding Strategy Choice

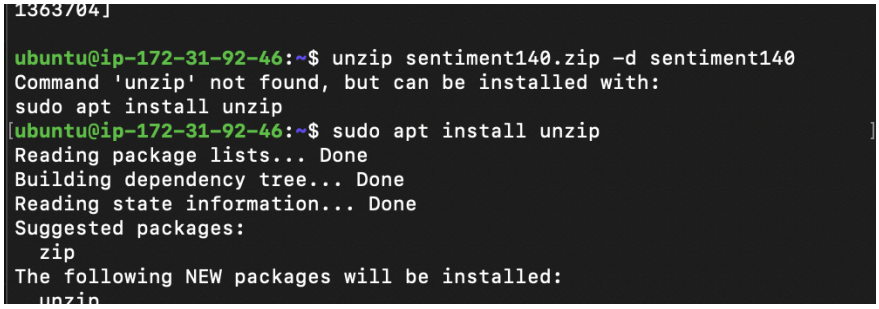
For our deployment, we chose an ascending `_id` field as the shard key with a range-based sharding strategy. This approach suits our application needs, where ordered access to data is preferred, allowing efficient range queries on `_id`. While it could lead to uneven distribution if data is frequently added only at the high end, it aligns well with our intended data structure and access patterns.

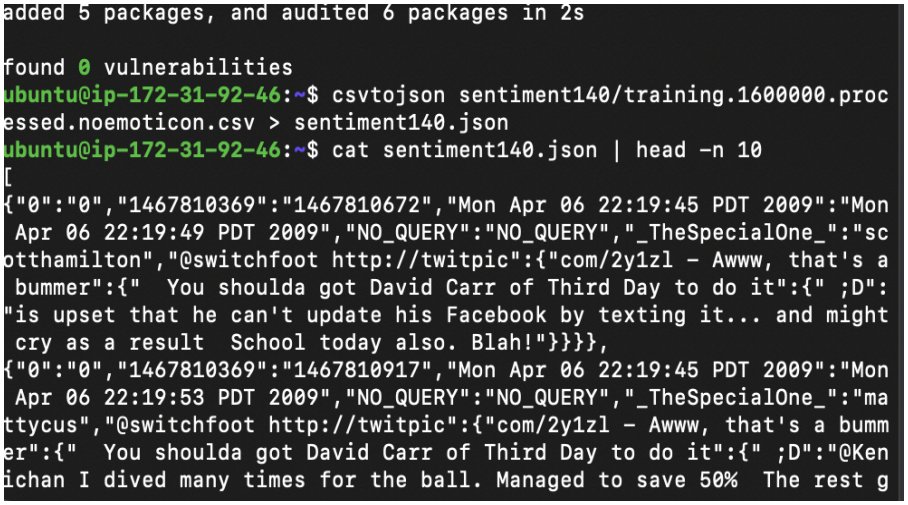
11. (10 points) Populate the cluster with data using a public dataset. Explain your collection and include the code to populate the data, along with the result of sh.status() after the data is populated. Specify the URL for the dataset. (Refer to Task 12 to choose an appropriate dataset for executing the given queries. You are allowed to clean and reduce the public dataset of your choice to populate a reasonable amount of data to be distributed across the shards. You may determine what constitutes a reasonable amount.) You are not permitted to use zips.json provided in the prior assignment. If the public dataset you choose is not large enough, you are allowed to synthesize additional data to scale it.

https://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip -O sentiment140.zip

The collection tweets in twitterDatabase represents a dataset of Twitter posts, sourced from the public sentiment140 dataset. Each document in this collection contains information about a single tweet, such as the tweet content, user information, timestamp, and sentiment score. The key fields include id, text, user, date, and sentiment, which help analyze tweet sentiment and track users' engagement over time.




1363704]
ubuntu@ip-172-31-92-46:~$ unzip sentiment140.zip -d sentiment140
Command 'unzip' not found, but can be installed with:
sudo apt install unzip
ubuntu@ip-172-31-92-46:~$ sudo apt install unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
zip
The following NEW packages will be installed:
unzip


