MONGO DB LAB REPORT 2

ABSTRACT

To learn about Replication and Sharding in Mongo DB.

Ponathipan Jawahar & Cyril Naves

Exercise 4: Install MongoDB

In this exercise, I try to configure replication for mongoDB and understand how it works and recover from failure.

1. First, we create different directories storing the replications. In MongoDB, we have 3 kinds of nodes, they are primary, secondary and arbiter node. Since we already created D:\data\db in exercise 1 for primary node, here we only need to create directories for secondary nodes and arbiter node.

D:\data\replicatedServer1
D:\data\replicatedServer2
D:\data\arbiter

2. Set the node in exercise 1 as primary node.

mongod --replSet rest -dbpath D:/mongodb/data/db --port 27017

```
D:\ProgramFiles\my_mongodb_directory\MongoDB\Server\3.6\bin>mongod --replSet rest -dbpath D:/mongodb/data --port 27017
2018-05-04T07:05:15.505-0700 I CONTROL  [initandlisten] MongoDB starting : pid=21544 port=27017 dbpath=D:/mongodb/data 64-bit host=Athip
2018-05-04T07:05:15.505-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-05-04T07:05:15.510-0700 I CONTROL  [initandlisten] db version v3.6.4
2018-05-04T07:05:15.535-0700 I CONTROL  [initandlisten] git version: d0181a711f7e7f39e60b5aeb1dc7097bf6ae5856
2018-05-04T07:05:15.569-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2o-fips  27 Mar 2018
2018-05-04T07:05:15.604-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2018-05-04T07:05:15.610-0700 I CONTROL  [initandlisten] modules: none
2018-05-04T07:05:15.638-0700 I CONTROL  [initandlisten] build environment:
2018-05-04T07:05:15.646-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-05-04T07:05:15.670-0700 I CONTROL  [initandlisten]     distarch: x86_64
2018-05-04T07:05:15.676-0700 I CONTROL  [initandlisten]     target_arch: x86_64
2018-05-04T07:05:15.703-0700 I CONTROL  [initandlisten] options: { net: { port: 27017 }, replication: { replSet: "rest" }, storage: { dbPath: "D:/mongodb/data" } }
2018-05-04T07:05:15.911-0700 I -        [initandlisten] Detected data files in D:/mongodb/data created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2018-05-04T07:05:15.912-0700 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7616M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),
che_cursors=false,log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2018-05-04T07:05:16.316-0700 I STORAGE  [initandlisten] WiredTiger message [1525442716:316622][21544:140732109119824], txn-recover: Main recovery loop: starting at 1/26112
2018-05-04T07:05:16.474-0700 I STORAGE  [initandlisten] WiredTiger message [1525442716:474435][21544:140732109119824], txn-recover: Recovering log 1 through 2
2018-05-04T07:05:16.634-0700 I STORAGE  [initandlisten] WiredTiger message [1525442716:634143][21544:140732109119824], txn-recover: Recovering log 2 through 2
2018-05-04T07:05:16.863-0700 I STORAGE  [initandlisten] WiredTiger message [1525442716:863277][21544:140732109119824], txn-recover: Set global recovery timestamp: 0
2018-05-04T07:05:17.252-0700 I CONTROL  [initandlisten]
2018-05-04T07:05:17.252-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-05-04T07:05:17.257-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-05-04T07:05:17.282-0700 I CONTROL  [initandlisten]
2018-05-04T07:05:17.317-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2018-05-04T07:05:17.321-0700 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2018-05-04T07:05:17.354-0700 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2018-05-04T07:05:17.362-0700 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2018-05-04T07:05:17.388-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2018-05-04T07:05:17.420-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2018-05-04T07:05:17.455-0700 I CONTROL  [initandlisten]
2018-05-04T07:05:17.483-0700 I CONTROL  [initandlisten]
2018-05-04T07:05:17.488-0700 I CONTROL  [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory
ressure and poor performance.
2018-05-04T07:05:17.517-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-05-04T07:05:17.550-0700 I CONTROL  [initandlisten]
2018-05-04T16:05:18.003+0200 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'D:/mongodb/data/diagnostic.data'
2018-05-04T16:05:18.051+0200 I STORAGE  [initandlisten] createCollection: local.me with generated UUID: bcfa8e35-76f4-4dde-b1e7-68d44312e44d
2018-05-04T16:05:18.299+0200 I STORAGE  [initandlisten] createCollection: local.replset.minvalid with generated UUID: 2d166289-e1e3-4106-abf8-e9e29c65f4c4
2018-05-04T16:05:18.527+0200 I REPL     [initandlisten] Did not find local voted for document at startup.
2018-05-04T16:05:18.527+0200 I REPL     [initandlisten] Did not find local Rollback ID document at startup. Creating one.
2018-05-04T16:05:18.531+0200 I STORAGE  [initandlisten] createCollection: local.system.rollback.id with generated UUID: db8d8f32-1de4-4c16-ab3d-a59c5ed3d79e
2018-05-04T16:05:18.697+0200 I REPL     [initandlisten] Initialized the rollback ID to 1
2018-05-04T16:05:18.698+0200 I REPL     [initandlisten] Did not find local replica set configuration document at startup;  NoMatchingDocument: Did not find replica set configuration document in loc
.system.replset
2018-05-04T16:05:18.765+0200 I NETWORK  [initandlisten] waiting for connections on port 27017
```

