NIELIT Gorakhpur

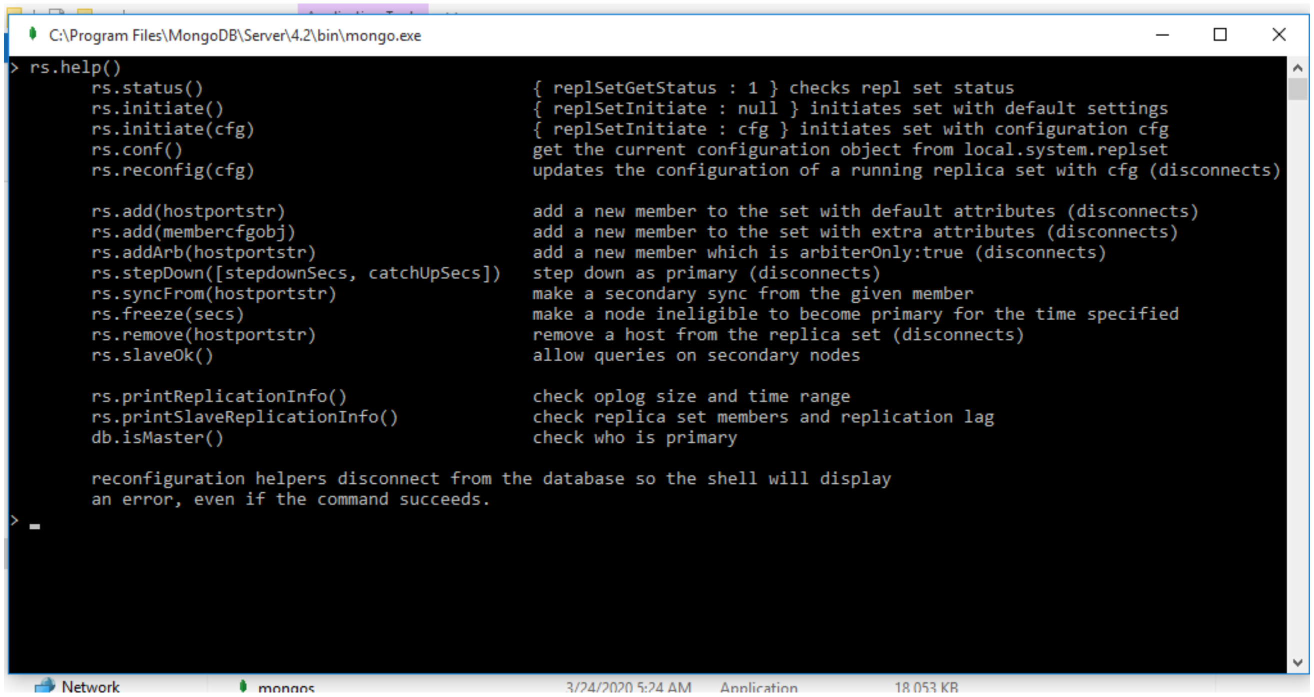
Course name: A level **SUBJECT: DATABASE TECHNOLOGIES**

Topic: MongoDB **DATE: 02/06/2020**

Replication Cont'd

Help on Replication

rs.help() command provides the basic help for various replica set function/ methods.



C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe

```
> rs.help()
        rs.status()                                 { replSetGetStatus : 1 } checks repl set status
        rs.initiate()                               { replSetInitiate : null } initiates set with default settings
        rs.initiate(cfg)                            { replSetInitiate : cfg } initiates set with configuration cfg
        rs.conf()                                   get the current configuration object from local.system.replset
        rs.reconfig(cfg)                            updates the configuration of a running replica set with cfg (disconnects)

        rs.add(hostportstr)                         add a new member to the set with default attributes (disconnects)
        rs.add(membercfgobj)                        add a new member to the set with extra attributes (disconnects)
        rs.addArb(hostportstr)                      add a new member which is arbiterOnly:true (disconnects)
        rs.stepDown([stepdownSecs, catchUpSecs])    step down as primary (disconnects)
        rs.syncFrom(hostportstr)                    make a secondary sync from the given member
        rs.freeze(secs)                             make a node ineligible to become primary for the time specified
        rs.remove(hostportstr)                      remove a host from the replica set (disconnects)
        rs.slaveOk()                                allow queries on secondary nodes

        rs.printReplicationInfo()                   check oplog size and time range
        rs.printSlaveReplicationInfo()              check replica set members and replication lag
        db.isMaster()                               check who is primary

        reconfiguration helpers disconnect from the database so the shell will display
        an error, even if the command succeeds.
> 
```