added 5 packages, and audited 6 packages in 2s
found 0 vulnerabilities
ubuntu@ip-172-31-92-46:~$ csvtojson sentiment140/training.1600000.proc
essed.noemoticon.csv > sentiment140.json
ubuntu@ip-172-31-92-46:~$ cat sentiment140.json | head -n 10
[
{"0":"0","1467810369":"1467810672","Mon Apr 06 22:19:45 PDT 2009":"Mon
Apr 06 22:19:49 PDT 2009","NO_QUERY":"NO_QUERY","_TheSpecialOne_":"sc
otthamilton","@switchfoot http://twitpic":{"com/2y1zl - Awww, that's a
bummer":{" You shoulda got David Carr of Third Day to do it":{" ;D":
"is upset that he can't update his Facebook by texting it... and might
cry as a result School today also. Blah!"}}}},
{"0":"0","1467810369":"1467810917","Mon Apr 06 22:19:45 PDT 2009":"Mon
Apr 06 22:19:53 PDT 2009","NO_QUERY":"NO_QUERY","_TheSpecialOne_":"ma
ttycus","@switchfoot http://twitpic":{"com/2y1zl - Awww, that's a bumm
er":{" You shoulda got David Carr of Third Day to do it":{" ;D":"@Ken
ichan I dived many times for the ball. Managed to save 50% The rest g

```
[direct: mongos] test> use twitterDatabase
switched to db twitterDatabase
[direct: mongos] twitterDatabase> db.tweets.countDocuments()
2350999
[direct: mongos] twitterDatabase> db.tweets.findOne()
{
  '0': '0',
  '1467810369': '1467810672',
  _id: ObjectId('672ce1c1a70902e628106d4a'),
  'Mon Apr 06 22:19:45 PDT 2009': 'Mon Apr 06 22:19:49 PDT 2009',
  NO_QUERY: 'NO_QUERY',
  _TheSpecialOne_: 'scotthamilton',
  '@switchfoot http://twitpic': {
    "com/2y1zl - Awww, that's a bummer": {
      '  You shoulda got David Carr of Third Day to do it': {
        ' ;D': "is upset that he can't update his Facebook by texting
it... and might cry as a result  School today also. Blah!"
      }
    }
  }
}
[direct: mongos] twitterDatabase>
```

```
}
[direct: mongos] twitterDatabase> sh.enableSharding("twitterDatabase")

{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730996809, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730996809, i: 1 })
}
[direct: mongos] twitterDatabase> sh.shardCollection("twitterDatabase.]
tweets", { _id: 1 })
{
  collectionsharded: 'twitterDatabase.tweets',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730996821, i: 29 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730996821, i: 25 })
}
```


xd — mongosh mongodb://127.0.0.1:27017/?directConnection=tr...
[direct: mongos] twitterDatabase> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
{
_id: 'shard1ReplSet',
host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
state: 1,
topologyTime: Timestamp({ t: 1730980480, i: 6 })
},
{
_id: 'shard2ReplSet',
host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025',
state: 1,
topologyTime: Timestamp({ t: 1730980567, i: 5 })
},
{
_id: 'shard3ReplSet',
host: 'shard3ReplSet/172.31.92.46:27026',
state: 1,
topologyTime: Timestamp({ t: 1730981004, i: 4 })
}
]
---
active mongoses
[{ '6.0.19': 1 }]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
'Currently running': 'no',
'Currently enabled': 'yes',
'Failed balancer rounds in last 5 attempts': 0,
'Migration Results for the last 24 hours': { '3': 'Success' }
}
---
shardedDataDistribution
[
{
ns: 'testDatabase.testCollection',
shards: [
{
shardName: 'shard3ReplSet',
numOrphanedDocs: 0,
numOwnedDocuments: 1000,
ownedSizeBytes: 40000,
orphanedSizeBytes: 0
}
]
},
{
ns: 'config.system.sessions',
shards: [
{
shardName: 'shard1ReplSet',
numOrphanedDocs: 0,
numOwnedDocuments: 11,
ownedSizeBytes: 1089,
orphanedSizeBytes: 0
}
]
},
child process started successfully, parent exiting
ubuntu@ip-172-31-28-169:~$

```
  {
    ns: 'twitterDatabase.tweets',
    shards: [
      {
        shardName: 'shard2ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 688294,
        ownedSizeBytes: 268434660,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard1ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 344147,
        ownedSizeBytes: 134217330,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard3ReplSet',
        numOrphanedDocs: 1032441,
        numOwnedDocuments: 1318558,
        ownedSizeBytes: 514237620,
        orphanedSizeBytes: 402651990
      }
    ]
  }
]
---
databases
[
  {
    database: { _id: 'config', primary: 'config', partitioned: true },
    collections: {
      'config.system.sessions': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shard1ReplSet', nChunks: 1024 } ],
        chunks: [
          { min: { _id: MinKey() }, max: { _id: { id: UUID('00400000-0
000-0000-0000-000000000000') } }, 'on shard': 'shard1ReplSet', 'last m
odified': Timestamp({ t: 1, i: 1 }) },
          { min: { _id: { id: UUID('00400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('00800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 2 }) },
          { min: { _id: { id: UUID('00800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('00c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 3 }) },
          { min: { _id: { id: UUID('00c00000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01000000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 4 }) },
          { min: { _id: { id: UUID('01000000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01400000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 5 }) },
          { min: { _id: { id: UUID('01400000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01800000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 6 }) },
          { min: { _id: { id: UUID('01800000-0000-0000-0000-0000000000
00') } }, max: { _id: { id: UUID('01c00000-0000-0000-0000-000000000000
') } }, 'on shard': 'shard1ReplSet', 'last modified': Timestamp({ t: 1
, i: 7 }) },
```

