Exercise:Advance Mongodb

1.Start MongoDB without authentication

2.Create Database ttn and insert some data into user collection.

3.Setup the replication with Artiber and verify that ttn database is replicated into secondary.

4.Add 2 more members to a replica set with

5.rs.add();

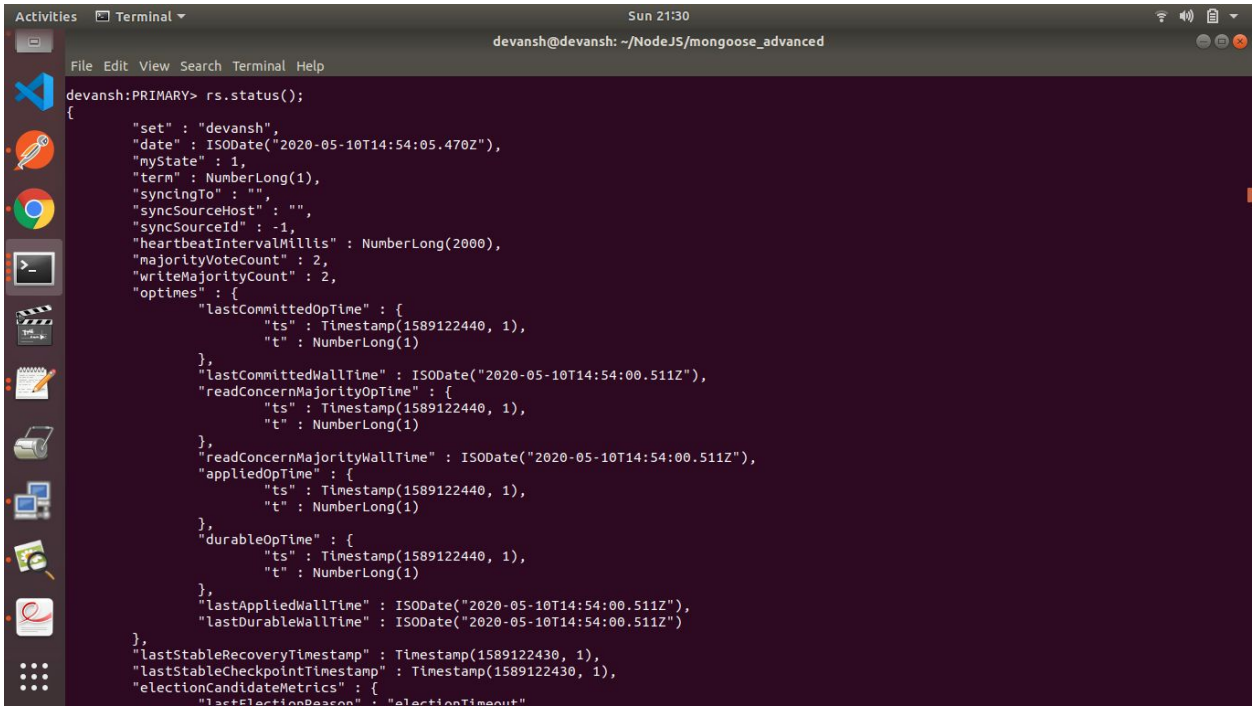
6.Try adding a member which already have data.

