Stanbic IBTC Bank Plc

MongoDB Consult Report

Sweta Attri<sweta.attri@mongodb.com>, MongoDB Inc.
May 2022

Participants:

- Ernest Oduba, Technical Project Manager, Stanbic IBTC Bank Plc
- Michael Okwuwa, Database Administrator, Stanbic IBTC Bank Plc
- Sweta Attri, Consulting Engineer, MongoDB

This document summarizes the installation of Ops Manager from a 5 days remote consultation with Stanbic IBTC Bank Plc 25 April 2022 to 28 March 2022 and 16 May 2022.

1 Background

The Stanbic IBTC Bank Plc team wanted to deploy an Ops Manager to monitor, backup and automate their current running Production Environment in MongoDB.

2 Goals of the Consulting Engagement

The goal of this consulting engagement is to deploy an Ops Manager for the team's Production Environment. The Production environment for the Stanbic IBTC Bank's team currently has 1 running replica set consisting of 4 nodes: 2 Production Nodes and 2 DR nodes. The 4 nodes are deployed in the same region but 2 Production Nodes are in a different DC from the DR nodes. Each node is running on RedHat OS v7.7 Enterprise.

3 Ops Manager Installation

3.1 Ops Manager Sizing

Current infrastructure of the Ops Manager is for the Production environment and the team wanted to deploy it with the minimum configuration. As per the sizing ticket below minimum configuration was recommended to the team:

Server	CPU	RAM	Disk	Components	OS
Server 1	4	32	504	OM Application Server 1, Backup Daemon	RedHat v7.7
Server 2	4	16	800	OplogDB + BlockStore, AppDB	RedHat v7.7
Server 3	4	16	800	OplogDB + BlockStore, AppDB	RedHat v7.7
Server 4	4	16	800	OplogDB + BlockStore, AppDB	RedHat v7.7

Before we jump on the discussion for best practices and steps for Installation of Ops Manager, let's understand the various components of Ops Manager:

Ops Manager Application Server:

The main Ops Manager component. The Ops Manager Application provides the user interface for managing MongoDB deployments and provides endpoints for MongoDB Agents to transmit data.

Application DB:

The dedicated set of MongoDB databases that store metadata for the Ops Manager installation and the managed MongoDB deployments.

Backup Daemon:

The Ops Manager component that creates and manages backups by maintaining head databases and snapshots.

Oplog Store:

The database where Ops Manager stores oplog slices before applying them to a deployment's backup.

Ref: Glossary for Ops Manager Components

3.2 Prepare the nodes for the Ops Manager

All 4 nodes will have MongoDB processes installed. These disks are mounted and configured in the */etc/fstab* for mounting on OS boot. A complete manual of production notes can be found here, and an operation checklist can be found here.

A complete script can be found after the below points that will help implement the production notes on the system.