The output from sh.status() shows that the tweets collection in twitterDatabase is successfully sharded across the three replica sets: shard1ReplSet, shard2ReplSet, and shard3ReplSet.

12. (12 points) Generate the following queries for the populated data. For each query, show its execution time and also show which shard served the query.

1. **A range query to find documents in a given range.**
2. **A query involving $elemMatch involving at least two conditions.**
3. **A query involving $in, $nin, or $all**
4. **A query involving aggregate()**
5. **A update**
6. **A delete**

A range query to find documents in a given range.

```
[direct: mongos] twitterDatabase> var start = new Date();

[direct: mongos] twitterDatabase> db.tweets.find({ _id: { $gte: Object
Id("672ce1d2a70902e62815ad9d"), $lt: ObjectId("672ce1e4a70902e6281aedf
0") } });
[
  {
    '0': '0',
    '1467810369': '2015609629',
    _id: ObjectId('672ce1d2a70902e62815ad9d'),
    'Mon Apr 06 22:19:45 PDT 2009': 'Wed Jun 03 05:06:00 PDT 2009',
    NO_QUERY: 'NO_QUERY',
    _TheSpecialOne_: 'danrowlands',
    '@switchfoot http://twitpic': {
      "com/2y1zl - Awww, that's a bummer": {
        '  You shoulda got David Carr of Third Day to do it': {
          ' ;D': 'Cannot find SS.INI file for user Dan. I hate Souce S
afe '
        }
      }
    }
  },
  {
    '0': '0',
    '1467810369': '2015609666',
    _id: ObjectId('672ce1d2a70902e62815ad9e'),
    'Mon Apr 06 22:19:45 PDT 2009': 'Wed Jun 03 05:06:00 PDT 2009',
    NO_QUERY: 'NO_QUERY',
    _TheSpecialOne_: 'kyelani',
    '@switchfoot http://twitpic': {
      "com/2y1zl - Awww, that's a bummer": {
        '  You shoulda got David Carr of Third Day to do it': {
          ' ;D': '@twosoups *sob* can u see my reply to @CatatonicCat?
 fucking @reply update '
        }
      }
    }
  },
  {
    '0': '0',
    '1467810369': '2015609768',
    _id: ObjectId('672ce1d2a70902e62815ad9f'),
    'Mon Apr 06 22:19:45 PDT 2009': 'Wed Jun 03 05:06:01 PDT 2009',
    NO_QUERY: 'NO_QUERY',
    _TheSpecialOne_: 'HumanGummybear',
    '@switchfoot http://twitpic': {
      "com/2y1zl - Awww, that's a bummer": {
        '  You shoulda got David Carr of Third Day to do it': {
          ' ;D': " I think I'm gonna die if I see another Smurf. Also
if I don't speak with someone my age  HELP."
        }
      }
```

```
    _TheSpecialOne_: 'seanvstherobots',
    '@switchfoot http://twitpic': {
      "com/2y1zl - Awww, that's a bummer": {
        '  You shoulda got David Carr of Third Day to do it': { ' ;D':
 'Hang out with people smarter than you... not me ' }
      }
    }
  },
  {
    '0': '0',
    '1467810369': '2015613081',
    _id: ObjectId('672ce1d2a70902e62815adae'),
    'Mon Apr 06 22:19:45 PDT 2009': 'Wed Jun 03 05:06:28 PDT 2009',
    NO_QUERY: 'NO_QUERY',
    _TheSpecialOne_: 'juanbarnard',
    '@switchfoot http://twitpic': {
      "com/2y1zl - Awww, that's a bummer": {
        '  You shoulda got David Carr of Third Day to do it': {
          ' ;D': '#Snaptu just went down for #maintenance. I hate twee
ting from m.twitter.com! '
        }
      }
    }
  },
  {
    '0': '0',
    '1467810369': '2015613119',
    _id: ObjectId('672ce1d2a70902e62815adaf'),
    'Mon Apr 06 22:19:45 PDT 2009': 'Wed Jun 03 05:06:28 PDT 2009',
    NO_QUERY: 'NO_QUERY',
    _TheSpecialOne_: 'owlonthesill',
    '@switchfoot http://twitpic': {
      "com/2y1zl - Awww, that's a bummer": {
        '  You shoulda got David Carr of Third Day to do it': {
          ' ;D': "@IslaLuna lol, well I've done that a few times mysel
f. But today it's oatmeal. I do TRY to be good, but it never works out
.  No willpower!"
        }
      }
    }
  },
  {
    '0': '0',
    '1467810369': '2015613409',
    _id: ObjectId('672ce1d2a70902e62815adb0'),
    'Mon Apr 06 22:19:45 PDT 2009': 'Wed Jun 03 05:06:31 PDT 2009',
    NO_QUERY: 'NO_QUERY',
    _TheSpecialOne_: 'gina_pina_14',
    '@switchfoot http://twitpic': {
      "com/2y1zl - Awww, that's a bummer": {
        '  You shoulda got David Carr of Third Day to do it': {
          ' ;D': "@lilmissfootyfan Plenty. I'm in heaven. I wish they
were in motion pics though, I miss seeing him play"
        }
      }
    }
  }
]
Type "it" for more
[direct: mongos] twitterDatabase> var end = new Date();

[direct: mongos] twitterDatabase> print("Execution Time: " + (end - st
art) + "ms");
Execution Time: 213ms

[direct: mongos] twitterDatabase>
```

