Shaded Cluster Setup

Replica Sets Setup :::

- Create Three directories /Users/vaneetkumar/data/mongocluster/replicaset/db1 , db2 , db3
- Start Three Mongo DB Instances with below commands
- mongod --dbpath /Users/vaneetkumar/data/mongocluster/replicaset/db1 --replSet rs0 (27017) --shardsvr
- mongod --dbpath /Users/vaneetkumar/data/mongocluster/replicaset/db2 --replSet rs0 --port 27018 —shardsvr
- mongod --dbpath /Users/vaneetkumar/data/mongocluster/replicaset/db3 --replSet rs0 --port 27019 —shardsvr
- — shardsvr is needed if you need to create a sharded cluster later .

- Go To mongo shell of any of the server lets goto the mongo shell of 27017 port mongo by just typing *mongo*
- Set a variable var configuration = var configuration = {_id : 'rs0' , members : [{_id : 0 , host : 'localhost:27017'} , {_id : 1 , host : 'localhost:27018' , slaveDelay : 20 , priority :0 } , {_id : 2 , host : 'localhost:27019'}] }
- Check configuration set by typing *configuration*

- Hit command *rs.initiate(configuration)*
- Connect to all mongo in different terminals by typing mongo —port 27108 and mongo —port 27019
- Now one of the server will become primary and other will become secondary
- Initially the one which is primary may be shown as secondary as may be election would have not been completed by then

- Different b/w Primary and Secondary
-
- Who can Vote and who can be elected as Primary .
- *By* default all servers are having values priority 1 and votes 1 . As in our replica set 27018 port mongo has priority as 0 it means mongo running on port 27018 can never become a primary . But all can vote in election as by default all has votes value as 0.
- There must be odd number of voting members in a replica set (3,5, 7) . In mongo db RS there can be max 7 voting members . Odd number of voting members prevent ties .

- *Reading and Writing*
- *We can write only In primary server . We can't write in a secondary server .*
- *Try writing on a secondary server you will get* WriteResult({ "writeError" : { "code" : 10107, "errmsg" : "not master" } }) .
- Try Wiring on a primary server . You will be able to

write and WriteResult({ "nInserted" : 1 }) will be returned

- *Try Reading from secondary server . You will not be able to read and get* "errmsg" : "not master and slaveOk=false",
- *Now go to a slave (27018)and issue* rs.slaveOk() . Now You will be able to check dbs , tables and query data fro tables .
- To disable a secondary to read data from it command rs.slaveOk(false) . Writing this command to primary will not have any effect.

- How does mongo db relocation process works
- Primary always have multiple databases and one of the db it has is name `local` . In this db there is a collection oplog.rs . All the operation being performed on any object In any collection in any db is written to this collection. All the operation are written in idempotent manner . Means all database can be reconstructed if we apply full oplog again .
- All secondaries read data from this oplog.rs collection and replicate data in them . Due to some network delay or some other factors as load there can be situation where data just written on primary is not found on secondary by read query as data is eventually consistent . This is temporary and after some time data will become fully consistent .
- Oplog.rs is capped collection and you can Check it by `db.oplog.rs.isCapped()` . It will return true . As this collection is responsible for all replication process and secondaries some time can go down as well hence this collection must be very large in size .
- Default size of oplog.rs collection is 5% of free disk space but not more that 50 GB and also not less that 990MB

- You can check its max size by use local ; `db.oplog.rs.stats()` and check the max size . It can vary on versions .
- Default is goof you can also change it but prefer changing before starting a replica set .

