Getting Started with MongoDB Ops Manager

MongoDB Ops Manager Workshop

Release 3.4

MongoDB, Inc.

May 30, 2017

Contents

1 MongoDB Ops Manager

1.1 Lab: Ops Manager Installation

Premise

Ops Manager is an On-Prem operational solution for the management of MongoDB clusters.

Enables features like:

- Automation
- Backup and Recovery
- Monitoring

Over the course of this lab we will be installing Ops Manager with high availability and scalability in mind.

Note: As a reminder, for details on how to setup clusters for this class, see:

- https://docs.google.com/document/d/1vhA6NvlTsPe1rw_fb7N5NrYzJ78odiHWBd5yf9vPd64

Ops Manager HA

Ops Manager requires a number of servers for high availability (HA).

- Monitoring and backup/recovery are essential for production operations.
- Therefore, it's important to assure high availability for Ops Manager.
- For this we need to follow a specific deployment topology.

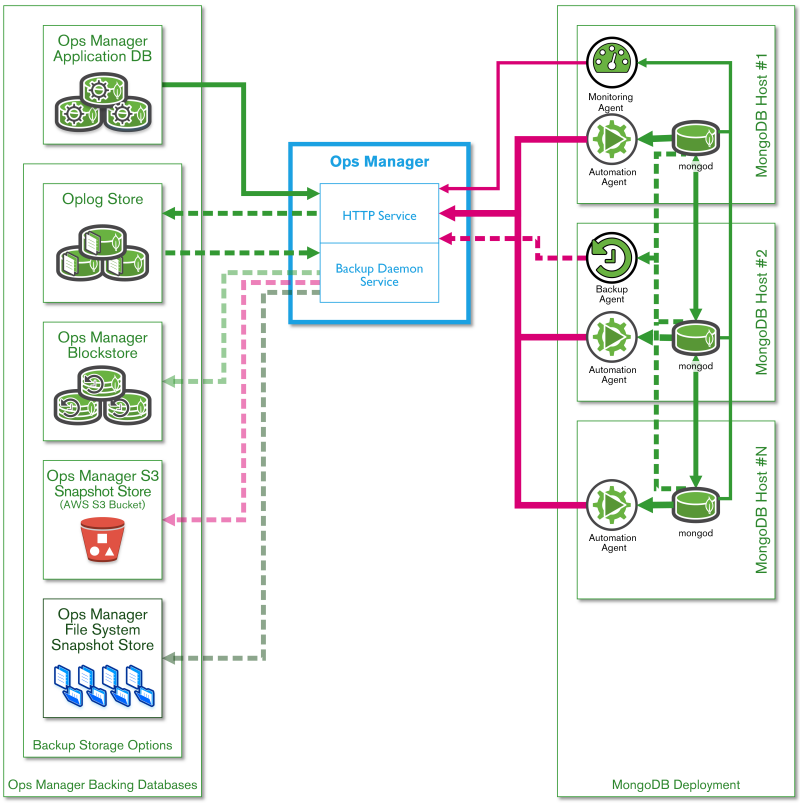
Ops Manager Scalability

Why do we need our operations tool to be scalable?

- The main reason is backup and recovery requirements
- The amount of data individual applications generate will grow
- The number of applications your Ops Manager deployment supports will grow
- Plan to accommodate both forms of growth

Ops Manager Architecture Review

Let's review the Ops Manager architecture[1] :



Note: Alternative deployments of Ops Manager are available at:

- https://docs.opsmanager.mongodb.com/current/core/deployments/

For this diagram raise students attention to the following:

- Ops Manager is composed out of two main services
  - HTTP Service: Ops Manager Application
  - Backup daemon
- Ops Manager will collect data from the different agents
  - backup
  - monitoring
  - automation
- Ops Manager has three main data stores
  - Application DB (MongoDB Cluster)
    - user info
    - monitoring data