3. Set 2 other secondary nodes.

mongod --replSet rest -dbpath D:\data\replicatedServer1 --port 27018

```
D:\ProgramFiles\my_mongodb_directory\MongoDB\Server\3.6\bin>mongod --replSet rest -dbpath D:\data\replicatedServer1 --port 27018
2018-05-04T07:22:54.436-0700 I CONTROL  [initandlisten] MongoDB starting : pid=26304 port=27018 dbpath=D:\data\replicatedServer1 64-bit host=Athip
2018-05-04T07:22:54.436-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-05-04T07:22:54.439-0700 I CONTROL  [initandlisten] db version v3.6.4
2018-05-04T07:22:54.439-0700 I CONTROL  [initandlisten] git version: d0181a711f7e7f39e60b5aeb1dc7097bf6ae5856
2018-05-04T07:22:54.453-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2o-fips  27 Mar 2018
2018-05-04T07:22:54.460-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2018-05-04T07:22:54.477-0700 I CONTROL  [initandlisten] modules: none
2018-05-04T07:22:54.482-0700 I CONTROL  [initandlisten] build environment:
2018-05-04T07:22:54.483-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-05-04T07:22:54.499-0700 I CONTROL  [initandlisten]     distarch: x86_64
2018-05-04T07:22:54.518-0700 I CONTROL  [initandlisten]     target_arch: x86_64
2018-05-04T07:22:54.537-0700 I CONTROL  [initandlisten] options: { net: { port: 27018 }, replication: { replSet: "rest" }, storage: { dbPath: "D:\data\replicatedServer1" } }
2018-05-04T07:22:54.548-0700 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7616M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),ca
che_cursors=false,log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2018-05-04T07:22:54.822-0700 I STORAGE  [initandlisten] WiredTiger message [1525443774:821358][26304:140732109119824], txn-recover: Set global recovery timestamp: 0
2018-05-04T07:22:55.329-0700 I CONTROL  [initandlisten]
2018-05-04T07:22:55.329-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-05-04T07:22:55.333-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-05-04T07:22:55.334-0700 I CONTROL  [initandlisten]
2018-05-04T07:22:55.347-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2018-05-04T07:22:55.369-0700 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2018-05-04T07:22:55.377-0700 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2018-05-04T07:22:55.386-0700 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2018-05-04T07:22:55.395-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2018-05-04T07:22:55.410-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2018-05-04T07:22:55.415-0700 I CONTROL  [initandlisten]
2018-05-04T07:22:55.416-0700 I CONTROL  [initandlisten]
2018-05-04T07:22:55.432-0700 I CONTROL  [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory p
ressure and poor performance.
2018-05-04T07:22:55.453-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-05-04T07:22:55.460-0700 I CONTROL  [initandlisten]
2018-05-04T16:22:55.484+0200 I STORAGE  [initandlisten] createCollection: local.startup_log with no UUID.
2018-05-04T16:22:55.897+0200 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'D:/data/replicatedServer1/diagnostic.data'
2018-05-04T16:22:55.903+0200 I STORAGE  [initandlisten] createCollection: local.me with no UUID.
2018-05-04T16:22:56.026+0200 W REPL     [ftdc] Rollback ID is not initialized yet.
2018-05-04T16:22:56.123+0200 I STORAGE  [initandlisten] createCollection: local.replset.minvalid with no UUID.
2018-05-04T16:22:56.282+0200 I REPL     [initandlisten] Did not find local voted for document at startup.
2018-05-04T16:22:56.282+0200 I REPL     [initandlisten] Did not find local Rollback ID document at startup. Creating one.
2018-05-04T16:22:56.286+0200 I STORAGE  [initandlisten] createCollection: local.system.rollback.id with no UUID.
2018-05-04T16:22:56.437+0200 I REPL     [initandlisten] Initialized the rollback ID to 1
2018-05-04T16:22:56.437+0200 I REPL     [initandlisten] Did not find local replica set configuration document at startup;  NoMatchingDocument: Did not find replica set configuration document in local
.system.replset
2018-05-04T16:22:56.442+0200 I NETWORK  [initandlisten] waiting for connections on port 27018
```