First of all i will start three instance of mongo on three different port .

```
devansh@devansh:~/NodeJS/mongoose_advanced$ mkdir -p rs1 rs2 rs3 rs4 rs5 rs6
devansh@devansh:~/NodeJS/mongoose_advanced$ mongod --replSet devansh --logpath "1.log" --dbpath rs1 --port 27017 &
[1] 26776
devansh@devansh:~/NodeJS/mongoose_advanced$ mongod --replSet devansh --logpath "2.log" --dbpath rs2 --port 27018 &
[2] 26777
devansh@devansh:~/NodeJS/mongoose_advanced$ mongod --replSet devansh --logpath "3.log" --dbpath rs3 --port 27019 &
[3] 26786
devansh@devansh:~/NodeJS/mongoose_advanced$ ps -ef | grep mongod
devansh  26776 25472  6 20:20 pts/1    00:00:00 mongod --replSet devansh --logpath 1.log --dbpath rs1 --port 27017
devansh  26777 25472  7 20:20 pts/1    00:00:00 mongod --replSet devansh --logpath 2.log --dbpath rs2 --port 27018
devansh  26786 25472  9 20:20 pts/1    00:00:00 mongod --replSet devansh --logpath 3.log --dbpath rs3 --port 27019
devansh  26888 25472  0 20:20 pts/1    00:00:00 grep --color=auto mongod
devansh@devansh:~/NodeJS/mongoose_advanced$ mongo
```

```
> config={
... _id:'devansh',
... members:[
... {_id: 0, host:"localhost:27017"},
... {_id: 1, host:"localhost:27018"},
... {_id: 2, host:"localhost:27019",arbiterOnly:true},
... ],
... }
{
        "_id" : "devansh",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:27017"
                },
                {
                        "_id" : 1,
                        "host" : "localhost:27018"
                },
                {
                        "_id" : 2,
                        "host" : "localhost:27019",
                        "arbiterOnly" : true
                }
        ]
}
```

```
> rs.initiate(config);
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1589122298, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1589122298, 1)
}
```

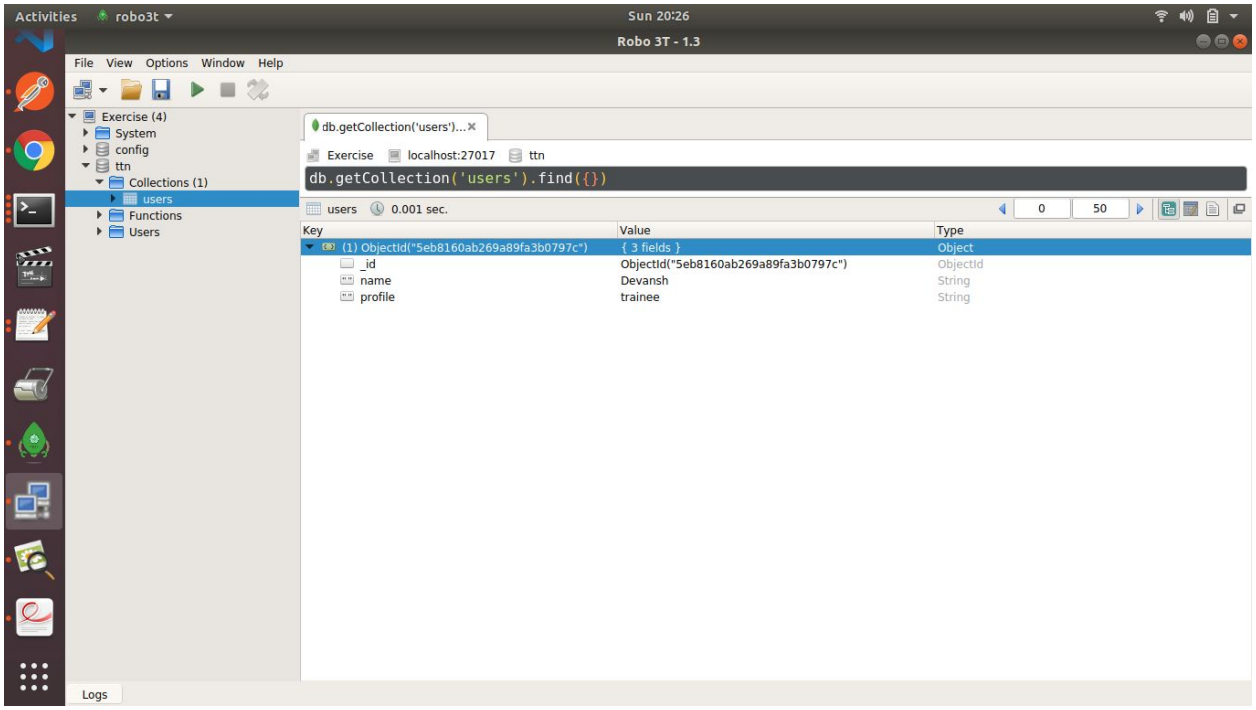



```
devansh:PRIMARY> rs.status();
{
        "set" : "devansh",
        "date" : ISODate("2020-05-10T14:54:05.470Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "heartbeatIntervalMillis" : NumberLong(2000),
        "majorityVoteCount" : 2,
        "writeMajorityCount" : 2,
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1589122440, 1),
                        "t" : NumberLong(1)
                },
                "lastCommittedWallTime" : ISODate("2020-05-10T14:54:00.511Z"),
                "readConcernMajorityOpTime" : {
                        "ts" : Timestamp(1589122440, 1),
                        "t" : NumberLong(1)
                },
                "readConcernMajorityWallTime" : ISODate("2020-05-10T14:54:00.511Z"),
                "appliedOpTime" : {
                        "ts" : Timestamp(1589122440, 1),
                        "t" : NumberLong(1)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1589122440, 1),
                        "t" : NumberLong(1)
                },
                "lastAppliedWallTime" : ISODate("2020-05-10T14:54:00.511Z"),
                "lastDurableWallTime" : ISODate("2020-05-10T14:54:00.511Z")
        },
        "lastStableRecoveryTimestamp" : Timestamp(1589122430, 1),
        "lastStableCheckpointTimestamp" : Timestamp(1589122430, 1),
        "electionCandidateMetrics" : {
```