[1] https://docs.opsmanager.mongodb.com/current/core/system-overview/

        * group data
        * backup job queries
        * backup metadata
    - Backup DB (MongoDB Cluster)
        * OpLog store
        * Sync store
        * Blockstore: data snapshots, divided in small chunks being compressed and de-deduplicated
        * or Filestore: data snapshots as files
    - Head Database
        * Keeps track of oplog and initial syncs
        * There's one head database for each backed-up replica set
        * Similar load as a Secondary

Exercise: Architect the Ops Manager Deployment

It's time to set up the our Ops Manager Deployment. As a team, make a plan for the following:

- Two replica sets of 3 nodes
    - Application Database replica set as **APPDB**
    - Backup Database replica set as **BACKUPDB**
- A redundant service of the Ops Manager Application
    - The hosts that will be supporting the OM App: `opsmgr1`, `opsmgr2` and `opsmgr3`
    - Load Balancer in front of those 3 instances
    - The load balancer is already set up. The name is in the info file

Solution: Architect the Ops Manager Deployment

- 2 possible solutions:
    - each `opsmgr` node gets App, BackupDaemon, AppDB and BackupDB
    - each `opsmgr` node gets App, BackupDaemon and BackupDB, each `nodeX` node gets AppDB
- Do we need a replica set to support the Backup Daemon?
    - Yes, you always need one, and we will be using the *Blockstore* part of the Backup DB
        * we can use a *Filestore* as an alternative but for the exercise let's use the *Blockstore*

Exercise: Configure Ops Manager Application Database

Ops Manager needs to store data:

- Configuration of nodes, groups, users
- Metrics for monitoring
- Backup metadata and job queries

Also consider relevant security settings[2] for this database.

From the available machines go ahead and set up a replica set to support the *Application Database*.

Name this replica set **APPDB**

You can install MongoDB by running:

```
yum install -y /share/downloads/mongodb_packages/mongodb-enterprise-3.4.2-1.el7.x86_
↪64.rpm
```

Solution: Configure Ops Manager Application Database

```
mongod -f /share/etc/appdb.conf
```

- The */share/etc/appdb.conf* file should have the following settings:

```
cat /share/etc/appdb.conf
replication:
  replSetName: "APPDB"
  oplogSizeMB: 100
storage:
  wiredTiger:
    engineConfig:
      cacheSizeGB: 2
      journalCompressor: zlib
net:
  port: 27001
```

Note: Pay attention to:

- cache size, because you can't run 2 DB with 50% of the memory each

---

[2] https://docs.mongodb.com/manual/administration/security-checklist/

Solution: Configure Ops Manager Application Database (con't)

- Connect to each `opsmgr` instance and launch a mongod
- Once all instances `mongod` instances are up initiate the replica set

```
mongo --host opsmgr1:27001
> rs.initiate({
  _id: "APPDB",
  members: [
    {_id: 1, host:"opsmgr1:27001"},
    {_id: 2, host:"opsmgr2:27001"},
    {_id: 3, host:"opsmgr3:27001"}
  ]
})
```

Note:

- To validate their deployment you should ask students to run `rs.status()` on one of their APPDB members.

Exercise: Configure Ops Manager Backup Database

Ops Manager needs to store backup blocks/snaphots, either

- in database
- file system

From the available machines go ahead and set up a replica set to support the *Backup Database*.

