MongoDB Ops Manager Workshop

Release 2.0

MongoDB, Inc.

June 20, 2016

Contents

1 MongoDB Ops Manager

1.1 Lab: Ops Manager Installation

Permise

Ops Manager is a solution for on-prem MongoDB cluster operational management.

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: For this lab we will group students into teams.

- There will be a maximum of 5 teams!
- Each team will have at their disposal a set of virtual machines
- Verify that all students have successfully pinged their designated VMs
- These VMs will be properly tagged (dns or ip) with their intended purpose

---

Ops Manager HA

Ops Manager requires a number of servers for high availability.

- 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.

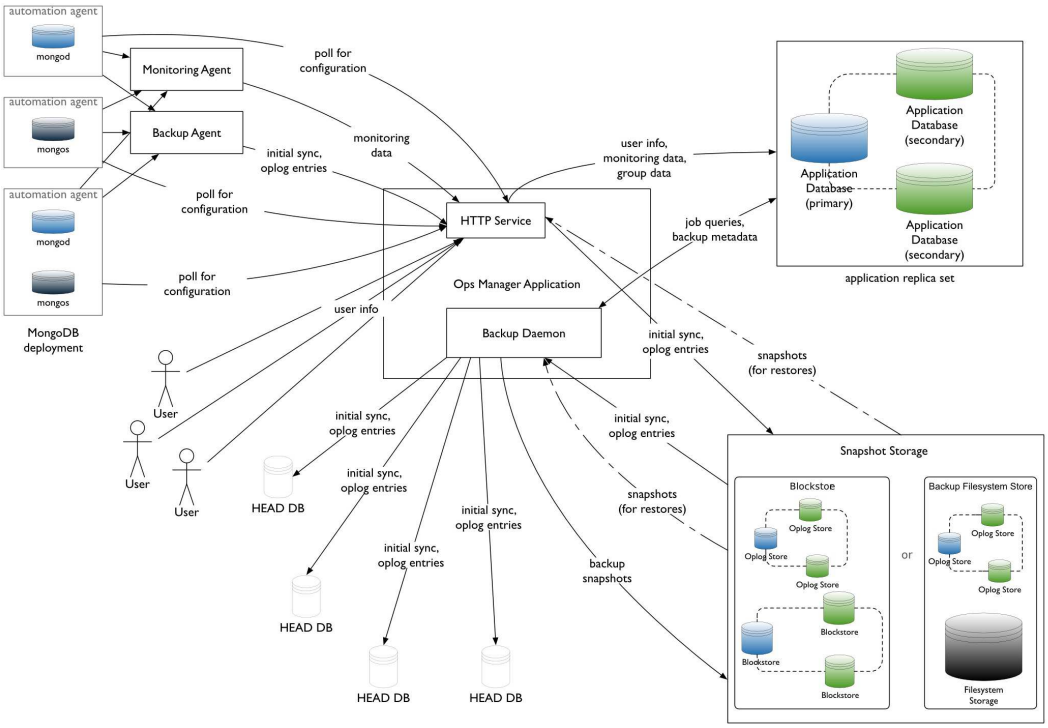
Ops Manager Scalability

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 Ops Manager architecture[1]:



Note: 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

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

- Has three main datastores
  - Application Replica Set (MongoDB Cluster)
    - user info
    - monitoring data
    - group data
    - backup job queries
    - backup metadata
  - Snapshot Storage (MongoDB Cluster)
    - Blockstore: MongoDB Instance
    - or Filestore: File snapshots
  - Head Database
    - Keeps track of oplog and initial syncs
    - There's one file per database being backed up

Launch Ops Manager MongoDB Instances

It's time to set up the virtual machines that will support Ops Manager Deployment.

We will need the following instances:

- Two Replica Set Clusters
  - Application Replica Set
  - BlockStore Replica Set
- Two instances to run redundant services of Ops Manager Application
- Load Balancer

Note: Ask students to make a deployment and configuration plan. Use the following questions as guide:

- Where do we start ?
  - Boot up Ops Manager Application Replica Set
- Do we need a Replica Set for the Backup Daemon?
  - Yes, we will be using Blockstore
    - we can use Filestore as alternative but for the exercise let's privilege Blockstore

All instances should already be pre-provisioned or running on local virtualbox

---

1. Configure Ops Manager Application Database

Ops Manager needs to store data:

- Configuration of nodes, groups, users
- Metrics for monitoring
- Backup metdata 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*.

Note:

- Students should be assembling a 3 node Replica Set to allow the HA.
- Extra "points" for students that setup their replica set under SSL
- To validate their deployment you should ask students to run validation script

---

2. Configure and Launch OpsManager Service

Habemus Replica Set!