- As the mount point for the data drive was changed from */opt/mongodb/mms* to */opt/mongodb*, remove the entry from the previous mount point in the */etc/fstab* file.
- Configure Swap Space.
- Install numactl (installed it along with the enterprise dependencies).
- Set vm.swappiness from 30 to 1.
- Set vm.max_map_count from 65,536 to 128000 (double of max user processes).
- Set net.ipv4.tcp_keepalive_time from 7200 to 120.
- Set kernel.pid_max from 32768 to 64000.
- Add all the parameters in */etc/sysctl.conf* so they are set on reboot.
- Disable Transparent Huge Pages.
- Configure the Readahead for */dev/nvme1n1* from 256 to 8 (recommended value is between 8 and 32).

```
# Production notes
mount | grep /data/mongodb | grep -q xfs && echo "Disk OK" || echo "/data/mongodb
Disk not mounted with XFS"
```

```
grep -q /dev/nvme1n1 /etc/fstab || echo "/dev/nvme1n1       /opt/mongodb
xfs     defaults,noatime   0  0" | sudo tee --append /etc/fstab

#Below command must not return any errors
sudo findmnt --verify

#Numa - CPU RAM access
grep -q 'vm.zone_reclaim_mode' /etc/sysctl.conf || echo "vm.zone_reclaim_mode=0
" | sudo tee --append /etc/sysctl.conf
sudo sysctl -w  vm.zone_reclaim_mode=0

#SWAP Usage
grep -q 'vm.swappiness' /etc/sysctl.conf || echo "vm.swappiness=1" | sudo tee
--append /etc/sysctl.conf
sudo sysctl -w  vm.swappiness=1

#Configure sufficient kernel pid limit
sudo grep -q 'kernel.pid_max' /etc/sysctl.conf || echo "kernel.pid_max=64000" |
sudo tee --append /etc/sysctl.conf
sudo sysctl -w kernel.pid_max=64000

#Modify keepalive time
sudo grep -q 'net.ipv4.tcp_keepalive_time' /etc/sysctl.conf || echo
"net.ipv4.tcp_keepalive_time=120" | sudo tee --append /etc/sysctl.conf
sudo sysctl -w net.ipv4.tcp_keepalive_time=120

#Configure sufficient file-handles
sudo grep -q 'fs.file-max' /etc/sysctl.conf || echo "fs.file-max=98000" | sudo tee
--append /etc/sysctl.conf
sudo sysctl -w fs.file-max=98000

#Configure maximum threads per process
sudo grep -q 'kernel.threads-max' /etc/sysctl.conf || echo
"kernel.threads-max=64000" | sudo tee --append /etc/sysctl.conf
sudo sysctl -w kernel.threads-max=64000

#Configure maximum number of memory map areas per process
sudo grep -q 'vm.max_map_count' /etc/sysctl.conf || echo "vm.max_map_count=128000"
| sudo tee --append /etc/sysctl.conf
sudo sysctl -w vm.max_map_count=128000

#User Resource Limit
#fsize - file size
for limit in fsize cpu as memlock
do
  grep "mongodb" /etc/security/limits.conf | grep -q $limit || echo -e "mongod
hard   $limit    unlimited\nmongod     soft    $limit   unlimited" | sudo tee
--append /etc/security/limits.conf
done
#nofile - open files
for limit in nofile noproc
do
```

```
  grep "mongodb" /etc/security/limits.conf | grep -q $limit || echo -e "mongod
hard   $limit    64000\nmongod     soft    $limit   64000" | sudo tee --append
/etc/security/limits.conf
done


#configure THP and readahead
SCRIPT=$(cat << 'ENDSCRIPT'
#!/bin/bash
### BEGIN INIT INFO
# Provides:          disable-transparent-hugepages
# Required-Start:    $local_fs
# Required-Stop:
# X-Start-Before:    mongod mongodb-mms-automation-agent
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Disable Linux transparent huge pages
# Description:       Disable Linux transparent huge pages, to improve
#                    database performance.
### END INIT INFO
case $1 in
  start)
    if [ -d /sys/kernel/mm/transparent_hugepage ]; then
      thp_path=/sys/kernel/mm/transparent_hugepage
    elif [ -d /sys/kernel/mm/redhat_transparent_hugepage ]; then
      thp_path=/sys/kernel/mm/redhat_transparent_hugepage
    else
      return 0
    fi
    echo 'never' > ${thp_path}/enabled
    echo 'never' > ${thp_path}/defrag
    re='^[0-1]+$'
    if [[ $(cat ${thp_path}/khugepaged/defrag) =~ $re ]]
    then
      # RHEL 7
      echo 0  > ${thp_path}/khugepaged/defrag
    else
      # RHEL 6
      echo 'no' > ${thp_path}/khugepaged/defrag
    fi
    # Set Readahead for Data Disk, depends on the nodes, 3 versions
    # Server 1 2 3
   # blockdev --setra 8 </where-the-data-directory-mounted>
    unset re
    unset thp_path
    ;;
esac
ENDSCRIPT
)
echo "$SCRIPT" | sudo tee /etc/init.d/disable-transparent-hugepages
sudo chmod 755 /etc/init.d/disable-transparent-hugepages
sudo chkconfig --add disable-transparent-hugepages
```

```
sestatus

sudo setenforce 0

# disable in selinux config
sudo sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config

sudo reboot
```

On the current systems SELINUX was enabled. After discussion with the team it was decided to disable SELINUX using the below command:

```
sestatus

sudo setenforce 0

# disable in selinux config
sudo sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

3.3 Create separate mountpoints for AppDB and BackupDB

Since, both the AppDB and BackupDB replica sets will be installed on the same servers (Server - 2,3,4), 2 separate mount points were created to efficiently utilize disk IOPS :

- **/data** (200GB) : for AppDB
- **/backup** (400GB) : for backupDB

3.4 Install AppDB for the Ops Manager

The AppDB will be installed on each of the 3 servers (Server 2, 3, 4). The complete installation procedure for MongoDB on CentOS can be found here: [Install MongoDB Enterprise Edition on Red Hat or CentOS — MongoDB Manual](). We first start with one of the nodes.

1. Install the dependencies.

```
sudo yum install cyrus-sasl cyrus-sasl-gssapi cyrus-sasl-plain krb5-libs
libcurl net-snmp openldap openssl xz-libs
```

2. Create a `/etc/yum.repos.d/mongodb-enterprise-5.0.repo` file so that you can install MongoDB enterprise directly using yum.

```
[mongodb-enterprise-5.0]
name=MongoDB Enterprise Repository
baseurl=https://repo.mongodb.com/yum/redhat/$releasever/mongodb-enterprise/5
.0/$basearch/
gpgcheck=1
enabled=1
```

```
gpgkey=https://www.mongodb.org/static/pgp/server-5.0.asc
```

3. Install the MongoDB Enterprise packages.

```
sudo yum install -y mongodb-enterprise
```

4. Create Data and Log Directories to host the MongoDB Data and Log Files.

```
mkdir /data/log/mongodb
mkdir /data/lib/mongo

sudo chown mongod:mongod -R /data/log/
sudo chown mongod:mongod -R /data/lib/
```

5. Create a Keyfile for Internal Authentication.