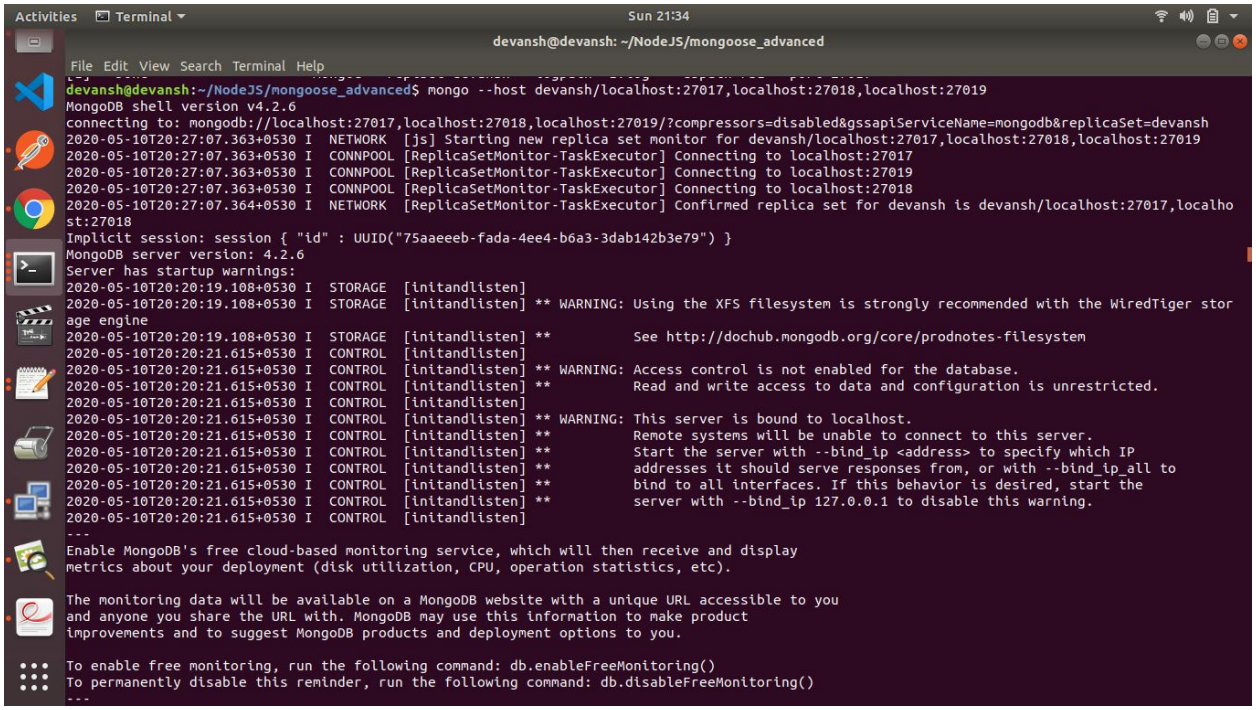
```
        "members" : [
                {
                        "_id" : 0,
                        "name" : "localhost:27017",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 226,
                        "optime" : {
                                "ts" : Timestamp(1589122440, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2020-05-10T14:54:00Z"),
                        "syncingTo" : "",
                        "syncSourceHost" : "",
                        "syncSourceId" : -1,
                        "infoMessage" : "",
                        "electionTime" : Timestamp(1589122310, 1),
                        "electionDate" : ISODate("2020-05-10T14:51:50Z"),
                        "configVersion" : 1,
                        "self" : true,
                        "lastHeartbeatMessage" : ""
                },
```

```
        {
                "_id" : 1,
                "name" : "localhost:27018",
                "health" : 1,
                "state" : 2,
                "stateStr" : "SECONDARY",
                "uptime" : 146,
                "optime" : {
                        "ts" : Timestamp(1589122440, 1),
                        "t" : NumberLong(1)
                },
                "optimeDurable" : {
                        "ts" : Timestamp(1589122440, 1),
                        "t" : NumberLong(1)
                },
                "optimeDate" : ISODate("2020-05-10T14:54:00Z"),
                "optimeDurableDate" : ISODate("2020-05-10T14:54:00Z"),
                "lastHeartbeat" : ISODate("2020-05-10T14:54:04.218Z"),
                "lastHeartbeatRecv" : ISODate("2020-05-10T14:54:03.797Z"),
                "pingMs" : NumberLong(0),
                "lastHeartbeatMessage" : "",
                "syncingTo" : "localhost:27017",
                "syncSourceHost" : "localhost:27017",
                "syncSourceId" : 0,
                "infoMessage" : "",
                "configVersion" : 1
        },
```