Name this replica set **BACKUPDB**

Solution: Configure Ops Manager Backup Database

```
mongod -f /share/etc/backupdb.conf
```

- The */share/etc/backupdb.conf* file should have the following settings:

```
cat /share/etc/backupdb.conf
replication:
  replSetName: "BACKUPDB"
  oplogSizeMB: 100
storage:
  wiredTiger:
    engineConfig:
      cacheSizeGB: 2
      journalCompressor: zlib
net:
  port: 27002
```

Note: Pay attention to:

- cache size, because you can't run 2 DB with 50% of the memory each

---

Solution: Configure Ops Manager Backup Database (con't)

- Connect to each `opsmgr` instance and launch a mongod
- Once all instances `mongod` instances are up initiate the replica set

```
mongo --host opsmgr1:27002
> rs.initiate({
  _id: "BACKUPDB",
  members: [
    {_id: 1, host:"opsmgr1:27002"},
    {_id: 2, host:"opsmgr2:27002"},
    {_id: 3, host:"opsmgr3:27002"}
  ]
})
```

Exercise: Install, Configure and Launch the Ops Manager Service

Habemus Replica Sets! Now it's time to launch the **Ops Manager** service. For this you will need to:

- Install Ops Manager
  - The files can be found in /share/downloads/opsmgr_packages
  - Follow the instructions to install from rpm[3]
- Edit Ops Manager configuration `conf-mms.properties`:
  - Point the config to the replica set: **APPDB**
- Launch the Ops Manager service
- Hint: there is a common keyfile shared by all 3 instances
- You can install Ops Manager by running:

```
yum install -y /share/downloads/opsmgr_packages/mongodb-mms-3.4.3.402-1.x86_64.rpm
```

Note: The next page, first part of the solution, is given in the student notes, as this one may prove complicated.

We are striving for an Highly Available Ops Manager deployment therefore:

- Before mentioning *again* the HA requirement you may ask students why do we need at least 2 `opsmgr` nodes?
- Make sure students use machines that are tagged with Ops Manager

---

[3] https://docs.opsmanager.mongodb.com/current/tutorial/install-on-prem-with-rpm-packages/#install-the-onprem-package-on-each-server-being-used-for-onprem

Solution: Install, Configure and Launch the Ops Manager Service

Details on how to configure HA configure HA app[4]

Generate a keyfile `gen.key` for the 3 hosts:

```
# Make sure you use replace opsmgr for the host ip
ssh -A centos@opsmgr1

# Install ops manager server
yum install -y /share/downloads/opsmgr_packages/mongodb-mms-3.4.3.402-1.x86_64.rpm

# Edit the configuration options
vi /opt/mongodb/mms/conf/conf-mms.properties

# Generate a gen.key file
openssl rand 24 > /share/gen.key
cp /share/gen.key /etc/mongodb-mms/

# Copy this generated file to all opsmgr hosts
# You might need to first scp to a folder you have permissions on
scp /share/gen.key centos@opsmgr2:/etc/mongodb-mms/
scp /share/gen.key centos@opsmgr3:/etc/mongodb-mms/
```

Solution: Install, Configure and Launch the Ops Manager Service (con't)

- Edit Ops Manager configuration file `/opt/mms/conf-mms.properties`

```
# Replace mongo.mongoUri with APPDB replica set connection string
# e.g., mongodb://opsmgr1:27001,opsmgr2:27001,opsmgr3:27001/?replicaSet=APPDB
sed -i.bak "s/^\(mongo.mongoUri=\).*/\1mongodb:\/\/opsmgr1\:27001,opsmgr2\:27001, \
opsmgr3\:27001\/\?replicaSet=APPDB/" /opt/mms/conf-mms.properties

# Repeat the operation on all opsmgr hosts
```

- Launch Ops Manager on all `opsmgr` hosts

```
ssh -A centos@opsmgr1 "service mongodb-mms start"
ssh -A centos@opsmgr2 "service mongodb-mms start"
ssh -A centos@opsmgr3 "service mongodb-mms start"
```

Note: To validate this exercise the students should be able to:

- Connect to the Ops Manager Installation via their load balancer URL
- Register admin user credentials and Ops Manager group

You can go ahead and walk them through the series of Ops Manager setup screens.