mongod --replSet rest - dbpath D:\data\replicatedServer2 --port 27019

```
Command Prompt - mongod  --replSet rest -dbpath D:\data\replicatedServer2 --port 27019

D:\ProgramFiles\my_mongodb_directory\MongoDB\Server\3.6\bin>mongod --replSet rest -dbpath D:\data\replicatedServer2 --port 27019
2018-05-04T07:24:22.702-0700 I CONTROL  [initandlisten] MongoDB starting : pid=27132 port=27019 dbpath=D:\data\replicatedServer2 64-bit host=Athip
2018-05-04T07:24:22.702-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-05-04T07:24:22.703-0700 I CONTROL  [initandlisten] db version v3.6.4
2018-05-04T07:24:22.703-0700 I CONTROL  [initandlisten] git version: d0181a711f7e7f39e60b5aeb1dc7097bf6ae5856
2018-05-04T07:24:22.703-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2o-fips  27 Mar 2018
2018-05-04T07:24:22.703-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2018-05-04T07:24:22.703-0700 I CONTROL  [initandlisten] modules: none
2018-05-04T07:24:22.703-0700 I CONTROL  [initandlisten] build environment:
2018-05-04T07:24:22.703-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-05-04T07:24:22.703-0700 I CONTROL  [initandlisten]     distarch: x86_64
2018-05-04T07:24:22.704-0700 I CONTROL  [initandlisten]     target_arch: x86_64
2018-05-04T07:24:22.704-0700 I CONTROL  [initandlisten] options: { net: { port: 27019 }, replication: { replSet: "rest" }, storage: { dbPath: "D:\data\replicatedServer2" } }
2018-05-04T07:24:22.705-0700 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7616M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),ca
che_cursors=false,log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2018-05-04T07:24:22.948-0700 I STORAGE  [initandlisten] WiredTiger message [1525443862:948034][27132:140732109119824], txn-recover: Set global recovery timestamp: 0
2018-05-04T07:24:23.379-0700 I CONTROL  [initandlisten]
2018-05-04T07:24:23.379-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-05-04T07:24:23.383-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-05-04T07:24:23.384-0700 I CONTROL  [initandlisten]
2018-05-04T07:24:23.397-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2018-05-04T07:24:23.401-0700 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2018-05-04T07:24:23.402-0700 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2018-05-04T07:24:23.418-0700 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2018-05-04T07:24:23.422-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2018-05-04T07:24:23.423-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2018-05-04T07:24:23.439-0700 I CONTROL  [initandlisten]
2018-05-04T07:24:23.463-0700 I CONTROL  [initandlisten]
2018-05-04T07:24:23.471-0700 I CONTROL  [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory p
ressure and poor performance.
2018-05-04T07:24:23.474-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-05-04T07:24:23.490-0700 I CONTROL  [initandlisten]
2018-05-04T16:24:23.535+0200 I STORAGE  [initandlisten] createCollection: local.startup_log with no UUID.
2018-05-04T16:24:23.910+0200 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'D:/data/replicatedServer2/diagnostic.data'
2018-05-04T16:24:23.927+0200 I STORAGE  [initandlisten] createCollection: local.me with no UUID.
2018-05-04T16:24:24.002+0200 W REPL     [ftdc] Rollback ID is not initialized yet.
2018-05-04T16:24:24.097+0200 I STORAGE  [initandlisten] createCollection: local.replset.minvalid with no UUID.
2018-05-04T16:24:24.243+0200 I REPL     [initandlisten] Did not find local voted for document at startup.
2018-05-04T16:24:24.243+0200 I REPL     [initandlisten] Did not find local Rollback ID document at startup. Creating one.
2018-05-04T16:24:24.259+0200 I STORAGE  [initandlisten] createCollection: local.system.rollback.id with no UUID.
2018-05-04T16:24:24.454+0200 I REPL     [initandlisten] Initialized the rollback ID to 1
2018-05-04T16:24:24.454+0200 I REPL     [initandlisten] Did not find local replica set configuration document at startup;  NoMatchingDocument: Did not find replica set configuration document in local
.system.replset
2018-05-04T16:24:24.459+0200 I NETWORK  [initandlisten] waiting for connections on port 27019
```

Hints: Here we should keep the replSet parameter same for all 3 nodes, or there will be error during initialization.

