MongoDB Ops Manager Manual

Release 1.6

MongoDB, Inc.

Dec 04, 2017

Contents

Ops Manager is a package for managing MongoDB deployments. Ops Manager provides Ops Manager Monitoring and Ops Manager Backup, which helps users optimize clusters and mitigate operational risk.

You can also download a PDF edition of the Ops Manager Manual.

Introduction Describes Ops Manager components and provides steps to install a test deployment.

Install Ops Manager Install Ops Manager.

Create New Deployments Set up servers and create MongoDB deployments.

Import Existing Deployments Import your existing MongoDB deployments to Ops Manager.

Manage Deployments Monitor, update, and manage your deployments.

Back Up Deployments Initiate and restore backups.

Security Describes Ops Manager security features.

Administration Configure and manage Ops Manager.

API Manage Ops Manager through the API.

Troubleshooting Troubleshooting advice for common issues.

Frequently Asked Questions Common questions about the operation and use of Ops Manager.

Reference Reference material for Ops Manager components and operations.

Release Notes Changelogs and notes on Ops Manager releases.

1 Ops Manager Introduction

Functional Overview Describes Ops Manager services and operations.

Ops Manager Components Describes Ops Manager components.

Install a Simple Test Ops Manager Set up a simple test installation in minutes.

1.1 Functional Overview

On this page

- *Automation*
- *Backup*

Overview

MongoDB Ops Manager is a service for managing, monitoring and backing up a MongoDB infrastructure. Ops Manager provides the services described here.

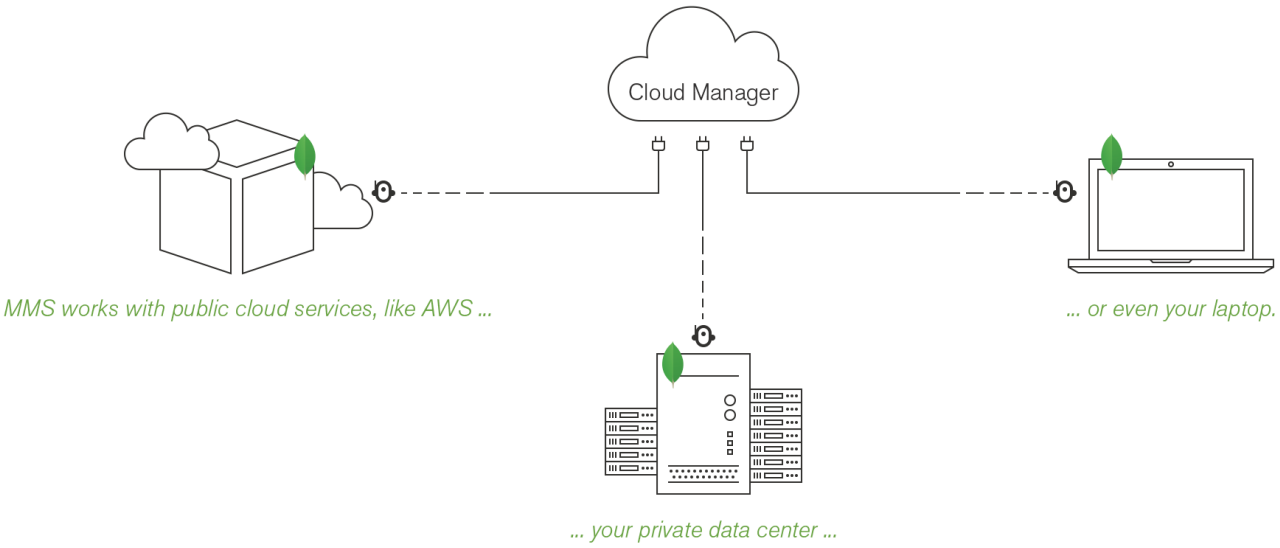
Monitoring

Ops Manager Monitoring provides real-time reporting, visualization, and alerting on key database and hardware indicators.

How it Works: A lightweight Monitoring Agent runs within your infrastructure and collects statistics from the nodes in your MongoDB deployment. The agent transmits database statistics back to Ops Manager to provide real-time reporting. You can set alerts on indicators you choose.

Automation

Ops Manager Automation provides an interface for configuring MongoDB nodes and clusters and for upgrading your MongoDB deployment.



How it Works: Automation Agents on each server maintain your deployments. The Automation Agent also maintains the Monitoring and Backup agents and starts, restarts, and upgrades the agents as needed.