A query involving $elemMatch involving at least two conditions.

```
[direct: mongos] twitterDatabase> var start = new Date();

[direct: mongos] twitterDatabase> db.tweets.find({ comments: { $elemMa
tch: { author: "JohnDoe", content: /MongoDB/i } } });

[direct: mongos] twitterDatabase> var end = new Date();

[direct: mongos] twitterDatabase> print("Execution Time: " + (end - st
art) + "ms");
Execution Time: 3563ms

[direct: mongos] twitterDatabase> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
  {
    _id: 'shard1ReplSet',
    host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1730980480, i: 6 })
  },
  {
    _id: 'shard2ReplSet',
    host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025',
    state: 1,
    topologyTime: Timestamp({ t: 1730980567, i: 5 })
  },
  {
    _id: 'shard3ReplSet',
    host: 'shard3ReplSet/172.31.92.46:27026',
    state: 1,
    topologyTime: Timestamp({ t: 1730981004, i: 4 })
  }
]
---
```

```
  },
  {
    ns: 'twitterDatabase.tweets',
    shards: [
      {
        shardName: 'shard2ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 688294,
        ownedSizeBytes: 268434660,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard1ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 344147,
        ownedSizeBytes: 134217330,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard3ReplSet',
        numOrphanedDocs: 1032441,
        numOwnedDocuments: 1318558,
        ownedSizeBytes: 514237620,
        orphanedSizeBytes: 402651990
      }
    ]
  },
```

```
  },
  collections: {
    'twitterDatabase.tweets': {
      shardKey: { _id: 1 },
      unique: false,
      balancing: true,
      chunkMetadata: [
        { shard: 'shard1ReplSet', nChunks: 1 },
        { shard: 'shard2ReplSet', nChunks: 2 },
        { shard: 'shard3ReplSet', nChunks: 1 }
      ],
      chunks: [
```

A query involving $in, $nin, or $all

```
]
[direct: mongos] twitterDatabase> var start = new Date();

[direct: mongos] twitterDatabase> db.tweets.find({ sentiment: { $in: [
"positive", "negative"] } });

[direct: mongos] twitterDatabase> var end = new Date();

[direct: mongos] twitterDatabase> print("Execution Time: " + (end - st
art) + "ms");
Execution Time: 2152ms

[direct: mongos] twitterDatabase> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
  {
    _id: 'shard1ReplSet',
    host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1730980480, i: 6 })
  },
  {
    _id: 'shard2ReplSet',
    host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025',
    state: 1,
    topologyTime: Timestamp({ t: 1730980567, i: 5 })
  },
  {
    _id: 'shard3ReplSet',
    host: 'shard3ReplSet/172.31.92.46:27026',
    state: 1,
    topologyTime: Timestamp({ t: 1730981004, i: 4 })
  }
]
---
```

```
  },
  {
    ns: 'twitterDatabase.tweets',
    shards: [
      {
        shardName: 'shard2ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 688294,
        ownedSizeBytes: 268434660,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard1ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 344147,
        ownedSizeBytes: 134217330,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard3ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 1318558,
        ownedSizeBytes: 514237620,
        orphanedSizeBytes: 0
      }
    ]
  },
```