Now it's time to launch **Ops Manager** service. For this you will need to:

- Configure Ops Manager into two separate Virtual Machines
- Point these to the previously configured Replica Set
- Allow offline binary access[3]
- Configure Load Balancer
- Launch Ops Manager services

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

- Before mentioning *again* the HA requirement you may ask students why we need 2 VM's?
- Make sure students use machines that are tagged with OpsManager

You may want to point to the following documentation page:

- configure HA app[4]

For the Load Balancer we will have two options:

- AWS: students should use Elastic Load Balancer
- Vagrant: students should use HAproxy instance

To validate this exercise the students should be able to:

- Login through Load Balancer ip address
- Register admin user credentials and Ops Manager group

---

[2] https://docs.mongodb.com/manual/administration/security-checklist/
[3] https://docs.opsmanager.mongodb.com/current/tutorial/configure-local-mode/
[4] https://docs.opsmanager.mongodb.com/current/tutorial/configure-application-high-availability/

3. Install OpsManager 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 VM, install the automation agent
- Make sure that all nodes are discoverable on the servers 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

1.2 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: "123ABCabc!"
  - role: Monitoring Admin[8]
- **consus@localhost.com** :

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

- password: “&o7chac0v3r3d”
- role: Backup Admin[9]

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

- enable the api[10]
- enable the public api on the group
- create different users using the HTTP Rest API

e.g.: **Owner**

```
curl -u "admin@localhost.com:$APIKEY" -H "Content-Type: application/json"
--digest -i -X POST "http://opsmgr.training/api/public/v1.0/users" --data '
{
"username": "aediles@localhost.com",
"emailAddress": "aediles@localhost.com",
"firstName": "This",
"lastName": "Mine",
"password": "aLLm1n3$3cr3t",
"roles": [{
  "groupId": "$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 console, the following users:

- First user with Global Automation Admin[11] role : *automater@localhost.com*
- Second user granted Global User Admin[12] user : *masterchef@localhost.com*

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 *thisismine@localhost.com* user.

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

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

```
// get the user
curl -u "masterchef@localhost.com:$APIKEY"
--digest -i "http://opsmgr.training/api/public/v1.0/users/byName/aediles@localhost.com"
curl -u "masterchef@localhost.com:$APIKEY" -H "Content-Type: application/json"
--digest -i -X PATCH "http://opsmgr.training/api/public/v1.0/users/$USERID" --data '
{
```

[9] https://docs.opsmanager.mongodb.com/current/reference/user-roles/#backup-admin
[10] https://docs.opsmanager.mongodb.com/current/tutorial/enable-public-api/
[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"
}'
```

1.3 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
- Configure Ops Manager groups to allow Public API requests

Exercise: Selective Node Rest Client

Ops Manager enables administrators to determine from where Public API calls are allowed. From which client nodes it accepts such interface requests.

Enable your deployment of Ops Manager to allow only one specific client to perform API calls.

- Generate an API Key called “generic”
- Add CIDR block for ip whitelisting enabling

Note: Enabling the Public API Ops Manager will require an ip whitelist CIDR block.

- The most permissive CIDR Block is 0.0.0.0/0
- for this exercise students should choose
  - a particular machine
  - or set of machines to accept API requests only from such nodes.

ex:

```
// accept only from 120.10.10.132
120.10.10.132/32

// from 123.10.10.132 to 123.10.10.135
123.10.10.132/30
```

Exercise: Enable Group Public API

Groups are more than just sets of machines and MongoDB instances. Groups also enclose security considerations.

Administrators have the ability to enable the Public API interface on a per group basis.

For this exercise go ahead and:

- Create a group called "LOVE_API_CALLS"
- Enable Public API for this group

Note: Students should

- create the group
- enable the API calls

To validate this you should ask students to submit the following request

```
// set APIKEY with `generic` key value
curl -u "admin@localhost.com:$APIKEY" --digest
-i "http://opsmgr.training/api/public/v1.0/groups"
```

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 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

---

[13] https://docs.mongodb.com/manual/core/security-x.509/

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 */shared/etc/ssl* 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)
- `dbadmin.pem`: MongoDB DB Admin certificate

Note: Ask students to list files under */shared/etc/ssl* on instances to validate that their installation process is correct. 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