---

[4] https://docs.opsmanager.mongodb.com/current/tutorial/configure-application-high-availability/

Exercise: Install Ops Manager Automation Agents

At this point **Ops Manager** should be up and running. Now it's time to install our Automation Agents[5]:

- In the remaining VMs (node1, node2, etc) install the automation agent
- Make sure that all nodes are discoverable on the server's dashboard
- Validate that all agents are reporting pings correctly

Note: You should ask the students to install the agents on the remaining VMs

- Validate that all nodes are discoverable
- All agents are reporting data without errors

You'll also want to make sure that students successfully changed their instance hostnames to the `nodeX` form. This enables them to use the certificates that are on the machines for later labs.

1.2 Lab: Enable the Ops Manager Public API

Learning Objectives

Upon completing this lab, students will be able to:

- Understand the requirements for enabling Ops Manager Public API

Exercise: Enable Public API Access

Ops Manager, for most users, is primarily controlled via it's web UI, but it has an API that supports most of the operations that users perform.

Enable your deployment of Ops Manager to allow API calls.

- Generate an API Key called "generic"

To verify that you've done this properly you can make the following request:

```
curl -u "$EMAIL:$APIKEY" --digest \
  -i "$OPSMGRURL/api/public/v1.0/groups"
```

Note: Enabling the Public API on Ops Manager enables it across all groups. Previously, the API had to be enabled on a group by group basis.

---

[5] https://docs.opsmanager.mongodb.com/current/tutorial/nav/install-automation-agent/

1.3 Lab: Ops Manager User Administration

Learning Objectives

Upon completing this lab, students will be able to:

- Administer Ops Manager groups
- Identify the differences between Ops Manager user roles
- Create and define Ops Manager users

Exercise: Create Group

Connect to your Ops Manager instance and create the following `group`:

- **CIRCUS_MAXIMUS**

Note: This is a very simple exercise.

Take the time to explore the `admin` menu with students.

Exercise: Create Users

Using the Ops Manager API[6] , create the following users:

- **aediles@localhost.com** :
  - password: "123ABCabc!"
  - role: Owner[7]
- **patrician@localhost.com** :
  - password: "DAxN3ZpM6U!"
  - role: Monitoring Admin[8]
- **consus@localhost.com** :
  - password: "&o7chac0v3r3d"
  - role: Backup Admin[9]

Note: To accomplish this task, the users will have to do the following steps:

- enable the api[10],
- create different users using the HTTP Rest API

e.g.: **Owner**

[6] https://docs.opsmanager.mongodb.com/current/api/
[7] https://docs.opsmanager.mongodb.com/current/reference/user-roles/#owner
[8] https://docs.opsmanager.mongodb.com/current/reference/user-roles/#monitoring-admin
[9] https://docs.opsmanager.mongodb.com/current/reference/user-roles/#backup-admin
[10] https://docs.opsmanager.mongodb.com/current/tutorial/enable-public-api/

```
curl -u "$EMAIL:$APIKEY" -H "Content-Type: application/json" \
  --digest -i -X POST "$OPSMGRURL/api/public/v1.0/users" --data '
  {
  "username": "aediles@localhost.com",
  "emailAddress": "aediles@localhost.com",
  "firstName": "This",
  "lastName": "Mine",
  "password": "123ABCabc!",
  "roles": [{
    "groupId": "$GROUPID",
    "roleName": "GROUP_OWNER"
    }]
  }'
```

Exercise: Create Global Users

In various different situations, we will need users with global roles.

Please create, either through the API or web UI, the following users:

- **automater@localhost.com** :
  - password: "84hjdpx%ea3m"
  - role: Global Automation Admin[11]
- **masterchef@localhost.com** :
  - password: "c6ny3n4x*8"
  - role: Global User Admin[12]

After creating these users, connect with the most appropriate user to change the password of the **CIRCUS_MAXIMUS** *Owner* user.

The new password should be *"$uperC00l"*

This last operation should be accomplished using the HTTP Rest API interface.

Note: Make sure that students understand which user to log in with and how to change the user password of *aediles@localhost.com*.