4. We connect to the primary node here.
mongo -- port 27017

5. Initialize the configuration, then we can find we are in the primary node now.

rs.initiate()

6. Add the secondary nodes as replications

```
rs.add (" localhost : 27018 ")
rs.add (" localhost: 27019 ")
```

7. Check the detail of replication config

```
rest:PRIMARY> rs.conf()
{
	"_id" : "rest",
	"version" : 3,
	"protocolVersion" : NumberLong(1),
	"members" : [
		{
			"_id" : 0,
			"host" : "localhost:27017",
			"arbiterOnly" : false,
			"buildIndexes" : true,
			"hidden" : false,
			"priority" : 1,
			"tags" : {

			},
			"slaveDelay" : NumberLong(0),
			"votes" : 1
		},
		{
			"_id" : 1,
			"host" : "localhost:27018",
			"arbiterOnly" : false,
			"buildIndexes" : true,
			"hidden" : false,
			"priority" : 1,
			"tags" : {

			},
			"slaveDelay" : NumberLong(0),
			"votes" : 1
		},
		{
			"_id" : 2,
```

```
            "host" : "localhost:27019",
            "arbiterOnly" : false,
            "buildIndexes" : true,
            "hidden" : false,
            "priority" : 1,
            "tags" : {

            },
            "slaveDelay" : NumberLong(0),
            "votes" : 1
        }
    ],
    "settings" : {
        "chainingAllowed" : true,
        "heartbeatIntervalMillis" : 2000,
        "heartbeatTimeoutSecs" : 10,
        "electionTimeoutMillis" : 10000,
        "catchUpTimeoutMillis" : -1,
        "catchUpTakeoverDelayMillis" : 30000,
        "getLastErrorModes" : {

        },
        "getLastErrorDefaults" : {
            "w" : 1,
            "wtimeout" : 0
        },
        "replicaSetId" : ObjectId("5aec6db4e93bb99cf7c87b5a")
    }
}
```

The configuration shows that we have successfully add the primary and secondary nodes.

8. Then we add the arbiter node.

```
mongod --replSet rest - dbpath D:\data\arbiter --port 27020

rs.addArb (" localhost: 27020 ")
```

```
rest:PRIMARY> rs.addArb("localhost:27020")
{
        "ok" : 1,
        "operationTime" : Timestamp(1525540657, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1525540657, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

We need the arbiter node because we need to have odd number of nodes to decide which node should serve as primary in case of failure.

9. Change to the restaurant database and try some commands.

use test;

db.restaurant.count();

```
rest:PRIMARY> use test
switched to db test
rest:PRIMARY> db.restaurant.count()
25360
rest:PRIMARY>
```

10.Connect to the database through a secondary node.

mongo -- port 27018

rs.slaveOk();

```
rest:SECONDARY> rs.slaveOk()
rest:SECONDARY>
```

Hints: slaveOk is a command for changing the authority setting, without this command, we cannot read or write on the database. After running this command,

we can read the database through this secondary node now.

```
rest:SECONDARY> use test
switched to db test
rest:SECONDARY> db.restaurant.count()
25360
rest:SECONDARY>
```

11.Insert new document to database

```
db.restaurant.insertOne(
{ "address" : {
"building" : "8008" ,
"coord" : [ -83.856077 , 44.848447 ],
"street" : "117 Main Road" ,
"zipcode" : "625515"
},
"borough" : "Chinnamanur" ,
"cuisine" : "Chicken" ,
"grades" : [
{ "date" : { "$date" : 1393804800000 }, "grade" : "A" , "score" : 10 },
{ "date" : { "$date" : 1378857600000 }, "grade" : "A" , "score" : 12},
{ "date" : { "$date" : 1358985600000 }, "grade" : "A" , "score" : 15 },
{ "date" : { "$date" : 1322006400000 }, "grade" : "A" , "score" : 7 },
{ "date" : { "$date" : 1299715200000 }, "grade" : "B" , "score" : 18 }
],
"name" : "Ponvillas" ,
"restaurant_id" : "100780" }
)
```