Automation allows only one agent of each type per machine and will remove additional agents. For example, when maintaining Backup Agents, automation will remove a Backup Agent from a machine that has two Backup Agents.

Backup

Ops Manager Backup provides scheduled snapshots and point-in-time recovery of your MongoDB replica sets and sharded clusters.

How it Works: A lightweight Backup Agent runs within your infrastructure and backs up data from the MongoDB processes you have specified.

Data Backup

When you start Backup for a MongoDB deployment, the agent performs an initial sync of the deployment's data as if it were creating a new, "invisible" member of a replica set. For a sharded cluster the agent performs a sync of each shard's primary and of each config server. The agent ships initial sync and oplog data over HTTPS back to Ops Manager.

The Backup Agent then tails each replica set's oplog to maintain on disk a standalone database, called a *head database*. Ops Manager maintains one head database for each backed-up replica set. The head database is consistent with the original primary up to the last oplog supplied by the agent.

Backup performs the initial sync and the tailing of the oplog using standard MongoDB queries. The production replica set is not aware of the copy of the backup data.

Backup uses a `mongod` with a version equal to or greater than the version of the replica set it backs up.

Backup takes and stores snapshots based on a user-defined *snapshot retention policy*. Sharded clusters snapshots temporarily stop the balancer via the `mongos` so that they can insert a marker token into all shards and config servers in the cluster. Ops Manager takes a snapshot when the marker tokens appear in the backup data.

Compression and block-level de-duplication technology reduce snapshot data size. The snapshot only stores the differences between successive snapshots. Snapshots use only a fraction of the disk space required for full snapshots.

Data Restoration

Ops Manager Backup lets you restore data from a scheduled snapshot or from a selected point between snapshots. For sharded clusters you can restore from checkpoints between snapshots. For replica sets, you can restore from selected points in time.

When you restore from a snapshot, Ops Manager reads directly from the Backup Blockstore database and transfers files either through an HTTPS download link or by sending them via HTTPS or SCP.

When you restore from checkpoint or point in time, Ops Manager first creates a local restore of a snapshot from the blockstore and then applies stored oplogs until the specified point is reached. Ops Manager delivers the backup via the same HTTPS or SCP mechanisms.

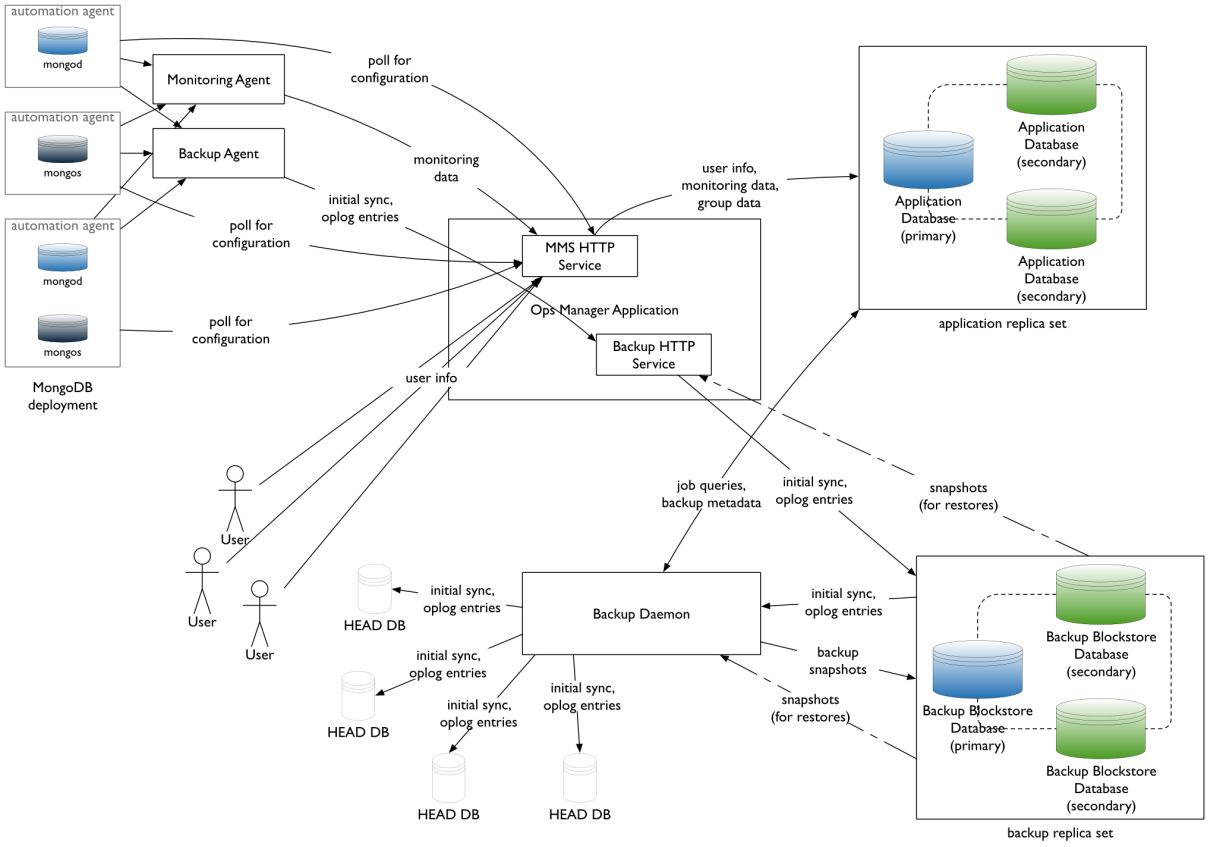
The amount of oplog to keep per backup is configurable and affects the time window available for checkpoint and point-in-time restores.