- Lets test the 27018 a which has been configured a delayed server by 20 minutes .
- Insert a document in primary .
- Check that it will be available in non delayed server . It is present .
- But in delayed one 27018 this will appear just after 20 seconds . Before that it will not appear .
-
- rs.conf() . It shows which members are Priority , Voting eligible , Hidden , delayed , arbiters and can build indexes
- {
  "_id" : "rs0",
  "version" : 1,
  "protocolVersion" : NumberLong(1),
  "writeConcernMajorityJournalDefault" : true,
  "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" : 0,
            "tags" : {

            },
            "slaveDelay" : NumberLong(20),
            "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("5f30fdea8f0b3c88e4628df9")
    }
}
```

- For a Regular server Priority can be set as 0 (It can't become a primary) and 1 as well . Its Voting rights can be 0 OR 1 . It is not hidden and delayed . It is not an arbiter . It builds indexes as it will handle traffic .
- For a Delayed member , Priority is always 0 , Voting can be 0 or 1 , It can or cannot be a hidden member . It's never an arbiter , It builds indexes as it can take queries on a delayed data as well. It is a backup server which is there to save the data in case if some deadly query deletes the full data etc.
- Arbiter is only for voting purposes . Its Priority is always 0 , Its Voting is always 1 , Its not hidden and delayed and It doesn't build indexes .

- Current State of a Replica Set:::

- rs.status() . Health 1 means server is healthy . StateStr tells which is primary and which is secondary . SyncingTo means to which a secondary member is syncing to . It is always a value of primary server address . Primary don’y sync to anyone . If this command is executed on a member which is network separated from all other servers then this command will show that all servers are down and lost .


```
{
    "set" : "rs0",
    "date" :
ISODate("2020-08-10T09:13:17.275Z"),
    "myState" : 2,
    "term" : NumberLong(2),
    "syncingTo" : "localhost:27017",
    "syncSourceHost" : "localhost:27017",
    "syncSourceId" : 0,
    "heartbeatIntervalMillis" :
NumberLong(2000),
    "majorityVoteCount" : 2,
    "writeMajorityCount" : 2,
    "optimes" : {
        "lastCommittedOpTime" : {
            "ts" : Timestamp(1597050787, 1),
            "t" : NumberLong(2)
        },
        "lastCommittedWallTime" :
ISODate("2020-08-10T09:13:07.296Z"),
```

```
        "readConcernMajorityOpTime" : {
            "ts" : Timestamp(1597050767, 1),
            "t" : NumberLong(2)
        },
        "readConcernMajorityWallTime" :
ISODate("2020-08-10T09:12:47.291Z"),
        "appliedOpTime" : {
            "ts" : Timestamp(1597050767, 1),
            "t" : NumberLong(2)
        },
        "durableOpTime" : {
            "ts" : Timestamp(1597050767, 1),
            "t" : NumberLong(2)
        },
        "lastAppliedWallTime" :
ISODate("2020-08-10T09:12:47.291Z"),
        "lastDurableWallTime" :
ISODate("2020-08-10T09:12:47.291Z")
    },
    "lastStableRecoveryTimestamp" :
Timestamp(1597050747, 1),
    "lastStableCheckpointTimestamp" :
Timestamp(1597050747, 1),
    "electionParticipantMetrics" : {
        "votedForCandidate" : true,
        "electionTerm" : NumberLong(2),
        "lastVoteDate" :
ISODate("2020-08-10T08:45:35.781Z"),
        "electionCandidateMemberId" : 2,
```

```
        "voteReason" : "",
        "lastAppliedOpTimeAtElection" : {
            "ts" : Timestamp(1597049062, 1),
            "t" : NumberLong(1)
        },
        "maxAppliedOpTimeInSet" : {
            "ts" : Timestamp(1597049135, 1),
            "t" : NumberLong(1)
        },
        "priorityAtElection" : 0,
        "newTermStartDate" :
ISODate("2020-08-10T08:45:35.846Z"),
        "newTermAppliedDate" :
ISODate("2020-08-10T08:45:55.066Z")
    },
    "members" : [
        {
            "_id" : 0,
            "name" : "localhost:27017",
            "health" : 1,
            "state" : 2,
            "stateStr" : "SECONDARY",
            "uptime" : 4545,
            "optime" : {
                "ts" : Timestamp(1597050787, 1),
                "t" : NumberLong(2)
            },
            "optimeDurable" : {
                "ts" : Timestamp(1597050787, 1),
```