```
openssl rand -base64 756 | sudo tee /data/keyfile

sudo chmod 400 /data/keyfile
sudo chown mongod:mongod /data/keyfile
```

6. The following configuration file should be used to start AppDB and stored in the default `/etc/mongod.conf`.

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /data/log/mongodb/mongod.log

# Where and how to store data.
storage:
  dbPath: /data/lib/mongo
  journal:
    enabled: true
#  engine:
  wiredTiger:
    engineConfig:
      cacheSizeGB: 8

# how the process runs
```

```
processManagement:
  fork: true  # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid  # location of pidfile
  timeZoneInfo: /usr/share/zoneinfo

# network interfaces
net:
  port: 27018
  bindIp: 0.0.0.0  # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses
or, alternatively, use the net.bindIpAll setting.

security:
  keyFile: /data/keyfile
  authorization: enabled

#operationProfiling:

replication:
  replSetName: appdbRS

#sharding:

## Enterprise-Only Options

#auditLog:

#snmp:
```

7. Start mongod service.

```
sudo systemctl start mongod
```

8. Initiate the BackupDB Replica Set and create a root user and opsmanager user.

```
mongo

rs.initiate()

use admin
// root user
db.createUser({
       user: "root",
       pwd: "<password>",
       roles: ['root']
});

db.auth('root', '<password>');
db.createUser({
  user: "mms",
  pwd: "<password>",
```

```
  roles: [{
      db: "admin",
      role: "readWriteAnyDatabase"
  }, {
      db: "admin",
      role: "dbAdminAnyDatabase"
  }, {
      db: "admin",
      role: "clusterMonitor"
  }]
})
```

9. Repeat Steps 1-4 and 6 on the other 2 nodes.

10. Copy the Keyfile created in Step 5 to the same directories in the other 2 nodes and assign the same permissions.

11. Start mongod service on the other 2 nodes using Step 9.

12. Login into the AppDB Replica Set and add the other 2 nodes. Confirm the health of the nodes using rs.status().

```
mongo --host appdbRS/<HOSTNAME-2>:27018 -u root --authenticationDatabase
admin -p

rs.add("<HOSTNAME-3>:27018")
rs.add("<HOSTNAME-4>:27018")

rs.status()
```

3.5 Install Oplog DB + BlockStore for Backup

The OplogDB + BlockStore will be installed on each of the 3 servers (Server 2, 3, 4). The complete installation procedure for MongoDB on CentOS can be found here: Install MongoDB Enterprise Edition on Red Hat or CentOS — MongoDB Manual. We first start with one of the nodes.

1. Install the dependencies.

```
sudo yum install cyrus-sasl cyrus-sasl-gssapi cyrus-sasl-plain krb5-libs
libcurl net-snmp openldap openssl xz-libs
```

2. Create a `/etc/yum.repos.d/mongodb-enterprise-5.0.repo` file so that you can install MongoDB enterprise directly using yum.

```
[mongodb-enterprise-5.0]
name=MongoDB Enterprise Repository
baseurl=https://repo.mongodb.com/yum/redhat/$releasever/mongodb-enterprise/5
.0/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-5.0.asc
```

3. Install the MongoDB Enterprise packages.

```
sudo yum install -y mongodb-enterprise
```

4. Create Data and Log Directories to host the MongoDB Data and Log Files.

```
mkdir /backup/log/mongodb
mkdir /backup/lib/mongo

sudo chown mongod:mongod -R /backup/log/
sudo chown mongod:mongod -R /backup/lib/
```

5. Create a Keyfile for Internal Authentication.

```
openssl rand -base64 756 | sudo tee /backup/keyfile

sudo chmod 400 /data/keyfile
sudo chown mongod:mongod /backup/keyfile
```

6. The following configuration file should be used to start backupDB and stored in the default `/etc/mongod_backup.conf`.

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /backup/log/mongodb/mongod.log

# Where and how to store data.
storage:
  dbPath: /backup/lib/mongo
  journal:
    enabled: true
```

```
#  engine:
  wiredTiger:
    engineConfig:
      cacheSizeGB: 8

# how the process runs
processManagement:
  fork: true  # fork and run in background
  pidFilePath: /var/run/mongodb/mongod_backup.pid  # location of pidfile
  timeZoneInfo: /usr/share/zoneinfo

# network interfaces
net:
  port: 27019
  bindIp: 0.0.0.0  # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses
or, alternatively, use the net.bindIpAll setting.

security:
  keyFile: /backup/keyfile
  authorization: enabled

#operationProfiling:

replication:
  replSetName: backupdbRS

#sharding:

## Enterprise-Only Options

#auditLog:

#snmp:
```

7. Copy the service control file **/usr/lib/systemd/system/mongod.service** to **/etc/systemd/system/backupdb.service**

```
cp /usr/lib/systemd/system/mongod.service
/etc/systemd/system/backupdb.service
```

8. Edit **/etc/systemd/system/backupdb.service** to change the value of these parameters following the table below:

Old Value	**New Value**
Environment="OPTIONS=-f /etc/mongod.conf"	Environment="OPTIONS=-f **/etc/mongod_backup.conf"**
ExecStartPre=/usr/bin/mkdir -p	ExecStartPre=/usr/bin/mkdir -p

Old Value	New Value
/var/run/mongodb	**/var/run/backupdb**
ExecStartPre=/usr/bin/chown mongod:mongod **/var/run/mongodb**	ExecStartPre=/usr/bin/chown mongod:mongod **/var/run/backupdb**
ExecStartPre=/usr/bin/chmod 0755 **/var/run/mongodb**	ExecStartPre=/usr/bin/chmod 0755 **/var/run/backupdb**
PIDFile=/var/run/mongodb/**mongod.pid**	PIDFile=/var/run/oplogdb/**backupdb.pid**

9. Reload services into systemctl table of events to process:

```
systemctl daemon-reload
```

10. Start backupdb service.

```
sudo systemctl start backupdb.service
```

11. Initiate the backupDB Replica Set and create a root user.