1.2 Ops Manager Components

On this page

An Ops Manager installation consists of the *Ops Manager Application* and optional *Backup Daemon*. Each package also requires a dedicated MongoDB database to hold operational data.

Network Diagram



Ops Manager Application

The front-end Ops Manager Application contains the UI the end user interacts with, as well as HTTPS services used by the Monitoring Agent and Backup Agent to transmit data to and from Ops Manager. All three components start automatically when the Ops Manager Application starts. These components are stateless. Multiple instances of the front-end package can run as long as each instance has the same configuration. Users and agents can interact with any instance.

For Monitoring, you **only** need to install the application package. The application package consists of the following components:

- *Ops Manager HTTP Service*
- *Backup HTTP Service*
- *Backup Alert Service*

Ops Manager HTTP Service

The HTTP server runs on port `8080` by default. This component contains the web interface for managing Ops Manager users, monitoring of MongoDB servers, and managing those server's backups. Users can sign up, create new accounts and groups, as well as join an existing group.

Backup HTTP Service

The HTTP server runs on port `8081` by default. The Backup HTTP Service contains a set of web services used by the Backup Agent. The agent retrieves its configuration from this service. The agent also sends back initial sync and oplog data through this interface. There is no user interaction with this service. The Backup HTTP service runs on port `8081` by default.

The Backup HTTP Service exposes an endpoint that reports on the state of the service and the underlying database to support monitoring of the Backup service. This status also checks the connections from the service to the *Ops Manager Application Database* and the *Backup Blockstore Database*. See *Backup HTTP Service Endpoint*.

Backup Alert Service

The Backup Alert Service watches the state of all agents, local copies of backed up databases, and snapshots. It sends email alerts as problems occur. The Backup Alert Service exposes a health-check endpoint. See *Backup Alert Service Endpoint*.

Backup Daemon

The Backup Daemon manages both the local copies of the backed-up databases and each backup's snapshots. The daemon does scheduled work based on data coming in to the Backup HTTP Service from the Backup Agents. No client applications talk directly to the daemon. Its state and job queues come from the *Ops Manager Application Database*.

The Backup Daemon's local copy of a backed-up deployment is called the *head database*. The daemon stores all its head databases in its `rootDirectory` path. To create each head database, the daemon's server acts as though it were an "invisible" secondary for each replica set designated for backup.

If you run multiple Backup Daemons, Ops Manager selects the Backup Daemon to use when a user enables backup for a deployment. The local copy of the deployment resides with that daemon's server.

The daemon will take scheduled snapshots and store the snapshots in the *Backup Blockstore database*. It will also act on restore requests by retrieving data from the Blockstore and delivering it to the requested destination.

Multiple Backup Daemons can increase your storage by scaling horizontally and can provide **manual** failover.

The Backup Daemon exposes a health-check endpoint. See *Backup Daemon Endpoint*.