```
rest:PRIMARY> db.restaurant.insertOne(
... { "address" : {
... "building" : "8008" ,
... "coord" : [ -83.856077 , 44.848447 ],
... "street" : "117 Main Road" ,
... "zipcode" : "625515"
... },
... "borough" : "Chinnamanur" ,
... "cuisine" : "Chicken" ,
... "grades" : [
... { "date" : { "$date" : 1393804800000 }, "grade" : "A" , "score" : 10 },
... { "date" : { "$date" : 1378857600000 }, "grade" : "A" , "score" : 12},
... { "date" : { "$date" : 1358985600000 }, "grade" : "A" , "score" : 15 },
... { "date" : { "$date" : 1322006400000 }, "grade" : "A" , "score" : 7 },
... { "date" : { "$date" : 1299715200000 }, "grade" : "B" , "score" : 18 }
... ],
... "name" : "Ponvillas" ,
... "restaurant_id" : "100780" }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5aedee6efb1c0904638abace")
```

1) We should do this through master node, because we don't have write access on slave node. We can only have read access through slaveOk();
2) After inserting this document, we can also find it through slave node.
db.restaurant.find({name:"Ponvillas"})

```
rest:PRIMARY> db.restaurant.find({name:"Ponvillas"})
{ "_id" : ObjectId("5aedee6efb1c0904638abace"), "address" : { "building" : "8008", "coord" : [ -83.856077, 44.848447 ],
"street" : "117 Main Road", "zipcode" : "625515" }, "borough" : "Chinnamanur", "cuisine" : "Chicken", "grades" : [ { "da
te" : { "$date" : 1393804800000 }, "grade" : "A", "score" : 10 }, { "date" : { "$date" : 1378857600000 }, "grade" : "A",
 "score" : 12 }, { "date" : { "$date" : 1358985600000 }, "grade" : "A", "score" : 15 }, { "date" : { "$date" : 132200640
0000 }, "grade" : "A", "score" : 7 }, { "date" : { "$date" : 1299715200000 }, "grade" : "B", "score" : 18 } ], "name" :
"Ponvillas", "restaurant_id" : "100780" }
rest:PRIMARY>
```

```
rest:SECONDARY> db.restaurant.find({name:"Ponvillas"})
{ "_id" : ObjectId("5aedee6efb1c0904638abace"), "address" : { "building" : "8008", "coord" : [ -83.856077, 44.848447 ],
"street" : "117 Main Road", "zipcode" : "625515" }, "borough" : "Chinnamanur", "cuisine" : "Chicken", "grades" : [ { "da
te" : { "$date" : 1393804800000 }, "grade" : "A", "score" : 10 }, { "date" : { "$date" : 1378857600000 }, "grade" : "A",
 "score" : 12 }, { "date" : { "$date" : 1358985600000 }, "grade" : "A", "score" : 15 }, { "date" : { "$date" : 132200640
0000 }, "grade" : "A", "score" : 7 }, { "date" : { "$date" : 1299715200000 }, "grade" : "B", "score" : 18 } ], "name" :
"Ponvillas", "restaurant_id" : "100780" }
rest:SECONDARY>
```

12.Insert another document to the database and kill the master node by ctrl+c. Then we run following commands in master terminal.

use admin
db.shutdownServer()

```
rest:PRIMARY> use admin
switched to db admin
rest:PRIMARY> db.shutdownServer()
server should be down...
2018-05-05T19:57:31.741+0200 I NETWORK  [thread1] trying reconnect to 127.0.0.1:27017 (127.0.0.1) failed
2018-05-05T19:57:32.006+0200 I NETWORK  [thread1] reconnect 127.0.0.1:27017 (127.0.0.1) ok
rest:SECONDARY>
```

13.Now we check the current status of the cluster.
rest:SECONDARY> rs.status()
{

