Setup MongoDB ReplicaSets

Two types of configuration is possible with ReplicaSet

1. 1 x Primary + 1 x Secondary (DR) + 1 x Arbiter
2. 1 x Primary + 2 x Secondary (1 in DC and 1 in DR)

In this tutorial, we'll go with type 2 since we require HA along with DR:

Example:

Replica Set Member	Hostname
Member 0	rs0.mongo-replicaset.com
Member 1	rs1.mongo-replicaset.com
Member 2	rs2.mongo-replicaset.com

Installing MongoDB

1. We'll store the binaries under `/app.` On the target machine, we enter the directory:

```
cd /app
```

2. Download the Tar ball (Get the download URL from https://www.mongodb.com/try/download/community)

```
wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel80-4.4.16.tgz
```

3. Extract the tarball

```
tar -xzvf mongodb-linux-x86_64-rhel80-4.4.16.tgz
```

4. Change Directory name to MongoDB

```
mv mongodb-linux-x86_64-rhel80-4.4.16 mongodb
```

5. Add MongoDB User

```
useradd mongod
```

6. We create directory where MongoDB will store its data

```
 mkdir -p /app/mongodb/data
mkdir -p /app/log/mongodb
```

7. Change folder permissions

```
 chown -R mongod:mongod /app/mongodb
chown -R mongod:mongod /app/log/mongodb
```

Configuring MongoDB instances

1. Start each member of the replica set with the appropriate options in `/app/mongodb/mongod.conf`:

```
 storage:
   dbPath: "/app/mongodb/data"
   journal:
     enabled: true
systemLog:
   destination: file
   path: "/app/log/mongodb/mongod.log"
   logAppend: false
   logRotate: rename
replication:
   replSetName: "rs0"
net:
   bindIp: localhost,0.0.0.0
   port: 27027
```

2. Create the MongoDB Service

```
 [Unit]
Description=MongoDB
After=syslog.target network.target

[Service]
Type=simple

User=mongod
Group=mongod

ExecStart=/app/mongodb/bin/mongod --config /app/mongodb/mongod.conf

[Install]
WantedBy=multi-user.target
```

3. Reload the Systemctl Daemon

```
sudo systemctl daemon-reload
```

4. Allow MongoDB in SELinux to listen to non default port

```
sudo semanage port -a -t mongod_port_t -p tcp 27027
```

5. MongoDB Service Commands

```
sudo systemctl start mongod
```

```
sudo systemctl status mongod
```

```
sudo systemctl enable mongod
```

Connecting and Configuring ReplicaSet

1. Connect a mongo shell to one of the mongod instances.

```
mongo
```

2. Initiate the replica set.

`Note:- Run rs.initiate() on just one and only one mongod instance for the replica set.`

```
rs.initiate( {
   _id : "rs0",
   members: [
      { _id: 0, host: "rs0.mongo-replicaset.com:27027" },
      { _id: 1, host: "rs1.mongo-replicaset.com:27027" },
      { _id: 2, host: "rs2.mongo-replicaset.com:27027" }
   ]
})
```

3. View the replica set configuration.

Use rs.conf() to display the replica set configuration object:

```
rs.conf()
```

The replica set configuration object resembles the following:

```
{
   "_id" : "rs0",
   "version" : 1,
   "protocolVersion" : NumberLong(1),
   "members" : [
      {
         "_id" : 0,
         "host" : "rs0.mongo-replicaset.com:27027",
         "arbiterOnly" : false,
         "buildIndexes" : true,
         "hidden" : false,
         "priority" : 1,
         "tags" : {

         },
         "slaveDelay" : NumberLong(0),
         "votes" : 1
      },
      {
         "_id" : 1,
         "host" : "rs1.mongo-replicaset.com:27027",
         "arbiterOnly" : false,
         "buildIndexes" : true,
         "hidden" : false,
         "priority" : 1,
         "tags" : {

         },
         "slaveDelay" : NumberLong(0),
         "votes" : 1
      },
      {
         "_id" : 2,
         "host" : "rs2.mongo-replicaset.com:27027",
         "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,
      "getLastErrorModes" : {

      },
      "getLastErrorDefaults" : {
         "w" : 1,
         "wtimeout" : 0
      },
      "replicaSetId" : ObjectId("585ab9df685f726db2c6a840")
   }
}
```

4. Ensure that the replica set has a primary.

Use rs.status() to identify the primary in the replica set.

```
rs.status()
```

Force a Member to be Primary Using Database Commands

Setting Priority (from the PRIMARY)

```
 cfg = rs.conf()
cfg.members[0].priority = 0.5
cfg.members[1].priority = 0.5
cfg.members[2].priority = 1
rs.reconfig(cfg)
```

Force a Member to be Primary by Setting its Priority High 1. Freeze the other secondary so that it doesnt become primary.

```
rs.freeze(120)
```

2. Make the current Primary step down

```
rs.stepDown(120)
```

Setting up DNS

Record Type | Record Name | IP | | ------------- | ------------- | ------------- | |A | rs0.mongo-replicaset.com| 40.0.0.1 | |A | rs1.mongo-replicaset.com| 40.0.0.2 | |A | rs2.mongo-replicaset.com| 40.0.0.3 | |SRV | _mongodb._tcp.rs.mongo-replicaset.com | 0 0 27027 rs0.mongo-replicaset.com
0 0 27027 rs1.mongo-replicaset.com
0 0 27027 rs2.mongo-replicaset.com | |TXT | rs | authSource=admin&replicaSet=rs|

Now access the cluster with the following URL:

```
mongodb+srv://rs.mongo-replicaset.com/dbname
```

This URL automatically translates to:

```
mongodb://rs0.mongo-replicaset.com:27027,rs1.mongo-replicaset.com:27027,rs2.mongo-replicaset.com:27027/
```

Ref: https://www.mongodb.com/developer/article/srv-connection-strings/

Write Concern for Replica Sets

Write concern for replica sets describe the number of data-bearing members (i.e. the primary and secondaries, but not arbiters) that must acknowledge a write operation before the operation returns as successful. A member can only acknowledge a write operation after it has received and applied the write successfully.

[For Synchronous Replication] For replica sets, the write concern of `w: "majority"` requires acknowledgement that the write operations have propagated to a calculated majority of the data-bearing voting members. For most replica set configurations, w: "majority" is the default write concern. To learn how MongoDB determines the default write concern, see Implicit Default Write Concern.

[For Semi-Syncronous Replication] Write operations with a write concern of `w: 1` require that only the primary replica set member acknowledge the write before returning write concern acknowledgment. You can specify an integer value greater than 1 to require acknowledgment from the primary and as many secondaries as needed to meet the specified value, up to the total number of data-bearing members in the replica set.

[For Asynchronous Replication] Write operations with a write concern of `w: 0` require that no replica set member acknowledge the write before returning write concern acknowledgment. Requests no acknowledgment of the write operation.

Modify the default MongoDB ReplicaSets WriteConcern using the following link:
https://www.mongodb.com/docs/manual/reference/command/setDefaultRWConcern/#mongodb-dbcommand-dbcmd.setDefaultRWConcern