Dedicated MongoDB Databases for Operational Data

Ops Manager uses dedicated MongoDB databases to store the Ops Manager Application's monitoring data and the Backup Daemon's snapshots. To ensure redundancy and high availability, the backing databases run as replica sets. The replica sets host **only** Ops Manager data. You must *set up the backing replica sets* before installing Ops Manager.

Ops Manager Application Database

This database contains application metadata used by the *Ops Manager Application*. The database stores:

- Monitoring data collected from Monitoring Agents.
- Metadata for Ops Manager users, groups, hosts, monitoring data, and backup state.

For topology and specifications, see *Ops Manager Application Database Hardware*.

Backup Blockstore Database

This database contains all snapshots of databases backed up and oplogs retained for point in time restores. The Backup Blockstore database requires disk space proportional to the backed-up databases.

Configure the Blockstore as a replica set to provide durability and automatic failover to the backup and restore components. The replica set must have at least three members that hold data.

You **cannot** back up the Blockstore database with Ops Manager Backup. To back up Ops Manager Backup, see *Back Up Ops Manager*.

For additional specifications, see *Ops Manager Backup Blockstore Database Hardware*.

1.3 Install a Simple Test Ops Manager Installation

On this page

Overview

To evaluate Ops Manager, you can quickly create a test installation by installing the *Ops Manager Application* and *Ops Manager Application Database* on a single server. This setup provides all the functionality of Ops Manager monitoring and automation but provides **no** failover or high availability. This is **not** a production setup.

Unlike a production installation, the simple test installation uses only one `mongod` for the Ops Manager Application database. In production, the database requires a dedicated replica set.

This procedure includes optional instructions to install Ops Manager Backup, in which case you would install the *Backup Daemon* and *Backup Blockstore database* on the same server as the other Ops Manager components. The Backup Blockstore database uses only one `mongod` and not a dedicated replica set, as it would in production.

This procedure installs the test deployment on servers running either **RHEL 6+** or **Amazon Linux**.

Procedure

Warning: This setup is not suitable for a production deployment.

To install Ops Manager for evaluation:

Step 1: Set up a RHEL 6+ or Amazon Linux server that meets the following requirements:

- The server must have 15 GB of memory and 50 GB of disk space for the root partition. You can meet the size requirements by using an Amazon Web Services EC2 `m3.xlarge` instance and changing the size of the root partition from 8 GB to 50 GB. When you log into the instance, execute "`df -h`" to verify the root partition has 50 GB of space.
- You must have root access to the server.

Step 2: Configure the `yum` package management system to install the latest stable release of MongoDB.

Issue the following command to set up a `yum` repository definition:

```
echo "[MongoDB]
name=MongoDB Repository
baseurl=http://downloads-distro.mongodb.org/repo/redhat/os/x86_64
gpgcheck=0
enabled=1" | sudo tee /etc/yum.repos.d/mongodb.repo
```

Step 3: Install MongoDB.

Issue the following command to install the latest stable release of MongoDB:

```
sudo yum install -y mongodb-org mongodb-org-shell
```

Step 4: Create the data directory for the Ops Manager Application database.

Issue the following two commands to create the data directory and change its ownership:

```
sudo mkdir -p /data/db
sudo chown -R mongod:mongod /data
```

OPTIONAL: To also install the Backup feature, issue following additional commands for the Backup Blockstore database:

```
sudo mkdir -p /data/backup
sudo chown mongod:mongod /data/backup
```

Step 5: Start the MongoDB backing instance for the Ops Manager Application database.

Issue the following command to start MongoDB as the `mongod` user. Start MongoDB on port `27017` and specify the `/data/db` for both data files and logs. Include the `--fork` option to run the process in the background and maintain control of the terminal.

```
sudo -u mongod mongod --port 27017 --dbpath /data/db --logpath /data/db/mongodb.log --
↪fork
```

OPTIONAL: To also install the Backup feature, issue following command to start a MongoDB instance similar to the other but on port `27018` and with the data directory and log path of the Backup Blockstore database:

```
sudo -u mongod mongod --port 27018 --dbpath /data/backup --logpath /data/backup/
↪mongodb.log --fork
```

Step 6: Download the Ops Manager Application package.