```
"set" : "rest",
"date" : ISODate("2018-05-05T17:59:00.140Z"),
"myState" : 1,
"term" : NumberLong(2),
"heartbeatIntervalMillis" : NumberLong(2000),
"optimes" : {
    "lastCommittedOpTime" : {
        "ts" : Timestamp(1525543050, 1),
        "t" : NumberLong(1)
    },
    "readConcernMajorityOpTime" : {
        "ts" : Timestamp(1525543050, 1),
        "t" : NumberLong(1)
    },
    "appliedOpTime" : {
        "ts" : Timestamp(1525543137, 1),
        "t" : NumberLong(2)
    },
    "durableOpTime" : {
        "ts" : Timestamp(1525543137, 1),
        "t" : NumberLong(2)
    }
},
"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("2018-05-05T17:58:52.243Z"),
"lastHeartbeatRecv" : ISODate("2018-05-05T17:57:31.441Z"),
"pingMs" : NumberLong(0),
"lastHeartbeatMessage" : "No connection could be made because the target machine actively refused it.",
"configVersion" : -1
},
{
"_id" : 1,
"name" : "localhost:27018",
"health" : 1,
"state" : 1,
"stateStr" : "PRIMARY",
"uptime" : 99366,
"optime" : {
"ts" : Timestamp(1525543137, 1),
"t" : NumberLong(2)
},
"optimeDate" : ISODate("2018-05-05T17:58:57Z"),
"infoMessage" : "could not find member to sync from",
"electionTime" : Timestamp(1525543062, 1),
"electionDate" : ISODate("2018-05-05T17:57:42Z"),
"configVersion" : 4,
"self" : true
},
{
"_id" : 2,
"name" : "localhost:27019",
"health" : 1,
"state" : 2,
"stateStr" : "SECONDARY",
"uptime" : 6054,
"optime" : {
"ts" : Timestamp(1525543137, 1),
"t" : NumberLong(2)
},
"optimeDurable" : {
"ts" : Timestamp(1525543137, 1),
"t" : NumberLong(2)

```
                },
                "optimeDate" : ISODate("2018-05-05T17:58:57Z"),
                "optimeDurableDate" : ISODate("2018-05-05T17:58:57Z"),
                "lastHeartbeat" : ISODate("2018-05-05T17:58:59.034Z"),
                "lastHeartbeatRecv" : ISODate("2018-05-05T17:58:58.324Z"),
                "pingMs" : NumberLong(0),
                "syncingTo" : "localhost:27018",
                "configVersion" : 4
            },
            {
                "_id" : 3,
                "name" : "localhost:27020",
                "health" : 1,
                "state" : 7,
                "stateStr" : "ARBITER",
                "uptime" : 2077,
                "lastHeartbeat" : ISODate("2018-05-05T17:58:59.034Z"),
                "lastHeartbeatRecv" : ISODate("2018-05-05T17:58:58.637Z"),
                "pingMs" : NumberLong(0),
                "configVersion" : 4
            }
        ],
        "ok" : 1,
        "operationTime" : Timestamp(1525543137, 1),
        "$clusterTime" : {
            "clusterTime" : Timestamp(1525543137, 1),
            "signature" : {
                "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                "keyId" : NumberLong(0)
            }
        }
}
```


We can find that the second slave node is serving as master node now.

14.Launch mongo again in master terminal, now we can connect to the 2 slave nodes.

Yes

15.We can find that the document also exists in this database, which proves that mongoDB can ensure data consistency.

Exercise 5: Basic Commands

1. Create the config server

1. In a terminal launch: mkdir /data/configdb

2. Launch the server: mongod

3. configsvr --dbpath /data/configdb –port

mongod --configsvr --replSet confrep1 --dbpath D:\data\configdb –port 27010

```
D:\ProgramFiles\my_mongodb_directory\MongoDB\Server\3.6\bin>mongod --configsvr --replSet confrep1 --dbpath D:\data\configdb --port 27010
2018-05-06T01:00:15.578-0700 I CONTROL  [initandlisten] MongoDB starting : pid=27092 port=27010 dbpath=D:\data\configdb 64-bit host=Athip
2018-05-06T01:00:15.578-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-05-06T01:00:15.583-0700 I CONTROL  [initandlisten] db version v3.6.4
2018-05-06T01:00:15.611-0700 I CONTROL  [initandlisten] git version: d0181a711f7e7f39e60b5aeb1dc7097bf6ae5856
2018-05-06T01:00:15.651-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2o-fips  27 Mar 2018
2018-05-06T01:00:15.656-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2018-05-06T01:00:15.690-0700 I CONTROL  [initandlisten] modules: none
2018-05-06T01:00:15.698-0700 I CONTROL  [initandlisten] build environment:
2018-05-06T01:00:15.724-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-05-06T01:00:15.732-0700 I CONTROL  [initandlisten]     distarch: x86_64
2018-05-06T01:00:15.752-0700 I CONTROL  [initandlisten]     target_arch: x86_64
```