`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 any 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
```

[14] https://docs.opsmanager.mongodb.com/current/tutorial/enable-ssl-for-a-deployment/

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

Secure Replica Set Deploy

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: port range {27000 -> 27002}
- **clusterAuthMechanism**: x509
- **sslMode**: requiredSSL
- **sslPEMKeyFile**: */shared/etc/ssl/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
  - Eligible Server Regex: `.*`
  - Port Range: **27000 27002**
  - DB Path Prefix: `/data`
  - clusterAuthMechanism: **x509**
  - sslMode: `requiredSSL`
  - Apply
- Once we have applied we now need to **Modify** the individual node members with the corresponding server **sslPEMKeyFile**
  - node1: */shared/etc/ssl/node1.pem*
- `nodeX.pem`: there should be a certificate file per each node.
- This setting needs to be configured on a per instance level.

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 `dbadmin.pem` certificate.
- To create users that authenticate using X509 certificates you should check Certificate Subject as user[16] docs
- 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 /share/etc/ssl/dbadmin.pem -inform PEM -subject -nameopt RFC2253
subject= C=US,ST=New York,L=New York City,O=MongoDB,OU=TRAINING,CN=client
-----BEGIN CERTIFICATE-----
MIIDezCCAmOgAwIBAgIDBZc1MA0GCSqGSIb3DQEBBQUAMGUxCzAJBgNVBAMTAkNB
```

and use that subject info to create the required user.

- We can create this user using the Ops Manager UI or we can connect to the replica set and perform the following operations:
- Authenticate to primary node using automation agent certificate

```
mongo --host PRIMARY_NODE --ssl --sslPEMKeyFile /share/etc/ssl/automation.pem --sslCAFile /share
> db.getSiblingDB("$external").auth({
  mechanism: "MONGODB-X509",
  user:"C=US,ST=New York,L=New York City,O=MongoDB,OU=EDU,CN=automation"
  })
```

- 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=TRAINING,CN=dbadmin",
  roles: [
    { role: 'dbAdminAnyDatabase', db: 'admin' }
  ],
  writeConcern: { w: "majority" , wtimeout: 5000 }
})
```

- Connect to primary using `dbadmin.pem`:

```
mongo --host PRIMARY_NODE  --ssl --sslPEMKeyFile /share/etc/ssl/dbadmin.pem
--sslCAFile /share/etc/ssl/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=TRAINING,CN=dbadmin"
})

db.createDatabase("allgood")
```

1.5 Lab: Reconfig Replica Set

Learning Objectives

Upon completing this lab, students should be able to:

- Reconfigure a replica set
- Outline the different stages of deployments

Dependencies

- In order to complete all purposed exercises we need to first enable the Public API.
- Go to your group settings and enable the Public API.
- Do not forget to set an appropriate CIDR block for the IP whitelist and Generate the API Key.

Note: Make sure students are aware of this dependency.

Exercise: Initial Replica Set

- Using the Ops Manager UI go ahead and create a 3 node replica set:
  - Replica set name `METAMORPHOSIS`
  - Two data bearing nodes
  - One arbiter
- *All instances should be installed using MongoDB 3.2.1*

Exercise: Add Replica Set Members

- Let's assume that we require highler level of High Availability (HA).
- Add 2 new data bearing nodes
  - First node should have priority 0
  - Second node should be an hidden replica.

Exercise: Decomission Replica Member

- One of your nodes is not making the cut.
- Change your replica set by "decommissioning" one of the instances
- Make sure that your replica set keeps up majority and previous level of node failure resilience

---

Note: Make sure students are aware that removing one of nodes might require adding an extra one.

- Not totally true in this case since there's an `arbiter` member
- They will keep the same availability by just removing a node
- We can also remove the `arbiter` to keep an odd number of nodes

Exercise: Upgrade MongoDB Version

- Our CTO, for compliance reasons, demands that all of our nodes should be on the latest version of MongoDB.
- Upgrade all nodes in your replica set without downtime.

Note:

- Please instruct students to change the version of MongoDB to the most recent version of MongoDB
- Make sure they know how to make the required version available in their Ops Manager deployment.
- Update the Version Manifest: Deployment -> Version Manager -> Update MongoDB Version Manifest

---

Exercise: Upgrade Replica Set Name

- We decided that we are not going to change our cluster going forward.
- Using Ops Manager API, update the replica set name to "PRODUCTION"

Note:

- Get the group Id and add it to an environment variable
- Settings -> Group Settings (group id)

```
export GROUPID=YOURGROUPID
```

- Create a new API key
- Settings -> Public API Key -> Generate

```
export apiKey=YOURAPIKEY
```

- Get your Cluster ID

```
// this will higlight your clusterid
curl -u "$username:$apiKey" -H "Content-Type: application/json" --digest -i
"http://opsmgr.training/api/public/v1.0/groups/$GROUPID/clusters/" | grep "id"

export CLUSTERID=YOURCLUSTERSID
```

- Update the replica set name

```
export username=yourusername
curl -u "$username:$apiKey" -H "Content-Type: application/json" --digest -i -X PATCH
"https://opsmgr.training/api/public/v1.0/groups/$GROUPID/clusters/$CLUSTERID" --data '
{
  "clusterName": "PRODUCTION"
}'
```