1. In a browser, go to http://www.mongodb.com/download.
2. Fill out and submit the subscription form.
3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation to contact MongoDB for production installs.
5. On the *MongoDB Ops Manager Downloads* page, copy the link address of the *RPM* link for Monitoring, Automation and Core. **OPTIONAL:** If you will install Backup, copy the link address of the *RPM* link for Backup as well.
6. Open a system prompt.
7. Download the Ops Manager Application package by issuing a `curl` command that uses the link address copied for the RPM for Monitoring, Automation and Core:

   ```
   curl -OL <link-address-for-monitoring-automation-core-rpm>
   ```

   OPTIONAL: Download the Backup Daemon package by issuing a `curl` command that uses the link address copied for the Backup RPM:

   ```
   curl -OL <link-address-for-backup-rpm>
   ```

Step 7: Install the Ops Manager Application.

Install the *Monitoring, Automation and Core* RPM package that you downloaded. Issue the `rpm --install` command with root privileges and specify the package name:

```
sudo rpm --install <rpm-package-for-monitoring-automation-core>
```

OPTIONAL: To also install Backup, issue the `rpm --install` command with root privileges and specify the *Backup* RPM package:

```
sudo rpm --install <rpm-package-for-backup>
```

Step 8: Get your server's public IP address.

If you are using an EC2 instance, this is available on the instance's `Description` tab.

Alternately, you can get the public IP address by issuing the following:

```
curl -s http://whatismijnip.nl |cut -d " " -f 5
```

Step 9: Configure the Ops Manager Application.

Edit `/opt/mongodb/mms/conf/conf-mms.properties` with root privileges and set the following options. For detailed information on each option, see *Ops Manager Configuration Files*.

Set *`mms.centralUrl`* and *`mms.backupCentralUrl`* as follows, where `<ip_address>` is the IP address of the server running the Ops Manager Application.

```
mms.centralUrl=http://<ip_address>:8080
mms.backupCentralUrl=http://<ip_address>:8081
```

Set the following *Email Address Settings* as appropriate. You can use the same email address throughout, or specify a different address for each field.

```
mms.fromEmailAddr=<email_address>
mms.replyToEmailAddr=<email_address>
mms.adminFromEmailAddr=<email_address>
mms.adminEmailAddr=<email_address>
mms.bounceEmailAddr=<email_address>
```

Set the *`mongo.mongoUri`* option to the port hosting the Ops Manager Application database:

```
mongo.mongoUri=mongodb://localhost:27017
```

OPTIONAL: If you installed the Backup Daemon, edit `/opt/mongodb/mms-backup-daemon/conf/conf-daemon.properties` with root privileges and set the *`mongo.mongoUri`* value to the port hosting the Ops Manager Application database:

```
mongo.mongoUri=mongodb://localhost:27017
```

Step 10: Start the Ops Manager Application.

To start the Ops Manager Application, issue the following:

```
sudo service mongodb-mms start
```

OPTIONAL: To start the Backup Daemon, issue the following:

```
sudo service mongodb-mms-backup-daemon start
```

Step 11: Open the Ops Manager home page.

In a browser, enter the following URL, where `<ip_address>` is the IP address of the server:

```
http://<ip_address>:8080
```

Step 12: To begin testing Ops Manager, click *Register* and follow the prompts to create the first user and group.

The first user receives *Global Owner* permissions for the test install.

Step 13: At the Welcome page, follow the prompts to set up Automation or Monitoring.

Automation lets you define a MongoDB deployment through the Ops Manager interface and rely on the *Automation Agent* to construct the deployment. If you select Automation, Ops Manager prompts you to download the Automation Agent and Monitoring Agent to the server.

Monitoring lets you manage a MongoDB deployment through the Ops Manager interface. If you select Monitoring, Ops Manager prompts you to download only the *Monitoring Agent* to the server.

OPTIONAL: If you installed the Backup Daemon, do the following to enable Backup: click the *Admin* link in at the top right of the Ops Manager page and click the *Backup* tab. In the *<hostname>:<port>* field, enter `localhost:27018` and click *Save*.

2 Install Ops Manager

Installation Checklist Prepare for your installation.

Example Installation Diagrams Provides diagrams of Ops Manager deployments.

Hardware and Software Requirements Describes the hardware and software requirements for the servers that run the Ops Manager components, including the servers that run the backing MongoDB replica sets.

Deploy Application and Backup Databases Set up the Ops Manager Application Database and Backup Database.

Install Ops Manager Operating-system specific instructions for installing the Ops Manager Application and the Backup Daemon.

Upgrade Ops Manager Operating-system specific instructions for upgrading the Ops Manager Application and the Backup Daemon.

Configure Offline Binary Access Configure local mode for an installation that uses Automation but has no internet access for downloading the MongoDB binaries.