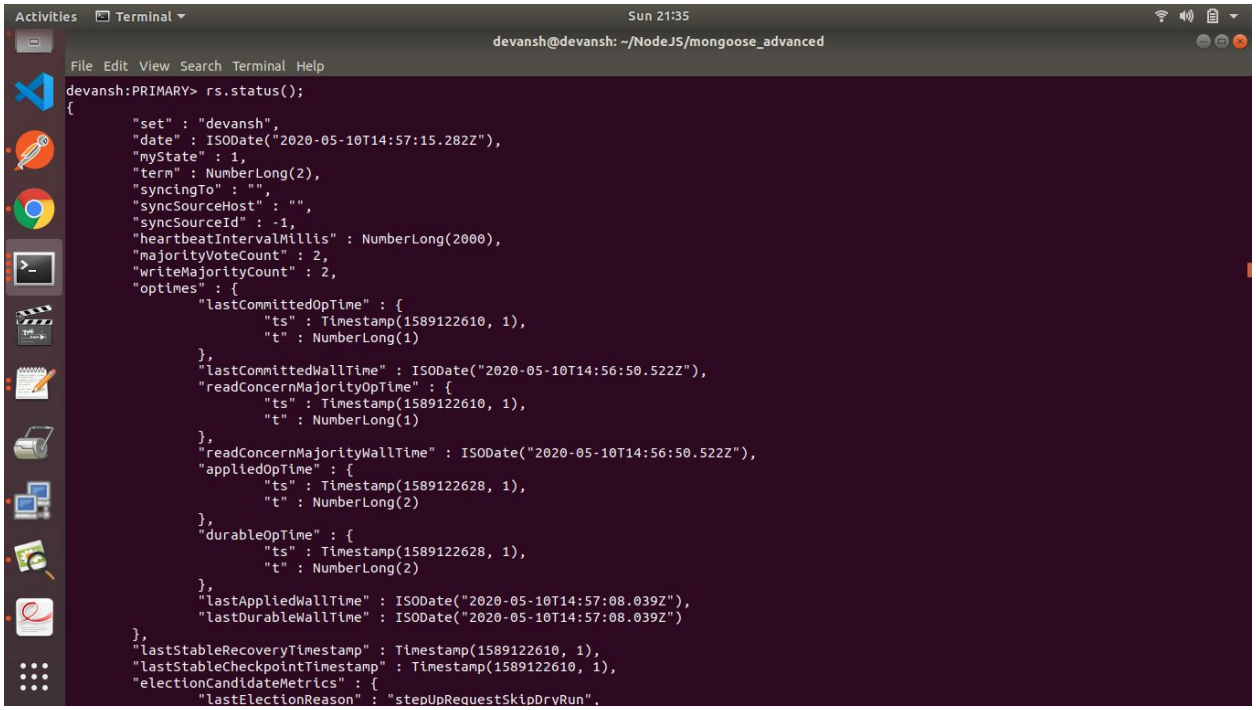
```
        {
                "_id" : 2,
                "name" : "localhost:27019",
                "health" : 1,
                "state" : 7,
                "stateStr" : "ARBITER",
                "uptime" : 146,
                "lastHeartbeat" : ISODate("2020-05-10T14:54:04.219Z"),
                "lastHeartbeatRecv" : ISODate("2020-05-10T14:54:03.916Z"),
                "pingMs" : NumberLong(0),
                "lastHeartbeatMessage" : "",
                "syncingTo" : "",
                "syncSourceHost" : "",
                "syncSourceId" : -1,
                "infoMessage" : "",
                "configVersion" : 1
        }
```




```
devansh@devansh:~/NodeJS/mongoose_advanced$ kill 26776
devansh@devansh:~/NodeJS/mongoose_advanced$ mongo
MongoDB shell version v4.2.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
2020-05-10T20:27:00.307+0530 E  QUERY    [js] Error: couldn't connect to server 127.0.0.1:27017, connection attempt failed: SocketException: E
rror connecting to 127.0.0.1:27017 :: caused by :: Connection refused :
connect@src/mongo/shell/mongo.js:341:17
@(connect):2:6
2020-05-10T20:27:00.308+0530 F  -        [main] exception: connect failed
2020-05-10T20:27:00.308+0530 E  -        [main] exiting with code 1
```