A query involving aggregate()

```
[direct: mongos] twitterDatabase> var start = new Date();

[direct: mongos] twitterDatabase> db.tweets.aggregate([{ $group: { _id
: "$sentiment", count: { $sum: 1 } } }]);
[ { _id: null, count: 2350999 } ]
[direct: mongos] twitterDatabase> var end = new Date();

[direct: mongos] twitterDatabase> print("Execution Time: " + (end - st
art) + "ms");
Execution Time: 2411ms

[direct: mongos] twitterDatabase> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
  {
    _id: 'shard1ReplSet',
    host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1730980480, i: 6 })
  },
  {
    _id: 'shard2ReplSet',
    host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025',
    state: 1,
    topologyTime: Timestamp({ t: 1730980567, i: 5 })
  },
  {
    _id: 'shard3ReplSet',
    host: 'shard3ReplSet/172.31.92.46:27026',
    state: 1,
    topologyTime: Timestamp({ t: 1730981004, i: 4 })
  }
]
```


shardedDataDistribution
[
{
ns: 'twitterDatabase.tweets',
shards: [
{
shardName: 'shard2ReplSet',
numOrphanedDocs: 0,
numOwnedDocuments: 688294,
ownedSizeBytes: 268434660,
orphanedSizeBytes: 0
},
{
shardName: 'shard1ReplSet',
numOrphanedDocs: 0,
numOwnedDocuments: 344147,
ownedSizeBytes: 134217330,
orphanedSizeBytes: 0
},
{
shardName: 'shard3ReplSet',
numOrphanedDocs: 0,
numOwnedDocuments: 1318558,
ownedSizeBytes: 514237620,
orphanedSizeBytes: 0
}
]
},

A update

```
]
[direct: mongos] twitterDatabase> var start = new Date();

[direct: mongos] twitterDatabase> db.tweets.updateMany({ sentiment: "n
eutral" }, { $set: { sentiment: "mixed" } });

{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
[direct: mongos] twitterDatabase> var end = new Date();

[direct: mongos] twitterDatabase> print("Execution Time: " + (end - st
art) + "ms");
Execution Time: 1129ms

[direct: mongos] twitterDatabase> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
  {
    _id: 'shard1ReplSet',
    host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1730980480, i: 6 })
  },
  {
    _id: 'shard2ReplSet',
    host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025',
    state: 1,
    topologyTime: Timestamp({ t: 1730980567, i: 5 })
  },
  {
    _id: 'shard3ReplSet',
    host: 'shard3ReplSet/172.31.92.46:27026',
    state: 1,
    topologyTime: Timestamp({ t: 1730981004, i: 4 })
  }
]
```

```
  },
  {
    ns: 'twitterDatabase.tweets',
    shards: [
      {
        shardName: 'shard2ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 688293,
        ownedSizeBytes: 268434270,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard1ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 344147,
        ownedSizeBytes: 134217330,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard3ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 1318558,
        ownedSizeBytes: 514237620,
        orphanedSizeBytes: 0
      }
    ]
  }
```

A delete

```
]
[direct: mongos] twitterDatabase> var start = new Date();

[direct: mongos] twitterDatabase> db.tweets.deleteMany({ sentiment: "m
ixed" });

{ acknowledged: true, deletedCount: 0 }
[direct: mongos] twitterDatabase> var end = new Date();

[direct: mongos] twitterDatabase> print("Execution Time: " + (end - st
art) + "ms");
Execution Time: 1110ms

[direct: mongos] twitterDatabase> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('672ca3ee17e23023753a5469') }
---
shards
[
  {
    _id: 'shard1ReplSet',
    host: 'shard1ReplSet/172.31.19.235:27022,172.31.30.77:27023',
    state: 1,
    topologyTime: Timestamp({ t: 1730980480, i: 6 })
  },
  {
    _id: 'shard2ReplSet',
    host: 'shard2ReplSet/172.31.25.208:27024,172.31.28.169:27025',
    state: 1,
    topologyTime: Timestamp({ t: 1730980567, i: 5 })
  },
  {
    _id: 'shard3ReplSet',
    host: 'shard3ReplSet/172.31.92.46:27026',
    state: 1,
    topologyTime: Timestamp({ t: 1730981004, i: 4 })
  }
]
---
```

```
  {
    ns: 'twitterDatabase.tweets',
    shards: [
      {
        shardName: 'shard2ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 688293,
        ownedSizeBytes: 268434270,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard3ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 1318558,
        ownedSizeBytes: 514237620,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'shard1ReplSet',
        numOrphanedDocs: 0,
        numOwnedDocuments: 344147,
        ownedSizeBytes: 134217330,
        orphanedSizeBytes: 0
      }
    ]
```