Configure High Availability Configure the Ops Manager application and components to be highly available.

Configure Backup Jobs and Storage Manage and control the jobs used by the Backup system to create snapshots.

Test Ops Manager Monitoring Set up a replica set for testing Ops Manager Monitoring.

2.1 Installation Checklist

On this page

Overview

You must make the following decisions before you install Ops Manager. During the install procedures you will make choices based on your decisions here.

If you have not yet read the *Ops Manager Components* page, please do so for a description of the system's components.

The sequence for installing Ops Manager is to:

- Plan your installation according to the questions on this page.
- Provision servers that meet the *Hardware and Software Requirements*
- Set up the Ops Manager Application Database and optional Backup Database.
- Install the Ops Manager Application and optional Backup Daemon.

Note: To install a simple evaluation deployment on a single server, see *Install a Simple Test Ops Manager Installation*.

Topology Decisions

Do you require durability and/or high availability?

Ops Manager stores application metadata and snapshots in the Ops Manager Application Database and Backup Database respectively. To provide data durability, run each database as a three-member replica set on multiple servers.

To provide high availability for write operations to the databases, set up each replica set so that all three members hold data. This way, if a member is unreachable the replica set can still write data. Ops Manager uses `w:2` write concern, which requires acknowledgement from the primary and one secondary for each write operation.

To provide high availability for the Ops Manager Application, run at least two instances of the application and use a load balancer. For more information, see *Configure a Highly Available Ops Manager Application*.

The following tables describe the pros and cons for each combination of durability and high availability.

Non-Durable, Test Install

This is a non-durable install that runs on one server. If you lose the server, you must start over from scratch.

Pros:	Needs only needs one server.
Cons:	If you lose the server, you lose everything: users and groups, metadata, backups, automation configurations, stored monitoring metrics, etc.

Durable Production Install

This install runs on at least three servers and provides durability for your metadata and snapshots. The replica sets for the Ops Manager Application Database and the Backup Database are each made up of two data-bearing members and an arbiter. This installation does not provide high availability.

Pros:	Can run on as few as three servers. Ops Manager metadata and backups are durable from the perspective of the Ops Manager Application.
Cons:	No high availability, neither for the databases nor the application: 1. If the Ops Manager Application Database or the Backup Database loses a data-bearing member, the data is durable but you must restart the member to gain back full Ops Manager functionality. For the Backup Database, Ops Manager will not write new snapshots until the member is again running. 2. Loss of the Ops Manager Application requires you to manually start a new Ops Manager Application. No Ops Manager functionality is available while the application is down.

Durable Production Install with Highly Available Backup and Application Data

This install requires at least three servers. The replica sets for the Ops Manager Application Database and the Backup Database each comprise at least three *data-bearing* members. This requires more storage and memory than for the *Durable Production Install*.

Pros:	You can lose a member of the Ops Manager Application Database or Backup Database and still maintain Ops Manager availability. No Ops Manager functionality is lost while the member is down.
Cons:	Loss of the Ops Manager Application requires you to manually start a new Ops Manager Application. No Ops Manager functionality is available while the application is down.

Durable Production Install with a Highly Available Ops Manager Application

This runs multiple Ops Manager Applications behind a load balancer and requires infrastructure outside of what Ops Manager offers. For details, see *Configure a Highly Available Ops Manager Application*.

Pros:	Ops Manager continues to be available even when any individual server is lost.
Cons:	Requires a larger number of servers, and requires a load balancer capable of routing traffic to available application servers.

Will you deploy managed MongoDB instances on servers that have no internet access?

If you use Automation and if the servers where you will deploy MongoDB do not have internet access, then you must configure Ops Manager to locally store and share the binaries used to deploy MongoDB so that the Automation agents can download them directly from Ops Manager.

You must configure local mode and store the binaries before you create the first managed MongoDB deployment from Ops Manager. For more information, see *Configure Local Mode if Ops Manager has No Internet Access*.

Will you use a proxy for the Ops Manager application's outbound network connections?

If Ops Manager will use a proxy server to access external services, you must configure the proxy settings in Ops Manager's `conf-mms.properties` configuration file. If you have already started Ops Manager, you must restart after configuring the proxy settings.

Security Decisions

Will you use authentication and/or SSL for the connections to the backing databases?