For the operation students will have to do the following steps:

- create a Public API key for user *masterchef@localhost.com*
- change user *aediles@localhost.com* password through the API

```
// get the user
curl -u "masterchef@localhost.com:$APIKEY" \
  --digest -i "$OPSMGRURL/api/public/v1.0/users/byName/aediles@localhost.com"

// change the user's password
curl -u "masterchef@localhost.com:$APIKEY" -H "Content-Type: application/json" \
  --digest -i -X PATCH "$OPSMGRURL/api/public/v1.0/users/$USERID" --data '
  {
```

[11] https://docs.opsmanager.mongodb.com/current/reference/user-roles/#global-automation-admin
[12] https://docs.opsmanager.mongodb.com/current/reference/user-roles/#global-user-admin

```
 "password": "$uperC00lpa22"
 }'
```

1.4 Lab: Secure Replica Set

Premise

- Setting up a MongoDB Replica set is quite easy and fast.
- Setting up a Secured MongoDB replica set requires a few extra steps.
- In this lab we will be exploring how to setup a secured Replica Set through Ops Manager.

Note: Security is an important topic of production deployments and we want the students to be fully acquainted with the different options of MongoDB Security.

X.509 Authentication Mechanism

We will be using X.509 certificates[13] for authentication and TLS/SSL network encryption.

Note: The actual X.509 details are out-of-scope for this training. Our purpose is **not**:

- to educate people on the authentication mechanism itself
- detailed explanation on how TLS/SSL works

Our purpose is to:

- Review the different authentication mechanisms
- How students can use such mechanism if they choose too
- The tradeoffs of X.509 when compared with other auth mechanisms

---

Ops Manager Group SSL and Auth

To build secured MongoDB deployments you first need to enable Auth and SSL[14] on your group.

All VMs have a set of certificates that you will be using to configure your secured deployment.

In folder `/share/downloads/certs` (linked to `/etc/ssl/mongodb`) you will find:

- `ca.pem`: SSL CA certificate
- `automation.pem`: Automation agent certificate
- `backup.pem`: Backup agent certificate
- `monitor.pem`: Monitoring agent certificate
- `nodeX.pem`: Replica set member certificates (X)

[13] https://docs.mongodb.com/manual/core/security-x.509/
[14] https://docs.opsmanager.mongodb.com/current/tutorial/enable-ssl-for-a-deployment/

- `dbadmin.pem`: MongoDB DB Admin certificate

Note: Ask students to list the files under `/share/downloads/certs` on instances to validate that their installation process is correct. All our answers use `/etc/ssl/mongodb` which is linked to the above.

Make sure to highlight that:

- The enabling of auth and ssl on a group level is to ensure correct communicate between all instances
- Ensuring the same Certificate Authority (CA) certificate
- Enabling agents to perform their normal operations
- Create the required agent users

Exercise: `VERYSAFE` Group

Let's start by creating a group called `VERYSAFE` that has SSL enabled.

- Using the existing certificates, configure the agents accordingly.
- You need to specify certificates for
  - Certificate Authority
  - Monitoring Agent
  - Backup Agent
  - Automation Agent
- **The existing certificates do not have a decryption password!**

Note: This might be a bit hard for students that are not experienced with Ops / Cloud Manager

- Take the time to navigate users through the UI to configure these settings
- Make sure that students do not provide a decypher password for the certificates

Once the group is created you will need to reconfigure your existing agents to use the new group

- Either you install new agents or just stop the service and reconfigure the agents configuration file

```
sudo service mongodb-mms-automation-agent stop

# edit /etc/mongodb-mms/automation-agent.config and add new APIKey and GroupId

sudo service mongodb-mms-automation-agent start
```

- Students will find the corresponding instructions on *Settings* -> *Agents* -> *Host Version Agent*

Exercise: Secure Replica Set Deployment

Once the automation agent has been reconfigured and servers are detected on your deployment, it's then time to deploy our secure replica set.