```
                "t" : NumberLong(2)
            },
            "optimeDate" :
ISODate("2020-08-10T09:13:07Z"),
            "optimeDurableDate" :
ISODate("2020-08-10T09:13:07Z"),
            "lastHeartbeat" :
ISODate("2020-08-10T09:13:16.373Z"),
            "lastHeartbeatRecv" :
ISODate("2020-08-10T09:13:15.921Z"),
            "pingMs" : NumberLong(0),
            "lastHeartbeatMessage" : "",
            "syncingTo" : "localhost:27019",
            "syncSourceHost" : "localhost:27019",
            "syncSourceId" : 2,
            "infoMessage" : "",
            "configVersion" : 1
        },
        {
            "_id" : 1,
            "name" : "localhost:27018",
            "health" : 1,
            "state" : 2,
            "stateStr" : "SECONDARY",
            "uptime" : 5769,
            "optime" : {
                "ts" : Timestamp(1597050767, 1),
                "t" : NumberLong(2)
            },
```

```
            "optimeDate" :
ISODate("2020-08-10T09:12:47Z"),
            "syncingTo" : "localhost:27017",
            "syncSourceHost" : "localhost:27017",
            "syncSourceId" : 0,
            "infoMessage" : "",
            "configVersion" : 1,
            "self" : true,
            "lastHeartbeatMessage" : ""
        },
        {
            "_id" : 2,
            "name" : "localhost:27019",
            "health" : 1,
            "state" : 1,
            "stateStr" : "PRIMARY",
            "uptime" : 4545,
            "optime" : {
                "ts" : Timestamp(1597050787, 1),
                "t" : NumberLong(2)
            },
            "optimeDurable" : {
                "ts" : Timestamp(1597050787, 1),
                "t" : NumberLong(2)
            },
            "optimeDate" :
ISODate("2020-08-10T09:13:07Z"),
            "optimeDurableDate" :
ISODate("2020-08-10T09:13:07Z"),
```