```
mongo

rs.initiate()

use admin
// root user
db.createUser({
       user: "root",
       pwd: "<password>",
       roles: ['root']
});
```

12. Repeat Steps 1-4 and 6 on the other 2 nodes.

13. Copy the Keyfile created in Step 5 to the same directories in the other 2 nodes and assign the same permissions.

14. Start mongod service on the other 2 nodes using Step 9.

15. Login into the AppDB Replica Set and add the other 2 nodes. Confirm the health of the nodes using rs.status().

```
mongo --host backupdbRS/<HOSTNAME-2>:27019 -u root --authenticationDatabase
admin -p

rs.add("<HOSTNAME-3>:27019")
rs.add("<HOSTNAME-4>:27019")

rs.status()
```

3.6 Install and Configure Ops Manager

We follow the below steps to install Ops Manager on *<HOSTNAME-1>*.

1. Download the latest version of the Ops Manager RPM package from the MongoDB Download Center.
    a. In the Version dropdown, select the latest version (*5.0.9* at the time of installation).
    b. In the Platform dropdown, select *Red Hat + CentOS 6, 7, 8 / SUSE 12 + 15 / Amazon Linux*.
    c. In the Package dropdown, select *rpm*.
    d. Click "Copy Link".

2. Download and install the Ops Manager package on the server. The below link uses the latest Ops Manager version available at the time of writing.

```
curl -OL
https://downloads.mongodb.com/on-prem-mms/rpm/mongodb-mms-5.0.9.100.20220407
T0303Z-1.x86_64.rpm
sudo rpm -ivh mongodb-mms-*.x86_64.rpm
```

3. We now need to configure Ops Manager to connect to the Application Database. This requires storing the login credentials and other necessary information in the Ops Manager's configuration file. To avoid storing the login credentials as Plain Text, we use the Ops Manager provided credentialstool to encrypt the username and password.

```
sudo /opt/mongodb/mms/bin/credentialstool --username root --password
```

4. Use the username and password generated in the above step to construct the connection string, which is as follows (the encrypted username and password shown here are different than actually generated).

```
mongodb://9ebe5c746447ab519208a3ac6d145e7594dca8db26e598b8fde47623dcf19c1e-6
458c7fce75263d578001efaaf1a535c-bcfb7f05e92ee7ca3a382e41b49e956e:9ebe5c74644
7ab519208a3ac6d145e7594dca8db26e598b8fde47623dcf19c1e-24021e1b0e6763586a0c91
e376a78468-c31563c11f04f73c9693c9cb9eff1cbc@<HOSTNAME-2>:27018,<HOSTNAME-3>:
```

```
27018,<HOSTNAME-4>:27018/?maxPoolSize=150
```

5. Edit the configuration file `/opt/mongodb/mms/conf/conf-mms.properties` and modify the following fields. Replace *$URI* with the above-formulated connection string:

```
mongo.mongoUri=$URI
mongo.encryptedCredentials=true
```

6. Start Ops Manager.

```
sudo systemctl start mongodb-mms
```

7. Verify the status of Ops Manager using the below command. It should successfully finish Pre-Flight Checks and enable the Backup Daemon.

```
sudo systemctl status mongodb-mms
```

8. Open the Ops Manager home page from a web browser. The team configured a DNS entry *<HOSTNAME-1>.com:8080*, which would serve as the address of the Load Balancer. It was initially pointed to *prod-<HOSTNAME-1>.com* only.

```
http://<HOSTNAME-1>.com:8080
```

9. Register your first user by clicking on the *Register* link. The first user is automatically assigned the Global Owner role.

10. Configure Ops Manager with your preferred settings over the next set of configuration pages. These settings can be changed later, but it's important to give the correct details at the initial stage to avoid issues later.
11. Since none of the VMs will be having Internet Access, we configured Ops Manager in Local Mode.

Installer Download Source	Local
Versions Directory	/opt/mongodb/mms/mongodb-releases
Backup Versions Auto Download	true
Backup Versions Auto Download Enterprise Builds	true
Required Module for Backup	Enterprise Required

3.6.1 Configure Ops Manager for Local

As mentioned in Section 3.6 Step 11, none of the VMs will be having Internet Access. Therefore, we have configured Ops Manager in Local Mode. To go forward with this installation and to be able to automate existing or create new deployments we need to place all the mongodb binaries in the Version Directory : **/opt/mongodb/mms/mongodb-releases** on Server 1.

Below binaries were downloaded and placed at the version directory:

- https://downloads.mongodb.com/linux/mongodb-linux-x86_64-enterprise-rhel70-4.4.13.tgz
- https://downloads.mongodb.com/linux/mongodb-linux-x86_64-enterprise-rhel70-5.0.8.tgz
- https://downloads.mongodb.com/linux/mongodb-linux-x86_64-enterprise-rhel70-4.2.1.tgz
- https://fastdl.mongodb.org/tools/db/mongodb-database-tools-rhel70-x86_64-100.5.2.tgz
- https://fastdl.mongodb.org/tools/db/mongodb-database-tools-rhel70-x86_64-100.5.1.tgz

Change ownership and permissions of all the files in the version directory:

```
sudo chown -R mongodb-mms:mongodb-mms /opt/mongodb/mms/mongodb-releases/*
sudo chmod -R 640 /opt/mongodb/mms/mongodb-releases/*.tgz
```

Once completed, update the Version manifest. Please refer to the documentation for details.

3.6.2 Configure Ops Manager for High Availability

The Ops Manager is currently installed on a single host only. This means that if this host goes down, we won't be able to access the Ops Manager application, even though the backing databases are deployed as replica sets (thus highly available).

To configure high availability for Ops Manager, we need to install the Ops Manager application on a second host (in the team's case: *prod-<HOSTNAME-2>.com*) and connect it to the Application Database. We then configure a Load Balancer to balance between the pool of Ops Manager Application hosts. The requirements for the Load Balancer are as follows:

- No need for persistent/sticky connections: The Ops Manager Application's components are stateless between requests. Any Ops Manager Application server can handle requests as long as all the servers read from the same Ops Manager Application Database.
- Client timeouts should be set to 600 seconds.
- Configured with the Round Robin Algorithm (or any algorithm that doesn't prefer a single host).
- Must support Layer 7 (the Application Layer) of the OSI model.
- Must not return cached content.

Follow the below steps to configure High Availability for Ops Manager:

1. Repeat Steps 1, 2, 4, 5 of Section 3.3 Install and Configure Ops Manager on the 2nd server. Do **not** generate encrypted credentials again as per Step 3. Use the **same** encrypted credentials generated before in the *conf-mms.properties* configuration file.

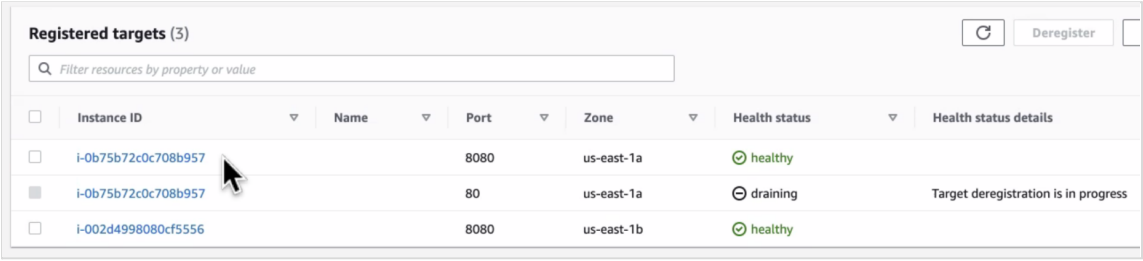
2. Copy the *gen.key* file to the 2nd server. This file is used to encrypt and decrypt the generated credentials.

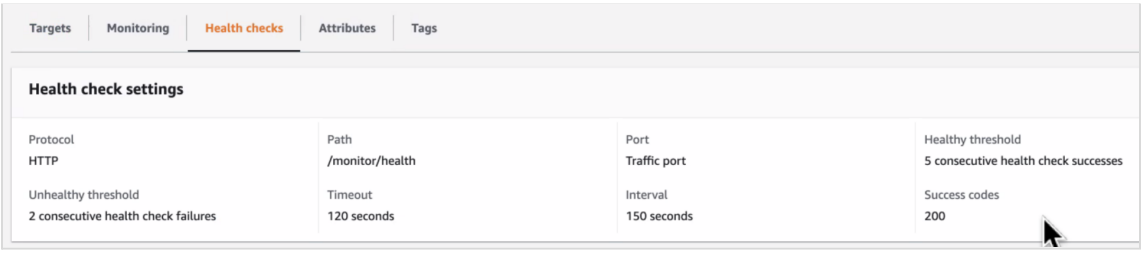
```
#Execute this on the 1st host. Keep the output safe

sudo cat /etc/mongodb-mms/gen.key | base64

#Execute this on the 2nd host

echo <OUTPUT_FROM_ABOVE> | base64 -d | sudo tee /etc/mongodb-mms/gen.key
sudo chown mongodb-mms:mongodb-mms /etc/mongodb-mms/gen.key
sudo chmod 600 /etc/mongodb-mms/gen.key
```

3. Repeat Steps 6 and 7 of Section 3.3 Install and Configure Ops Manager on the 2nd server.

4. Configure the Load Balancer to point to the 2nd host as well. Use the **same DNS** (*dr-<HOSTNAME-1>.com*) configured in Step 8 of Section 3.6 Install and Configure Ops Manager for the Load Balancer URL. Also configure the load balancer to perform a health check on each Ops Manager health API endpoint as per this step in the documentation.

   For the Load Balancer, the team configured AWS ELB, which:
   a. auto-generated a DNS A record for the LB URL
      i. We configured a DNS CNAME record on *dr-<HOSTNAME-1>.com* pointing to the Load Balancer URL
   b. Listened on port 443
   c. Generated a TLS certificate using AWS Certificate Manager
      i. The TLS certificate on the Load Balancer was configured incorrectly, so we had to disable certificate validation from the Agent until this was fixed.
   d. Included the 2 OM instances on port 8080 as the Target Groups
   e. Configured a health check on */monitor/health*.





5. Configure Ops Manager to use the load balancer as per this step in the documentation.

    a. Because the Load Balancer listener was configured on port 443, we had to change the *URL to Access Ops Manager* as follows:

    ```
    https://<HOSTNAME-1>:443
    ```

    b. We used X-Forwarded-Host as the Load Balancer Remote IP Header.

6. Restart both the Ops Manager instances individually to successfully enable the new settings. Wait for one of the instances to successfully start before moving on to the next.

    ```
    sudo systemctl restart mongodb-mms
    ```

After the load balancer is configured and started, you **cannot** log in to the Ops Manager Application from its *individual* host URLs.

3.7 Configure Backups

While we must have deployed the MongoDB process for the Oplog Store, we still haven't configured Ops Manager (specifically the Backup Daemon running on the Ops Manager host) to use them for the Backup Process. The following sections explain the process.

3.7.1 Configure Head Directory

The backup daemon maintains a head database for each shard or replica set it backs up and creates periodic snapshots. The Daemon stores the head databases in the head directory (note that head databases are not used in MongoDB v4.2+, but the facility has been left to support previous versions).

The head database directory will need to be created on the servers running the Backup daemon (even if not used, Ops Manager will still require this). Follow the below steps for the same:

1. Go to *Admin -> Backup* tab. This will show the *Backup Initial Configuration* page as Backups haven't been configured yet.

2. The Backup Daemon's location will be shown. Create the head directory at this location. In the team's case, this was done under the directory */data/head* on the server running Ops Manager.