Create a replica set named **SECURE** with the following configuration:

- 3 Nodes:
  - **node1**, **node2** and **node3**
  - Port *27000*
- **sslMode**: requiredSSL
- **sslPEMKeyFile**: */etc/ssl/mongodb/nodeX.pem*

Note: Students should create a replica set from Ops Manager UI that will reflect the wanted configuration:

- Once the `VERYSAFE` group has been create we need to **Add -> New Replica Set**
  - Name: SECURE
  - Port Range: **27000**
  - DB Path Prefix: `/data`
  - sslMode: `requiredSSL`
  - Apply
- Before the replica set is created, we need to **Modify** the individual node members with the corresponding server **sslPEMKeyFile**
  - node1: */etc/ssl/mongodb/node1.pem*
- `nodeX.pem`: there should be a certificate file per each node.
- This setting needs to be configured on a per instance level.

Exercise: X509 Users

Time to create users that will authenticate using an X.509 certificate.

- Go ahead and create a dbAdminAnyDatabase[15] user that authenticates using the `dbadmin.pem` certificate.
- To create users that authenticate using X509 certificates you should check the Certificate Subject as user[16] documentation.
- After the user has been created, connect to the *Primary* node of the replica set and create database "`allgood`".

Note:

- Students might not be familiar with this mechanism so point them to Certificate Subject as user[17].
- Students will have to extract the certificate `subject` info

[15] https://docs.mongodb.com/manual/reference/built-in-roles/#dbAdminAnyDatabase
[16] https://docs.mongodb.com/manual/tutorial/configure-x509-client-authentication/#add-x-509-certificate-subject-as-a-user
[17] https://docs.mongodb.com/manual/tutorial/configure-x509-client-authentication/#add-x-509-certificate-subject-as-a-user

```
$ openssl x509 -in /etc/ssl/mongodb/dbadmin.pem -inform PEM -subject -nameopt␣
↪RFC2253
subject= C=US,ST=New York,L=New York City,O=MongoDB,OU=USERS,CN=dbadmin
-----BEGIN CERTIFICATE-----
MIIDeDCCAmCgAwIBAgICmDQwDQYJKoZIhvcNAQEFBQAwcjESMBAGA1UEAxMJbG9j
...
```

and use that subject info to create the required user.

- We can create this user by connecting to the replica set and perform the following operations:
- Authenticate to primary node using automation agent certificate (students can do this from a opsmgrX machine, where `mongo` is installed)

```
mongo --host PRIMARY_NODE --ssl --sslPEMKeyFile /etc/ssl/mongodb/automation.pem --
↪sslCAFile /etc/ssl/mongodb/ca.pem
```

- Create the new `dbAdminAnyDatabase` user with the `dbadmin.pem` certificate

```
db.getSiblingDB("$external").runCommand({
  createUser: "C=US,ST=New York,L=New York City,O=MongoDB,OU=USERS,CN=dbadmin",
  roles: [
    { role: 'dbAdminAnyDatabase', db: 'admin' }
  ],
  writeConcern: { w: "majority" , wtimeout: 5000 }
})
```

- Connect to primary using `dbadmin.pem`:

```
mongo --host PRIMARY_NODE  --ssl --sslPEMKeyFile /etc/ssl/mongodb/dbadmin.pem
  --sslCAFile /etc/ssl/mongodb/ca.pem
```

- Authenticate and create new database `allgood`

```
db.getSiblingDB("$external").auth({
  mechanism: "MONGODB-X509",
  user:"C=US,ST=New York,L=New York City,O=MongoDB,OU=USERS,CN=dbadmin"
})

db.createDatabase("allgood")
```

Find out more
mongodb.com | mongodb.org
university.mongodb.com

Having trouble?
File a JIRA ticket:
jira.mongodb.org

Follow us on twitter
@MongoDBInc
@MongoDB