13. (5 points) Show shards are replicated using rs.status().

configReplSet

```
configReplSet [direct: primary] test> rs.status()
{
  set: 'configReplSet',
  date: ISODate('2024-11-07T17:14:36.082Z'),
  myState: 1,
  term: Long('3'),
  syncSourceHost: '',
  syncSourceId: -1,
  configsvr: true,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1730999675, i: 3 }), t: Lo
ng('3') },
    lastCommittedWallTime: ISODate('2024-11-07T17:14:35.391Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1730999675, i: 3 }),
 t: Long('3') },
    appliedOpTime: { ts: Timestamp({ t: 1730999675, i: 3 }), t: Long('3'
) },
    durableOpTime: { ts: Timestamp({ t: 1730999675, i: 3 }), t: Long('3'
) },
    lastAppliedWallTime: ISODate('2024-11-07T17:14:35.391Z'),
    lastDurableWallTime: ISODate('2024-11-07T17:14:35.391Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1730999617, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-11-07T15:59:46.747Z'),
    electionTerm: Long('3'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 0, i: 0 }), t: L
ong('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1730994663, i: 10169
}), t: Long('2') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2024-11-07T15:59:46.754Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-11-07T15:59:47.292Z')
  },
  members: [
    {
      _id: 0,
      name: '172.31.92.46:27019',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 4503,
      optime: { ts: Timestamp({ t: 1730999675, i: 3 }), t: Long('3') },
      optimeDate: ISODate('2024-11-07T17:14:35.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T17:14:35.391Z'),
      lastDurableWallTime: ISODate('2024-11-07T17:14:35.391Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1730995186, i: 1 }),
      electionDate: ISODate('2024-11-07T15:59:46.000Z'),
      configVersion: 1,
      configTerm: 3,
      self: true,
      lastHeartbeatMessage: ''
    }
```

```
    },
    {
      _id: 1,
      name: '172.31.92.46:27020',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 4493,
      optime: { ts: Timestamp({ t: 1730999674, i: 1 }), t: Long('3') },
      optimeDurable: { ts: Timestamp({ t: 1730999674, i: 1 }), t: Long('
3') },
      optimeDate: ISODate('2024-11-07T17:14:34.000Z'),
      optimeDurableDate: ISODate('2024-11-07T17:14:34.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T17:14:35.391Z'),
      lastDurableWallTime: ISODate('2024-11-07T17:14:35.391Z'),
      lastHeartbeat: ISODate('2024-11-07T17:14:35.056Z'),
      lastHeartbeatRecv: ISODate('2024-11-07T17:14:35.965Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.92.46:27019',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 3
    },
    {
      _id: 2,
      name: '172.31.92.46:27021',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 4483,
      optime: { ts: Timestamp({ t: 1730999674, i: 1 }), t: Long('3') },
      optimeDurable: { ts: Timestamp({ t: 1730999674, i: 1 }), t: Long('
3') },
      optimeDate: ISODate('2024-11-07T17:14:34.000Z'),
      optimeDurableDate: ISODate('2024-11-07T17:14:34.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T17:14:35.391Z'),
      lastDurableWallTime: ISODate('2024-11-07T17:14:35.391Z'),
      lastHeartbeat: ISODate('2024-11-07T17:14:35.054Z'),
      lastHeartbeatRecv: ISODate('2024-11-07T17:14:34.327Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.92.46:27020',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 1,
      configTerm: 3
    }
  ],
  ok: 1,
  lastCommittedOpTime: Timestamp({ t: 1730999675, i: 3 }),
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730999675, i: 3 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730999675, i: 3 })
}
configReplSet [direct: primary] test>
```