```
sudo mkdir /data/head
sudo chown mongodb-mms:mongodb-mms /opt/mongodb/backup/heads
```

3. Enter the same directory path on the Ops Manager UI and click on *Set*.

4. Once the Ops Manager finds the directory, it will show the available space in it. Click *Enable Daemon* to configure the database.

5. You will now be presented with 3 options — Create New Blockstore, Create New S3 Blockstore, and Create New Filesystem Store.
    a. You can just scroll down to the bottom and click on *Advanced Options* to separately configure Oplog Store and Blockstore.

The Backup Initial Configuration is now complete. Follow the steps in the next sections to complete the Backup configuration.

3.7.2 Configure Oplog Store

We first configured the Oplog Store.

1. Go to *Admin -> Backup -> Oplog Storage*.
2. Click *Create New Oplog Store*.
3. Provide the Oplog Store details. Any other options can be used as seen fit.
    a. Name: OplogDB
    b. Datastore Type: Replica Set
    c. MongoDB Hostnames: <HOSTNAME-2>:27019,<HOSTNAME-2>:27019,<HOSTNAME-3>:27019.
    d. Username: mms-automation (The user created for the Agent)
    e. Password: The password for the above user.
        i. To get the password, go to *Security -> Authentication & TLS -> Edit Settings*, and continue through until you get to the last page that shows the *mms-automation* user and its *hidden* password. Click on the *Show Password* icon to show the password.

3.7.3 Configure BlockStore backup in Ops Manager

Next we should configure BlockStore storage. We will be using the same as above configuration for the BlockStore.

4 Adding Existing Replica Sets

The team had 1 replica set to be imported in Ops Manager:

Before importing the replica set, it is important to understand that the MongoDB Agent restarts the processes using its own set of MongoDB binaries when importing the database for *automation*.

The following process should be followed:

1. For Ops Manager to manage a deployment, the MongoDB Agent needs to be installed on each of the servers hosting the MongoDB Processes, which will be *<HOSTNAME-1>, <HOSTNAME-2>, and<HOSTNAME-3>*.
2. Go to *Deployment -> Agents -> Downloads & Settings*.
3. Under *MongoDB Agent*, click on the *Select your operating system* drop-down and select the OS for your server (In this case, we selected *RHEL/CentOS (7.X/8.X), SUSE12, SUSE15, Amazon Linux2 - RPM*).
4. Complete the instructions shown.
    a. First, download the agent on the server using the *curl* command.
    b. Install the rpm package.
    c. Generate an API key if not already generated. Treat this as a password and store it safely. *It is only shown once*.
    d. Open the Agent's config file `/etc/mongodb-mms/automation-agent.config`. and enter the values for *mmsGroupId*, *mmsApiKey*, and *mmsBaseUrl*.
    e. Skip the step to create a */data* directory as we would be hosting the data files under a different directory.
    f. Start the agent and enable the service so that it automatically starts on reboot of the server.

```
sudo systemctl start mongodb-mms-automation-agent.service
sudo systemctl enable mongodb-mms-automation-agent.service
```

2. Check the status of the agent using the below command. Also verify that you can see the Agent pinging under *Deployment -> Agents* tab *-> All Agents* tab.