2. Create a shard router : mongos -configdb -port

mongos --configdb confrep1/localhost:27010 --port 27011

3. Launch the two data server:

1. mongod --dbpath /data/shard1 --port

2. mongod --dbpath /data/shard2 --port

3. mongod --dbpath /data/shard3 --port

mongod --dbpath D:\data\shard1 --shardsvr --port 27012

mongod --dbpath D:\data\shard2 --shardsvr --port 27013

mongod --dbpath D:\data\shard3 --shardsvr --port 27014

```
D:\ProgramFiles\my_mongodb_directory\MongoDB\Server\3.6\bin>mongod --dbpath D:\data\shard1 --shardsvr --port 27012
2018-05-06T03:12:15.929-0700 I CONTROL  [initandlisten] MongoDB starting : pid=19316 port=27012 dbpath=D:\data\shard1 64-bit host=Athip
2018-05-06T03:12:15.929-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-05-06T03:12:15.933-0700 I CONTROL  [initandlisten] db version v3.6.4
2018-05-06T03:12:15.961-0700 I CONTROL  [initandlisten] git version: d0181a711f7e7f39e60b5aeb1dc7097bf6ae5856
2018-05-06T03:12:15.999-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2o-fips  27 Mar 2018
2018-05-06T03:12:16.032-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2018-05-06T03:12:16.060-0700 I CONTROL  [initandlisten] modules: none
2018-05-06T03:12:16.092-0700 I CONTROL  [initandlisten] build environment:
2018-05-06T03:12:16.125-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-05-06T03:12:16.158-0700 I CONTROL  [initandlisten]     distarch: x86_64
2018-05-06T03:12:16.165-0700 I CONTROL  [initandlisten]     target_arch: x86_64
```

4. Launch the router and two shard server:

1. mongo –port

mongo -- port 27011

2. sh.addShard(“localhost:shard1_port”);

sh.addShard(“localhost:27012”)

3. sh.addShard(“localhost:shard2_port”);

sh.addShard(“localhost:27013”)

```
Command Prompt - mongo  --port 27010
ind_ip_all to
018-05-06T01:00:17.487-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior
start the
018-05-06T01:00:17.520-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disabl
ng.
018-05-06T01:00:17.528-0700 I CONTROL  [initandlisten]
018-05-06T01:00:17.528-0700 I CONTROL  [initandlisten]
018-05-06T01:00:17.528-0700 I CONTROL  [initandlisten] ** WARNING: The file system cache of this machine is
o be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
018-05-06T01:00:17.528-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-
018-05-06T01:00:17.529-0700 I CONTROL  [initandlisten]
 rs.initiate()

        "info2" : "no configuration specified. Using a default configuration for the set",
        "me" : "localhost:27010",
        "ok" : 1,
        "operationTime" : Timestamp(1525594922, 1),
        "$gleStats" : {
                "lastOpTime" : Timestamp(1525594922, 1),
                "electionId" : ObjectId("000000000000000000000000")
        },
        "$clusterTime" : {
                "clusterTime" : Timestamp(1525594922, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }

onfrep1:OTHER>
```

```
D:\ProgramFiles\my_mongodb_directory\MongoDB\Server\3.6\bin>mongo --port 27011
MongoDB shell version v3.6.4
connecting to: mongodb://127.0.0.1:27011/
MongoDB server version: 3.6.4
Server has startup warnings:
2018-05-06T10:03:04.777+0200 I CONTROL  [main]
2018-05-06T10:03:04.777+0200 I CONTROL  [main] ** WARNING: Access control is not enabled for the database.
2018-05-06T10:03:04.780+0200 I CONTROL  [main] **          Read and write access to data and configuration is unrestrict
ed.
2018-05-06T10:03:04.780+0200 I CONTROL  [main]
2018-05-06T10:03:04.780+0200 I CONTROL  [main] ** WARNING: This server is bound to localhost.
2018-05-06T10:03:04.780+0200 I CONTROL  [main] **          Remote systems will be unable to connect to this server.
2018-05-06T10:03:04.780+0200 I CONTROL  [main] **          Start the server with --bind_ip <address> to specify which IP
```

```
mongos> sh.addShard("localhost:27012")
{
        "shardAdded" : "shard0000",
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1525595443, 4),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1525595443, 3)
}
mongos> sh.addShard("localhost:27013")
{
        "shardAdded" : "shard0001",
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1525595502, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1525595502, 2)
}
```

5. Create a collection in the router terminal

1. use test ;

2. db.createCollection(“athip”);

3. db.createIndex(“index”:1);

```
mongos> use test
switched to db test
mongos> db.createCollection("athip")
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1525599756, 6),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1525599756, 6)
}
mongos> db.tempcreateIndex({"index":1})
2018-05-06T11:43:15.088+0200 E QUERY    [thread1] TypeError: db.tempcreateIndex is not a function :
@(shell):1:1
mongos> db.temp.createIndex({"index":1})
{
        "raw" : {
                "localhost:27012" : {
                        "ok" : 0,
                        "errmsg" : "Cannot accept sharding commands if not started with --shardsvr",
                        "code" : 193,
                        "codeName" : "NoShardingEnabled"
                }
        },
```