Activities
Terminal
Sun 21:34
devansh@devansh: ~/NodeJS/mongoose_advanced
File Edit View Search Terminal Help
devansh@devansh:~/NodeJS/mongoose_advanced$ mongo --host devansh/localhost:27017,localhost:27018,localhost:27019
MongoDB shell version v4.2.6
connecting to: mongodb://localhost:27017,localhost:27018,localhost:27019/?compressors=disabled&gssapiServiceName=mongodb&replicaSet=devansh
2020-05-10T20:27:07.363+0530 I NETWORK [js] Starting new replica set monitor for devansh/localhost:27017,localhost:27018,localhost:27019
2020-05-10T20:27:07.363+0530 I CONNPOOL [ReplicaSetMonitor-TaskExecutor] Connecting to localhost:27017
2020-05-10T20:27:07.363+0530 I CONNPOOL [ReplicaSetMonitor-TaskExecutor] Connecting to localhost:27019
2020-05-10T20:27:07.363+0530 I CONNPOOL [ReplicaSetMonitor-TaskExecutor] Connecting to localhost:27018
2020-05-10T20:27:07.364+0530 I NETWORK [ReplicaSetMonitor-TaskExecutor] Confirmed replica set for devansh is devansh/localhost:27017,localhost:27018
Implicit session: session { "id" : UUID("75aaeeeb-fada-4ee4-b6a3-3dab142b3e79") }
MongoDB server version: 4.2.6
Server has startup warnings:
2020-05-10T20:20:19.108+0530 I STORAGE [initandlisten]
2020-05-10T20:20:19.108+0530 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-05-10T20:20:19.108+0530 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten]
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten]
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2020-05-10T20:20:21.615+0530 I CONTROL [initandlisten]