```
sudo systemctl status mongodb-mms-automation-agent.service
```

3. The server should now show under *Deployment -> Servers*. Enable Monitoring for the server by clicking on the Ellipsis (. . .) and click on *Activate Monitoring*.

    a. This will allow you to monitor the health of the cluster via Ops Manager and is recommended for Managed Deployments.

4. Click on *Review & Deploy*, and confirm the changes.

5. We need to create the *mms-automation* user in the cluster and specify it when importing the cluster. Follow the below steps for the same:

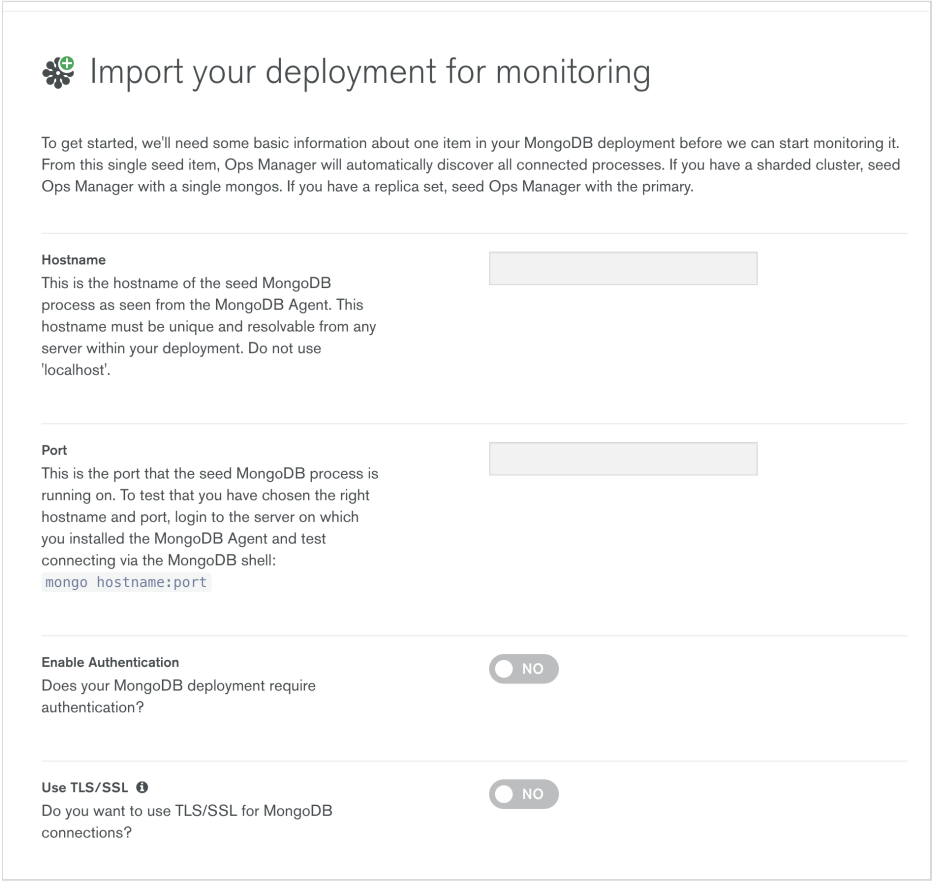
    i. Go to Deployment -> Security -> Authentication & TLS.
    ii. Click on the EDIT SETTINGS button.
    iii. Enable the Authentication Mechanisms as per the requirements.
    iv. In the team's case, SCRAM-SHA-1 and SCRAM-SHA-256 were selected.
    v. Click Next and enable TLS if required.
    vi. Continue through the Modal and configure any additional settings as required.
    vii. Click on Save.
    viii. Go to Deployment -> Security -> MongoDB Users.
    ix. Click on + ADD NEW USER.
    x. Add a root user on the admin database for administration purposes. Following details will be required.
    xi. Identifier: admin
    xii. Username: This can be as per your choice and will identify the root user. The team used mongoadmin.
    xiii. Roles: Select root@admin.
    xiv. Password: Enter the password for this user.
    xv. Click on Review & Deploy, and confirm the changes.
    xvi. Once Security is enabled, get the password for the *mms-automation* user by going to *Security -> Authentication & TLS -> Edit Settings*, and continue through to show the *mms-automation* user and its hidden password. Click on the *Show Password* icon to show the password.

a. Create the *mms-automation* user on your cluster using the following command.

```
db.getSiblingDB("admin").createUser(
   {
     user: "mms-automation",
     pwd: <password>,
     roles: [
       'clusterAdmin',
       'dbAdminAnyDatabase',
       'readWriteAnyDatabase',
```