6. Enable sharding on this database

sh.enableSharding("test")

sh.shardCollection("test.athip" ,{id: 1 })

```
mongos> sh.enableSharding("test")
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1525600279, 5),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1525600279, 5)
}
mongos> sh.shardCollection("test.athip",{id:1})
{
        "collectionsharded" : "test.athip",
        "collectionUUID" : UUID("8a44f08b-7fab-402c-a836-586f2febf566"),
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1525600390, 10),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1525600390, 10)
}
```

7. Then we can check the status of the cluster

```
mongos> sh.status()

--- Sharding Status ---
  sharding version: {
      "_id" : 1,
      "minCompatibleVersion" : 5,
      "currentVersion" : 6,
      "clusterId" : ObjectId("5aeebb2fb49dd49070e12e29")
  }
  shards:
      {  "_id" : "shard0000",  "host" : "localhost:27012",  "state" : 1 }
      {  "_id" : "shard0001",  "host" : "localhost:27013",  "state" : 1 }

  active mongoses:
      "3.6.4" : 1
  autosplit:
      Currently enabled: yes
  balancer:
      Currently enabled:  yes
      Currently running:  no
```

```
    Failed balancer rounds in last 5 attempts:  0
    Migration Results for the last 24 hours:
        No recent migrations
  databases:
    {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
        config.system.sessions
            shard key: { "_id" : 1 }
            unique: false
            balancing: true
            chunks:
                shard0000      1
            { "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on :
shard0000 Timestamp(1, 0)
    {  "_id" : "test",  "primary" : "shard0000",  "partitioned" : true }
        test.athip
            shard key: { "id" : 1 }
            unique: false
            balancing: true
            chunks:
                shard0000      1
            { "id" : { "$minKey" : 1 } } -->> { "id" : { "$maxKey" : 1 } } on :
shard0000 Timestamp(1, 0)
```

Before checking the distribution of data, we insert some test data into test.athip collection.

```
mongos> for (var i=1;i<=10000;i++){
... db.athip.save({id:1,"message":"message"+i});
... }
```

```
mongos> for (var i=1;i<=10000;i++){ db.athip.save({id:1,"message":"message"+i}); }
WriteResult({ "nInserted" : 1 })
mongos>
```

Then we can launch shard1 and shard2 to check how many documents are stored in each shard. Below is the result:

```
//shard1
mongo  --port 27012
> use test
switched to db test
> db.athip.count()
99999
```

```
//shard2
mongo --port 22013
use test
switched to db test
db.athip.count()
1
```

The result shows that 99999 documents are stored in shard1, and 1 document is stored in shard2, which means the documents are distributed in different shards.

8. Here, we try to add a new shard and check the new status of data distribution.

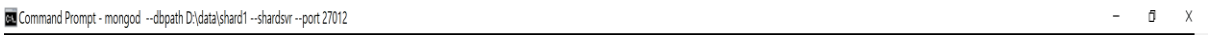
sh.addShard("localhost:27014")

```
//shard1
> use test
switched to db test
> db.athip.count()
99981
//shard2
> use test
switched to db test
>db.athip.count()
1
//shard3
> use test
switched to db test
>db.athip.count()
18
```

We can find that the number of documents stored in different shards are: 99981, 1, 18. The result shows that the new shard also help balance the data distribution.

9. Kill the shard<shard 1 port>

I am using windows machine, I close the shard 1 command prompt which is running on the port 27012



10.In the end, I check the new data distribution status, but the result is out of my expectation.

```
//shard2
> use test
switched to db test
> db.athip.count()
1
//shard3
> use test
switched to db test
> db.athip.count()
18
```

The data on shard1 is not reallocated to other shards. Then I try to count the number of documents in router server. There is an error returned like this below.

```
E QUERY [thread1] Error : count failed: {
"code" : 133 ,
"ok" : 0 ,
"errmsg" : "could not find host matching read preference { mode:
\"primary\", tags: [ {} ] }
```

It shows that the recovery is not available in this case. Then I remove shard1 in router terminal by following command.

```
db.runCommand({ "removeshard" : "localhost:27012”" })
```

```
mongos> use admin
switched to db admin
mongos> db.runCommand({"removeshard":"localhost:27012"})
{
        "msg" : "draining started successfully",
        "state" : "started",
        "shard" : "shard0000",
        "note" : "you need to drop or movePrimary these databases",
        "dbsToMove" : [
                "test"
        ],
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1525605048, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1525605048, 2)
}
mongos>
```

After running this command, the data is reallocated.

```
//shard2
> use test
switched to db test
> db.athip.count()
99982
//shard3
> use test
switched to db test
> db.athip.count()
18
```