```
            "lastHeartbeat" :
ISODate("2020-08-10T09:13:16.582Z"),
            "lastHeartbeatRecv" :
ISODate("2020-08-10T09:13:16.926Z"),
            "pingMs" : NumberLong(0),
            "lastHeartbeatMessage" : "",
            "syncingTo" : "",
            "syncSourceHost" : "",
            "syncSourceId" : -1,
            "infoMessage" : "",
            "electionTime" :
Timestamp(1597049135, 2),
            "electionDate" :
ISODate("2020-08-10T08:45:35Z"),
            "configVersion" : 1
        }
    ],
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1597050787,
1),
        "signature" : {
            "hash" :
BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA
AA="),
            "keyId" : NumberLong(0)
        }
    },
    "operationTime" : Timestamp(1597050767, 1)
```

}

- Go to one of the Primary mongo and write use admin . Now db.shutDownServer() .
- Just check the `rs.status()` on any running server and you will see that the server which was primary is having stateStr as “Not Reachable” and some one else has become a primary.
- For getting a server to be elected as primary there must be odd number of voting members present in the RS . Other wise primary can’t be elected .
- If there are 5 members including primary then a new primary can be elected only if 2 members dies . And now in 3 1 is Primary and 2 are secondary . Now if again primary dies now Primary cannot be elected .
-
-
- Read Preference :::
- Primary : Read from Primary Only(Default)
- Primary Preferred : Read from Primary but if not avaiable read from secondary .
- Similarly Secondary and Secondary Preffered
- Near (Read which is near by network latency)
- Connect to many RS members by application to reduce the possibility of failure.
-
- Best Practices
- Don’t Create RS with 2 members . If one dies no

voting will be there and complete RS will breakdown .

- If you can’t afford to have 3 members then add 2 normal and 1 arbiter
- Always have odd number of voting members .
- Max members possible are 50 .
- Max voting members possible are 7 . (3, 5, 7)
- Always have odd number of members having voting rights .
- IN a 5 member RS you can have 3, 5 members having voting rights and in 7 you can have 3 , 5, 7 voting members.
-
- Sharding :::
- Setup two replica sets as above
-
- For setting up Config server replica set ensure that config server replica set must not contain arbiters , no delayed members and all must build indexes .
- mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/configsvr/db3 --port 29030 --replSet configsvr --configsvr (This must be specified while starting instances)
- While creating config variable in config server mongo shell also pass `configsvr : true ` along with _id and members.
-
- Start mongos now with below command:: mongos --configdb configsvr/localhost:29018,localhost:29019,localhost:29030
- Now open a new terminal and type mongo
-
- Now call sh.addShard(‘rs0/localhost:27018’) and sh.addShard(‘rs1/localhost:28018’)

- 503 35582 35431 0 3:45PM ttys002 0:33.55 mongod --shardsvr dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/shard1/db1 --port 27018 --replSet rs0 —shardsvr
- 503 35596 35592 0 3:46PM ttys003 0:34.82 mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/shard1/db2 --port 27019 --replSet rs0 —shardsvr

503 35601 35598 0 3:46PM ttys004 0:34.51 mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/shard1/db3 --port 27030 --replSet rs0 —shardsvr

503 35830 35827 0 3:49PM ttys005 0:32.42 mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/shard2/db2 --port 28019 --replSet rs1 —shardsvr
503 36274 35832 0 3:55PM ttys007 0:30.08 mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/shard2/db3 --port 28030 --replSet rs1 —shardsvr
503 35819 35678 0 3:49PM ttys008 0:31.66 mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/shard2/db1 --port 28018 --replSet rs1 —shardsvr
503 35913 35842 0 3:50PM ttys009 0:00.37 mongo --port 27018
503 36054 36051 0 3:52PM ttys010 0:00.38 mongo --port 28018
503 36987 36832 0 4:05PM ttys011 0:24.35 mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/configsvr/db1 --port 29018 --replSet configsvr --configsvr
503 36995 36992 0 4:05PM ttys012 0:24.03 mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/configsvr/db2 --port 29019 --replSet configsvr --configsvr
503 37066 36997 0 4:06PM ttys013 0:23.62 mongod --dbpath /Users/vaneetkumar/data/mongocluster/shardedcluster/configsvr/db3 --port 29030 --replSet configsvr --configsvr
503 37413 37071 0 4:11PM ttys014 0:00.32 mongo --port 29018
503 38357 38066 0 4:24PM ttys015 0:01.58 mongos --configdb configsvr/localhost:29018,localhost:29019,localhost:29030
503 38368 38362 0 4:25PM ttys016 0:00.33 mongo
503 39903 39899 0 4:47PM ttys017 0:00.00 grep mongo

- Create two databases and add 10000 rows in use collection in both the databases . Check status by sh.status() on mongos you will see that both the databases have their primary shard defined . It means data has not been sharded yet and the database’s data is physically present on one of the replica set .
- It means every database has its primary shard . All the collections by default live in primary shard only . But once you shard one or more collections they are stored across multipe shards . Non

shared collection still stay with their primary shard only .

- Now to shard the data :::
- Enable sharing on a database from mongos by `sh.enableSharding('dbName')` .
- After that run sh.status() on mongos and you will get : { "_id" : "sharded", "primary" : "rs0", "partitioned" : true, "version" : { "uuid" : UUID("60f5a8fc-0a5d-4fd3-8d26-7b6613912864"), "lastMod" : 1 } } here partitioned is true but for nonsharded it is false .
-
-
- Mongo DB works on Balancing algorithm . Its not like Cassandra that every document will be directly written to a shard calculated after taking hash of its shard key value . It works on Dara chunking . Chunk size ranges from 1 to 1024 MB . By default chunk size is 64 MB . When we start inserting data in a collection and we shard it by some key like *sh.shardCollection('sharded.user' , {_id : 1 })* . Then data keeps on inserting on primary shard until a chunk of 64 MB is created . Once chunk size exceeds 64 MB it is split into equal chunks on various shards . But to see balancing happening we need to reduce the chunk size so that we can see it practically . Otherwise we will have to insert records of 64 MB . We can change it by below query
- db.settings.save({ _id:"chunksize", value: 1 })
- Now Insert 20000 documents in use collection and then issue *sh.status()* Now you will be

able to see { "_id" : { "$minKey" : 1 } } -->> { "_id" : 1 } on : rs0 Timestamp(3, 0) .

{ "_id" : 1 } -->> { "_id" : 15198 } on : rs1 Timestamp(3, 1)

{ "_id" : 15198 } -->> { "_id" : { "$maxKey" : 1 } } on : rs1 Timestamp(2, 3)

- The above result means that Jumbo chunk has been created on rs1 and now whatever new data is inserted will be inserted in rs1 only .
- You can see after going to rs0 that it has only one recored with _id : 0 value .
-
-
- Hashed Sharding :::
- for(var i = 0 ; i< 10000 ; i++) {db.hashuser.insert({_id : i , name : "random"+i , phonenumber: "87429102"+i})}
- db.hashuser.ensureIndex({name: "hashed"})
- sh.shardCollection('sharded.hashuser' , {name : 'hashed'})
- Now add more 1000 documents
- sh.status() will give
- chunks: rs01

rs1 2

{ "name" : { "$minKey" : 1 } } -->>{ "name" :NumberLong("-4020259338504487 00") } on : rs0 Timestamp(3, 0)

{ "name" : NumberLong("-4020259338504487 00") } -->> { "name" : NumberLong("8354342865592419461") } on : rs1 Timestamp(3, 1)

{ "name" : NumberLong("8354342865592419461") } -->> { "name" : { "$maxKey" : 1 } } on : rs1 Timestamp(2, 3)