Adding Arbitrator

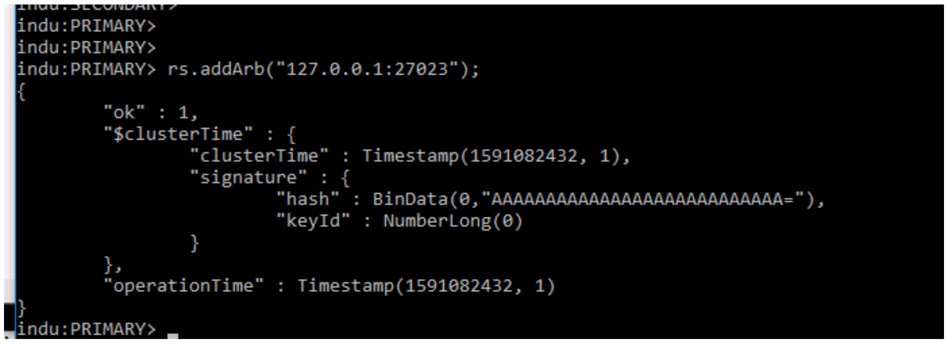
An arbiter is a mongodb instance that don't store a copy of data set and even cannot become a primary. It is 1 vote to pick a primary instance in the replicat set in when the case arises. Usually an arbiter instance is added to an existing replicat set that has even number of mongodb instances. By adding an arbiter instance, it allows the replicat set to have an uneven number of mongodb instances to avoid running into a tied situation when doing the election of picking a primary.

Syntax

rs.addArb("host name:port");

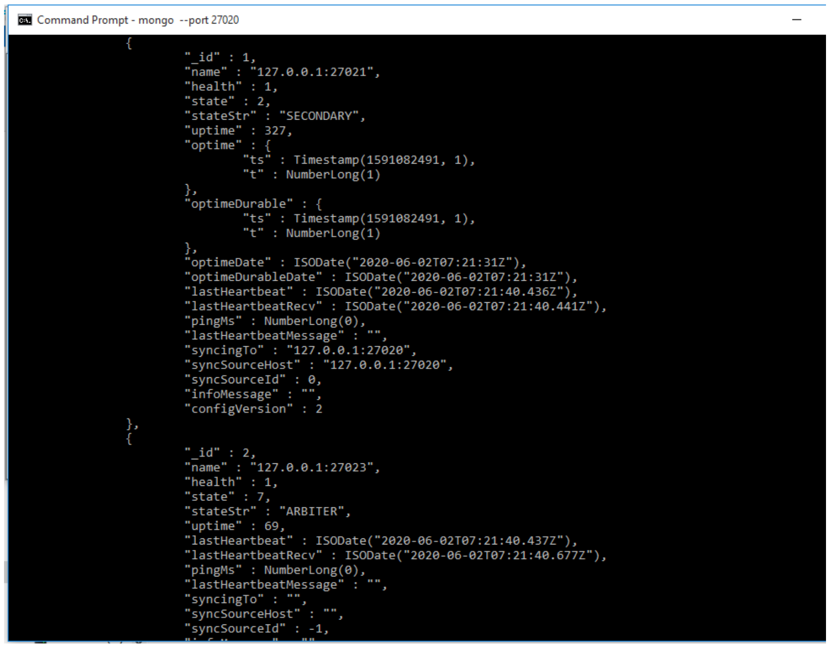
Example

rs.addArb("127.0.0.1:27023");



```
indu:PRIMARY>
indu:PRIMARY>
indu:PRIMARY> rs.addArb("127.0.0.1:27023");
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1591082432, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1591082432, 1)
}
indu:PRIMARY> _
```