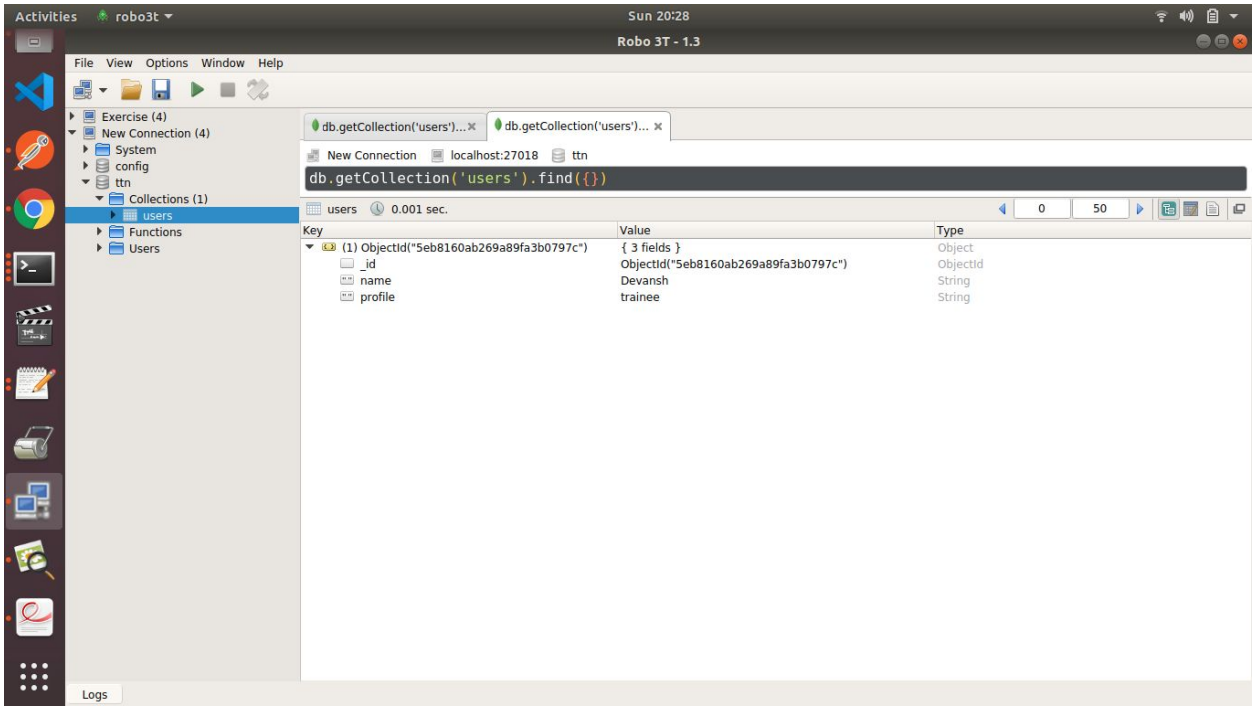
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. 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()