```
        'userAdminAnyDatabase',
        'restore',
        'backup'
      ]
    }
)
```

6. Go to *Deployment -> Add New -> Existing MongoDB Deployment*. Add the hostname of the Primary node, its port, and the authentication credentials.



7. Continue through the steps to discover the MongoDB Process, and import it for automation. Finally import the users and roles and then *Review & Deploy*.

5 Other Notes and Issues

5.1 Authentication failures with the Automation agent on 2 nodes

During the consultation, when enabling automation on the production replica set we faced “Authentication failure” error on 2 of the nodes. Later it was identified that the **mmsGroupId** copied was different in these 2 servers.

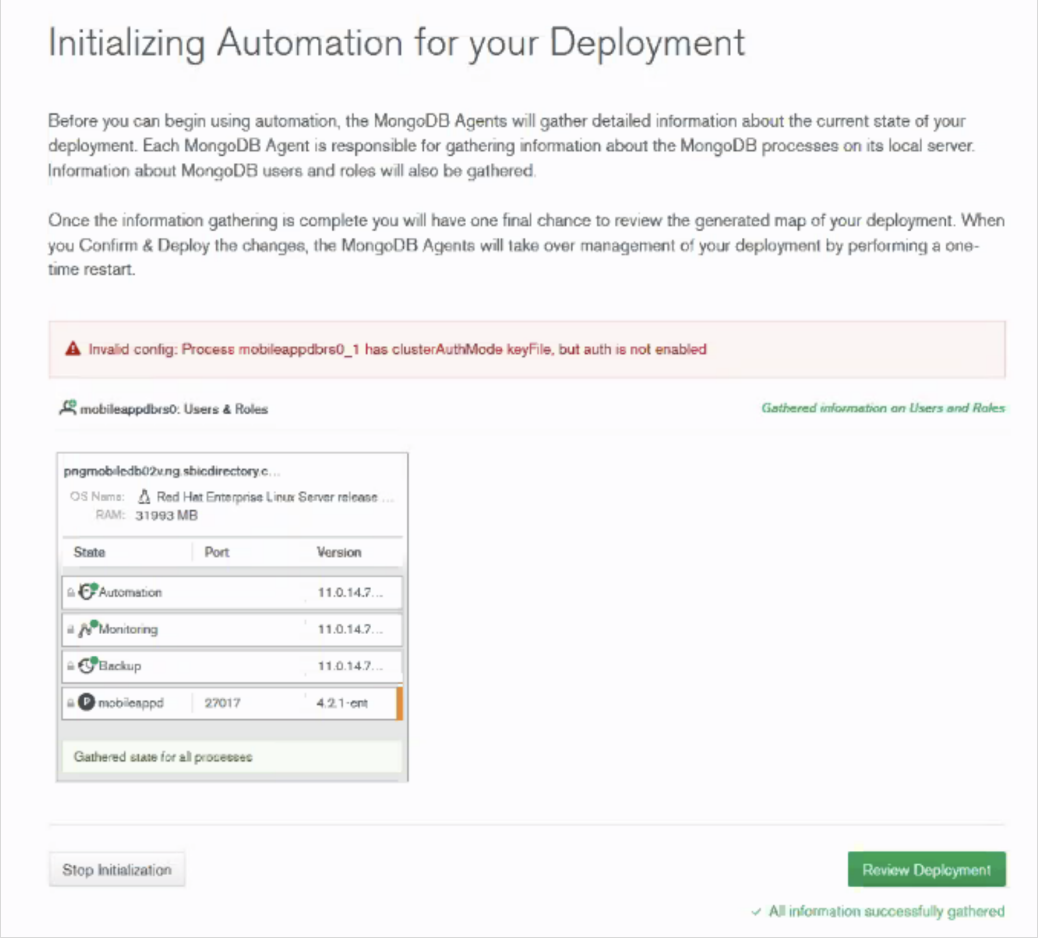
5.2 MongoDB Database version mismatch error

When enabling automation on the production replica set, we got the error of “Version mismatch” on the nodes of the replica set. After troubleshooting the issue it was identified that one of the nodes of the replica set was running on the Community Version of MongoDB while the rest were running on the Enterprise version.

We removed the Community and installed the Enterprise version of MongoDB on this node during the consultation to resolve the issue.

5.3 MongoDB node has clusterAuthNode keyFile, but auth is not enabled error

During the consult , when initializing the Automation for the deployment we faced the error “Process <prodcess_name> has clusterAuthNode keyFile, but auth is not enabled”.

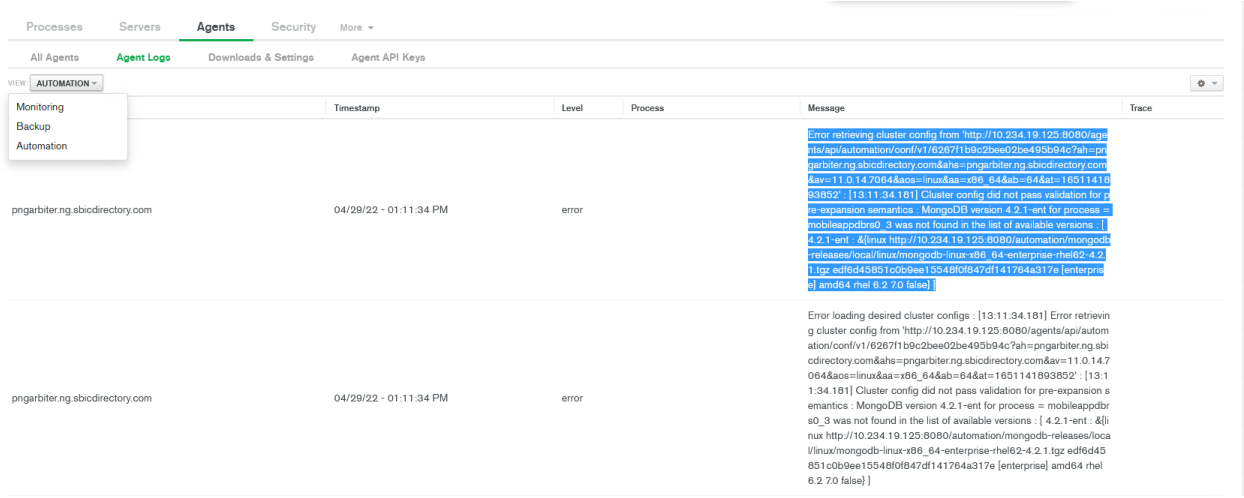


After troubleshooting the issue, it was identified that for the production cluster mongodb configuration, the team has mentioned the keyfile but has not enabled Authentication/Authorization.

Configuration was updated with “**authorization: enabled**” config to resolve the issue.

5.4 Error while initiating the Automation on the cluster: “Error retrieving cluster config, MongoDB version 4.2.1 not found in the list of available versions”

The team has been observing the below error after the Automation initialization:



After troubleshooting the issue it was recommended to perform the below steps to resolve the issue:

1. Download the binary : https://downloads.mongodb.com/linux/mongodb-linux-x86_64-enterprise-rhel70-4.2.1.tgz on the Ops-manager node at location : `/opt/mongodb/mms/mongodb-releases/`
2. Update the Version Manifest from Ops Manager Admin screen.
3. Restart the service of Ops Manager and the mongodb node.

6 Recommended Further Consulting and Training

6.1 Training

MongoDB offers a comprehensive set of instructor-led training courses covering all aspects of building and running applications with MongoDB. Instructor-led training is the fastest and best way to learn MongoDB in depth. Both public and private training classes are available - for more information or to enroll in classes, see www.mongodb.com/products/training/instructor-led