shard1ReplSet

```
[shard1ReplSet [direct: primary] test> rs.status()                    ]
{
  set: 'shard1ReplSet',
  date: ISODate('2024-11-07T17:17:32.147Z'),
  myState: 1,
  term: Long('7'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 2,
  writableVotingMembersCount: 2,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1730999848, i: 1 }), t:
Long('7') },
    lastCommittedWallTime: ISODate('2024-11-07T17:17:28.696Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1730999848, i: 1 }
), t: Long('7') },
    appliedOpTime: { ts: Timestamp({ t: 1730999848, i: 1 }), t: Long('
7') },
    durableOpTime: { ts: Timestamp({ t: 1730999848, i: 1 }), t: Long('
7') },
    lastAppliedWallTime: ISODate('2024-11-07T17:17:28.696Z'),
    lastDurableWallTime: ISODate('2024-11-07T17:17:28.696Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1730999828, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-11-07T11:51:47.868Z'),
    electionTerm: Long('7'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1730980222, i:
 1 }), t: Long('5') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1730980222, i: 1 })
, t: Long('5') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2024-11-07T11:51:47.878Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-11-07T11:51:47.974Z'
)
  },
  members: [
    {
      _id: 0,
      name: '172.31.19.235:27022',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 19588,
      optime: { ts: Timestamp({ t: 1730999848, i: 1 }), t: Long('7') }
,
      optimeDate: ISODate('2024-11-07T17:17:28.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T17:17:28.696Z'),
      lastDurableWallTime: ISODate('2024-11-07T17:17:28.696Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1730980307, i: 1 }),
      electionDate: ISODate('2024-11-07T11:51:47.000Z'),
      configVersion: 1,
      configTerm: 7,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
```

```
    {
      _id: 1,
      name: '172.31.30.77:27023',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 19553,
      optime: { ts: Timestamp({ t: 1730999848, i: 1 }), t: Long('7') }
,
      optimeDurable: { ts: Timestamp({ t: 1730999848, i: 1 }), t: Long
('7') },
      optimeDate: ISODate('2024-11-07T17:17:28.000Z'),
      optimeDurableDate: ISODate('2024-11-07T17:17:28.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T17:17:28.696Z'),
      lastDurableWallTime: ISODate('2024-11-07T17:17:28.696Z'),
      lastHeartbeat: ISODate('2024-11-07T17:17:31.371Z'),
      lastHeartbeatRecv: ISODate('2024-11-07T17:17:31.778Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: '172.31.19.235:27022',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 7
    }
  ],
  ok: 1,
  lastCommittedOpTime: Timestamp({ t: 1730999848, i: 1 }),
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1730999848, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0)
,
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1730999848, i: 1 })
}
shard1ReplSet [direct: primary] test>
```

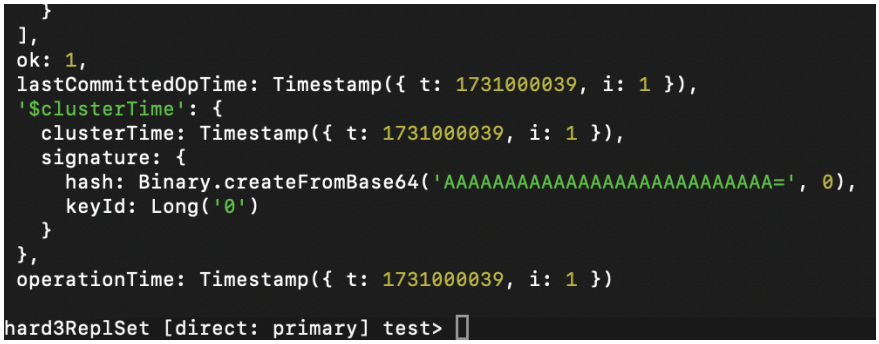
shard2ReplSet

```
shard2ReplSet [direct: primary] test> rs.status()
{
  set: 'shard2ReplSet',
  date: ISODate('2024-11-07T17:19:14.141Z'),
  myState: 1,
  term: Long('7'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 2,
  writableVotingMembersCount: 2,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1730999944, i: 1 }), t:
 Long('7') },
    lastCommittedWallTime: ISODate('2024-11-07T17:19:04.921Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1730999944, i: 1
}), t: Long('7') },
    appliedOpTime: { ts: Timestamp({ t: 1730999944, i: 1 }), t: Long(
'7') },
    durableOpTime: { ts: Timestamp({ t: 1730999944, i: 1 }), t: Long(
'7') },
    lastAppliedWallTime: ISODate('2024-11-07T17:19:04.921Z'),
    lastDurableWallTime: ISODate('2024-11-07T17:19:04.921Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1730999904, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-11-07T11:52:44.164Z'),
    electionTerm: Long('7'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1730980307, i
: 1 }), t: Long('5') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1730980307, i: 1 }
), t: Long('5') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2024-11-07T11:52:44.172Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-11-07T11:52:44.397Z
')
  },
  members: [
    {
      _id: 0,
      name: '172.31.25.208:27024',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 19624,
      optime: { ts: Timestamp({ t: 1730999944, i: 1 }), t: Long('7')
},
      optimeDate: ISODate('2024-11-07T17:19:04.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T17:19:04.921Z'),
      lastDurableWallTime: ISODate('2024-11-07T17:19:04.921Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1730980364, i: 1 }),
      electionDate: ISODate('2024-11-07T11:52:44.000Z'),
      configVersion: 1,
      configTerm: 7,
      self: true,
      lastHeartbeatMessage: ''
    },
```