Activities
Terminal
Sun 21:35
devansh@devansh: ~/NodeJS/mongoose_advanced
File Edit View Search Terminal Help
devansh:PRIMARY> rs.status();
{
"set" : "devansh",
"date" : ISODate("2020-05-10T14:57:15.282Z"),
"myState" : 1,
"term" : NumberLong(2),
"syncingTo" : "",
"syncSourceHost" : "",
"syncSourceId" : -1,
"heartbeatIntervalMillis" : NumberLong(2000),
"majorityVoteCount" : 2,
"writeMajorityCount" : 2,
"optimes" : {
"lastCommittedOpTime" : {
"ts" : Timestamp(1589122610, 1),
"t" : NumberLong(1)
},
"lastCommittedWallTime" : ISODate("2020-05-10T14:56:50.522Z"),
"readConcernMajorityOpTime" : {
"ts" : Timestamp(1589122610, 1),
"t" : NumberLong(1)
},
"readConcernMajorityWallTime" : ISODate("2020-05-10T14:56:50.522Z"),
"appliedOpTime" : {
"ts" : Timestamp(1589122628, 1),
"t" : NumberLong(2)
},
"durableOpTime" : {
"ts" : Timestamp(1589122628, 1),
"t" : NumberLong(2)
},
"lastAppliedWallTime" : ISODate("2020-05-10T14:57:08.039Z"),
"lastDurableWallTime" : ISODate("2020-05-10T14:57:08.039Z")
},
"lastStableRecoveryTimestamp" : Timestamp(1589122610, 1),
"lastStableCheckpointTimestamp" : Timestamp(1589122610, 1),
"electionCandidateMetrics" : {
"lastElectionReason" : "stepUpRequestSkipDryRun",

```
"members" : [
        {
                "_id" : 0,
                "name" : "localhost:27017",
                "health" : 0,
                "state" : 8,
                "stateStr" : "(not reachable/healthy)",
                "uptime" : 0,
                "optime" : {
                        "ts" : Timestamp(0, 0),
                        "t" : NumberLong(-1)
                },
                "optimeDurable" : {
                        "ts" : Timestamp(0, 0),
                        "t" : NumberLong(-1)
                },
                "optimeDate" : ISODate("1970-01-01T00:00:00Z"),
                "optimeDurableDate" : ISODate("1970-01-01T00:00:00Z"),
                "lastHeartbeat" : ISODate("2020-05-10T14:57:14.064Z"),
                "lastHeartbeatRecv" : ISODate("2020-05-10T14:56:58.809Z"),
                "pingMs" : NumberLong(0),
                "lastHeartbeatMessage" : "Error connecting to localhost:27017 (127.0.0.1:27017) :: caused by :: Connection refused",
                "syncingTo" : "",
                "syncSourceHost" : "",
                "syncSourceId" : -1,
                "infoMessage" : "",
                "configVersion" : -1
        },
```

```
{
        "_id" : 1,
        "name" : "localhost:27018",
        "health" : 1,
        "state" : 1,
        "stateStr" : "PRIMARY",
        "uptime" : 416,
        "optime" : {
                "ts" : Timestamp(1589122628, 1),
                "t" : NumberLong(2)
        },
        "optimeDate" : ISODate("2020-05-10T14:57:08Z"),
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "infoMessage" : "",
        "electionTime" : Timestamp(1589122618, 1),
        "electionDate" : ISODate("2020-05-10T14:56:58Z"),
        "configVersion" : 1,
        "self" : true,
        "lastHeartbeatMessage" : ""
},
```

```
{
        "_id" : 2,
        "name" : "localhost:27019",
        "health" : 1,
        "state" : 7,
        "stateStr" : "ARBITER",
        "uptime" : 335,
        "lastHeartbeat" : ISODate("2020-05-10T14:57:14.043Z"),
        "lastHeartbeatRecv" : ISODate("2020-05-10T14:57:14.010Z"),
        "pingMs" : NumberLong(0),
        "lastHeartbeatMessage" : "",
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "infoMessage" : "",
        "configVersion" : 1
}
```

```
devansh@devansh:~/NodeJS/mongoose_advanced$ mongod --replSet devansh --logpath "4.log" --dbpath rs4 --port 27020 &
[4] 27393
devansh@devansh:~/NodeJS/mongoose_advanced$ mongod --replSet devansh --logpath "5.log" --dbpath rs5 --port 27021 &
[5] 27401
devansh@devansh:~/NodeJS/mongoose_advanced$ ps -ef | grep mongod
devansh  26777 25472  1 20:20 pts/1    00:00:13 mongod --replSet devansh --logpath 2.log --dbpath rs2 --port 27018
devansh  26786 25472  1 20:20 pts/1    00:00:09 mongod --replSet devansh --logpath 3.log --dbpath rs3 --port 27019
devansh  27393 25472 18 20:33 pts/1    00:00:00 mongod --replSet devansh --logpath 4.log --dbpath rs4 --port 27020
devansh  27401 25472 26 20:33 pts/1    00:00:00 mongod --replSet devansh --logpath 5.log --dbpath rs5 --port 27021
devansh  27440 25472  0 20:33 pts/1    00:00:00 grep --color=auto mongod
devansh@devansh:~/NodeJS/mongoose_advanced$ mongo
MongoDB shell version v4.2.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
2020-05-10T20:34:05.984+0530 E  QUERY    [js] Error: couldn't connect to server 127.0.0.1:27017, connection attempt failed: SocketException: E
rror connecting to 127.0.0.1:27017 :: caused by :: Connection refused :
connect@src/mongo/shell/mongo.js:341:17
@(connect):2:6
2020-05-10T20:34:05.986+0530 F  -        [main] exception: connect failed
2020-05-10T20:34:05.986+0530 E  -        [main] exiting with code 1
```