```
Last login: Thu Aug 13 05:01:45 on ttys017
vaneetkumar@197NODMB26129 ~ % ps -aef | grep java
  503 42190 42186   0  5:03AM ttys000    0:00.00 grep java
vaneetkumar@197NODMB26129 ~ % ps -aef | grep mongo
  503 34996     1   0  3:34AM ??         1:41.42 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/shard1/db1 --port 27018 --
replSet rs0 --shardsvr
  503 37218     1   0  4:06AM ??         0:46.47 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/shard3/db1 --port 30018 --
replSet rs2 --shardsvr
  503 37362     1   0  4:08AM ??         0:39.64 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/shard4/db --port 31018 --
replSet rs3 --shardsvr
  503 37843     1   0  4:16AM ??         0:41.14 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/shard1/db2 --port 27019 --
replSet rs0 --shardsvr
  503 40850     1   0 Mon05PM ??        18:02.14 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/shard1/db3 --port 27030 --
replSet rs0 --shardsvr
  503 40928     1   0 Mon05PM ??        17:13.61 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/shard2/db2 --port 28019 --
replSet rs1 --shardsvr
  503 41003     1   0 Mon05PM ??        17:57.95 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/shard2/db3 --port 28030 --
replSet rs1 --shardsvr
  503 41008     1   0 Mon05PM ??        19:08.67 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/shard2/db1 --port 28018 --
replSet rs1 --shardsvr
  503 41886     1   0 Mon05PM ??        17:23.99 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/configsvr/db1 --port 29018 --
replSet configsvr --configsvr
  503 41895     1   0 Mon05PM ??        18:02.01 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/configsvr/db2 --port 29019 --
replSet configsvr --configsvr
  503 41970     1   0 Mon05PM ??        18:34.08 mongod --dbpath /Users/
vaneetkumar/data/mongocluster/shardedcluster/configsvr/db3 --port 29030 --
replSet configsvr --configsvr
  503 42047     1   0 Mon05PM ??         7:29.32 mongos --configdb configsvr/
localhost:29018,localhost:29019,localhost:29030
  503 42192 42186   0  5:03AM ttys000    0:00.01 grep mongo
```