lastHeartbeatMessage: ''
},
{
_id: 1,
name: '172.31.28.169:27025',
health: 1,
state: 2,
stateStr: 'SECONDARY',
uptime: 19595,
optime: { ts: Timestamp({ t: 1730999944, i: 1 }), t: Long('7')
},
optimeDurable: { ts: Timestamp({ t: 1730999944, i: 1 }), t: Lon
g('7') },
optimeDate: ISODate('2024-11-07T17:19:04.000Z'),
optimeDurableDate: ISODate('2024-11-07T17:19:04.000Z'),
lastAppliedWallTime: ISODate('2024-11-07T17:19:04.921Z'),
lastDurableWallTime: ISODate('2024-11-07T17:19:04.921Z'),
lastHeartbeat: ISODate('2024-11-07T17:19:13.521Z'),
lastHeartbeatRecv: ISODate('2024-11-07T17:19:13.520Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: '172.31.25.208:27024',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 7
}
],
ok: 1,
lastCommittedOpTime: Timestamp({ t: 1730999944, i: 1 }),
'$clusterTime': {
clusterTime: Timestamp({ t: 1730999953, i: 1 }),
signature: {
hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0
),
keyId: Long('0')
}
},
operationTime: Timestamp({ t: 1730999944, i: 1 })
}

shard3ReplSet

```
shard3ReplSet [direct: primary] test> rs.status()
{
  set: 'shard3ReplSet',
  date: ISODate('2024-11-07T17:20:40.804Z'),
  myState: 1,
  term: Long('2'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 1,
  writeMajorityCount: 1,
  votingMembersCount: 1,
  writableVotingMembersCount: 1,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1731000039, i: 1 }), t: Lo
ng('2') },
    lastCommittedWallTime: ISODate('2024-11-07T17:20:39.140Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1731000039, i: 1 }),
 t: Long('2') },
    appliedOpTime: { ts: Timestamp({ t: 1731000039, i: 1 }), t: Long('2'
) },
    durableOpTime: { ts: Timestamp({ t: 1731000039, i: 1 }), t: Long('2'
) },
    lastAppliedWallTime: ISODate('2024-11-07T17:20:39.140Z'),
    lastDurableWallTime: ISODate('2024-11-07T17:20:39.140Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1730999999, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-11-07T16:23:09.022Z'),
    electionTerm: Long('2'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 0, i: 0 }), t: L
ong('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1730994663, i: 10168
}), t: Long('1') },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    newTermStartDate: ISODate('2024-11-07T16:23:09.028Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-11-07T16:23:09.038Z')
  },
  members: [
    {
      _id: 0,
      name: '172.31.92.46:27026',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 3465,
      optime: { ts: Timestamp({ t: 1731000039, i: 1 }), t: Long('2') },
      optimeDate: ISODate('2024-11-07T17:20:39.000Z'),
      lastAppliedWallTime: ISODate('2024-11-07T17:20:39.140Z'),
      lastDurableWallTime: ISODate('2024-11-07T17:20:39.140Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1730996589, i: 1 }),
      electionDate: ISODate('2024-11-07T16:23:09.000Z'),
      configVersion: 1,
      configTerm: 2,
      self: true,
      lastHeartbeatMessage: ''
    }
```



14.(5 points) List of hosts and describe what are deployed in each host. For example,
node0:port# config server PRIMARY
node1:port# config server SECONDARY

Config Server Replica Set (all deployed on a single instance with three simulated members)

- Instance: 172.31.92.46 (config-server node)
  - `172.31.92.46:27019` - Config Server PRIMARY
  - `172.31.92.46:27020` - Config Server SECONDARY
  - `172.31.92.46:27021` - Config Server SECONDARY
  - `172.31.92.46:27017` - Mongos Router

Shard1 Replica Set

- Instance: 172.31.19.235 (shard1-node1)
  - `172.31.19.235:27022` - Shard1 PRIMARY
- Instance: 172.31.30.77 (shard1-node2)
  - `172.31.30.77:27023` - Shard1 SECONDARY

Shard2 Replica Set

- Instance: 172.31.25.208 (shard2-node1)
  - `172.31.25.208:27024` - Shard2 PRIMARY
- Instance: 172.31.28.169 (shard2-node2)
  - `172.31.28.169:27025` - Shard2 SECONDARY

Shard3 Replica Set (single-node setup on the config server instance)

- Instance: 172.31.92.46
  - `172.31.92.46:27026` - Shard3 PRIMARY (Single node, no SECONDARY)

The Config Server is deployed on a single instance, simulating three members by using separate ports.
Each Shard (Shard1 and Shard2) has two nodes on different AWS instances, with a PRIMARY and a SECONDARY node.
Shard3 is configured as a single-node replica set on the same instance as the Config Server, using an additional port.