If we run rs.status() now.



```
                {
                        "_id" : 1,
                        "name" : "127.0.0.1:27021",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 327,
                        "optime" : {
                                "ts" : Timestamp(1591082491, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDurable" : {
                                "ts" : Timestamp(1591082491, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2020-06-02T07:21:31Z"),
                        "optimeDurableDate" : ISODate("2020-06-02T07:21:31Z"),
                        "lastHeartbeat" : ISODate("2020-06-02T07:21:40.436Z"),
                        "lastHeartbeatRecv" : ISODate("2020-06-02T07:21:40.441Z"),
                        "pingMs" : NumberLong(0),
                        "lastHeartbeatMessage" : "",
                        "syncingTo" : "127.0.0.1:27020",
                        "syncSourceHost" : "127.0.0.1:27020",
                        "syncSourceId" : 0,
                        "infoMessage" : "",
                        "configVersion" : 2
                },
                {
                        "_id" : 2,
                        "name" : "127.0.0.1:27023",
                        "health" : 1,
                        "state" : 7,
                        "stateStr" : "ARBITER",
                        "uptime" : 69,
                        "lastHeartbeat" : ISODate("2020-06-02T07:21:40.437Z"),
                        "lastHeartbeatRecv" : ISODate("2020-06-02T07:21:40.677Z"),
                        "pingMs" : NumberLong(0),
                        "lastHeartbeatMessage" : "",
                        "syncingTo" : "",
                        "syncSourceHost" : "",
                        "syncSourceId" : -1,
```

Reconfiguring the Replica Set

rs.reconfig() method is used to reconfigure an existing replica set. It will overwrite all the existing replica set configuration. To reconfigure, we have to first connect to the primary replica set to run this method.

To reconfigure an existing replica set,

- first retrieve the current configuration with rs.config(),
- modify the configuration document as needed,
- pass the modified document to rs.reconfig()

Syntax

rs.reconfig(configuration, force)

Where,

Configuration- is the configuration document that specifies the configuration of a replica set.

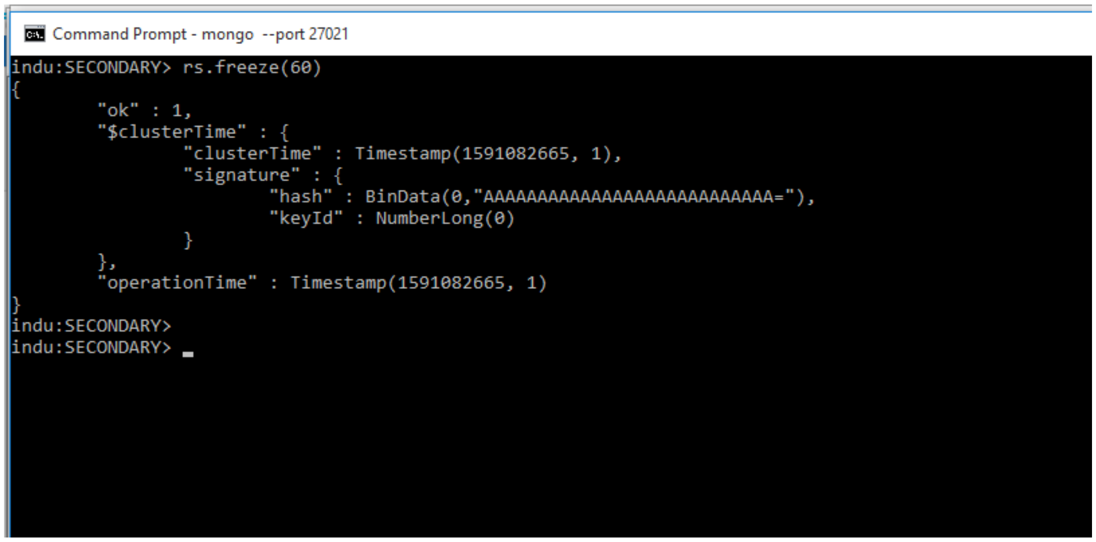
Force- If set as { force: true }, this forces the replica set to accept the new configuration even if a majority of the members are not accessible. It is Optional.

Freezing a member

rs.freeze() method prevents the current member from seeking election as primary for a period of time specified in seconds.

Syntax

rs.freeze(seconds)


Command Prompt - mongo --port 27021

```
indu:SECONDARY> rs.freeze(60)
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1591082665, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1591082665, 1)
}
indu:SECONDARY>
indu:SECONDARY> _
```

<u>Assignment</u>

1. How to add an arbitrar?

2. How to freeze a member?