rs.status();

```
devansh@devansh:~/NodeJS/mongoose_advanced$ mongo --port 27018
MongoDB shell version v4.2.6
connecting to: mongodb://127.0.0.1:27018/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("01e650e2-c868-4511-955f-ffae482145dd") }
MongoDB server version: 4.2.6
Server has startup warnings:
2020-05-10T20:20:19.108+0530 I  STORAGE  [initandlisten]
2020-05-10T20:20:19.108+0530 I  STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger stor
age engine
2020-05-10T20:20:19.108+0530 I  STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten]
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten]
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2020-05-10T20:20:21.615+0530 I  CONTROL  [initandlisten]
---
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()
---
```

```
devansh:PRIMARY> rs.add({_id: 3, host:"localhost:27020"})
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1589123720, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1589123720, 1)
}
devansh:PRIMARY> rs.add({_id: 4, host:"localhost:27021"})
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1589123730, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1589123730, 1)
}
```

Now check _id : 3 and _id : 4 is also added as secondary database which has same data which is present on primary one.

```
        {
                "_id" : 3,
                "name" : "localhost:27020",
                "health" : 1,
                "state" : 2,
                "stateStr" : "SECONDARY",
                "uptime" : 25,
                "optime" : {
                        "ts" : Timestamp(1589123736, 3),
                        "t" : NumberLong(2)
                },
                "optimeDurable" : {
                        "ts" : Timestamp(1589123736, 3),
                        "t" : NumberLong(2)
                },
                "optimeDate" : ISODate("2020-05-10T15:15:36Z"),
                "optimeDurableDate" : ISODate("2020-05-10T15:15:36Z"),
                "lastHeartbeat" : ISODate("2020-05-10T15:15:44.920Z"),
                "lastHeartbeatRecv" : ISODate("2020-05-10T15:15:43.931Z"),
                "pingMs" : NumberLong(0),
                "lastHeartbeatMessage" : "",
                "syncingTo" : "localhost:27018",
                "syncSourceHost" : "localhost:27018",
                "syncSourceId" : 1,
                "infoMessage" : "",
                "configVersion" : 3
        },
```

```
        {
                "_id" : 4,
                "name" : "localhost:27021",
                "health" : 1,
                "state" : 2,
                "stateStr" : "SECONDARY",
                "uptime" : 15,
                "optime" : {
                        "ts" : Timestamp(1589123736, 3),
                        "t" : NumberLong(2)
                },
                "optimeDurable" : {
                        "ts" : Timestamp(1589123736, 3),
                        "t" : NumberLong(2)
                },
                "optimeDate" : ISODate("2020-05-10T15:15:36Z"),
                "optimeDurableDate" : ISODate("2020-05-10T15:15:36Z"),
                "lastHeartbeat" : ISODate("2020-05-10T15:15:44.920Z"),
                "lastHeartbeatRecv" : ISODate("2020-05-10T15:15:44.382Z"),
                "pingMs" : NumberLong(0),
                "lastHeartbeatMessage" : "",
                "syncingTo" : "localhost:27020",
                "syncSourceHost" : "localhost:27020",
                "syncSourceId" : 3,
                "infoMessage" : "",
                "configVersion" : 3
        }
```