If you will use authentication or SSL for connections to the Ops Manager Application Database and Backup Database, you must configure those options on each database when *deploying the database* and then you must configure Ops Manager with the necessary certificate information for accessing the databases. For details, see *Configure the Connections to the Backing MongoDB Instances*

Will you use LDAP for user authenticate to Ops Manager?

If you will use LDAP for user management, you must configure LDAP authentication **before** you register any Ops Manager user or group. If you have already created an Ops Manager user or group, you must start from scratch with a fresh Ops Manager install.

During the procedure to install Ops Manager, you are given the option to configure LDAP before creating users or groups. For details on LDAP authentication, see *Configure Users and Groups with LDAP for Ops Manager*.

Will you use SSL (HTTPS) for connections to the Ops Manager application?

If you will use SSL for connections to Ops Manager from agents, users, and the API, then you must configure Ops Manager to use SSL. The procedure to install Ops Manager includes the option to configure SSL access.

Backup Decisions

Will the servers that run your Backup Daemons have internet access?

If the servers that run your Backup Daemons have no internet access, you must configure offline binary access for the Backup Daemon before running the Daemon. The *install procedure* includes the option to configure offline binary access.

Are certain backups required to be in certain data centers?

If you need to assign backups of particular MongoDB deployments to particular data centers, then each data center requires its own Ops Manager Application, Backup Daemon, and Backup Agent. The separate Ops Manager Application instances must share a single dedicated Ops Manager Application Database. The Backup Agent in each data center must use the URL for its local Ops Manager Application, which you can configure through either different hostnames or split-horizon DNS. For detailed requirements, see *Configure Multiple Blockstores in Multiple Data Centers*.

2.2 Example Installation Diagrams

On this page

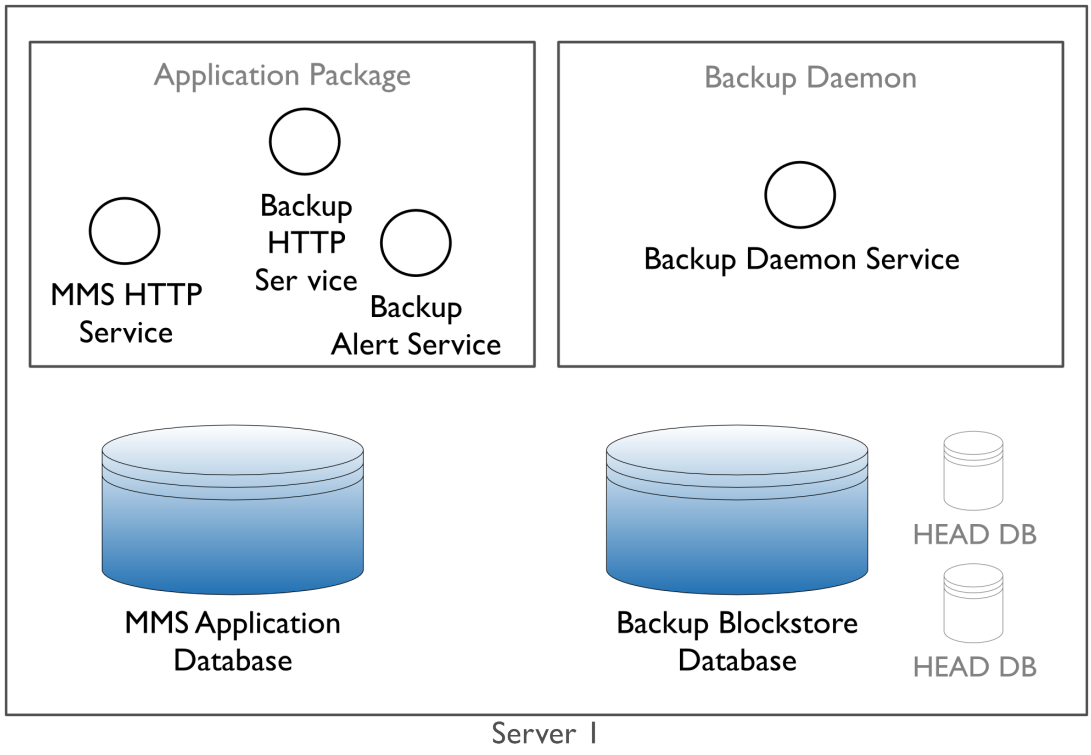
- *Durable, Highly Available Install with Multiple Backup Databases*

Overview

The following diagrams show example Ops Manager deployments.

Non-Durable, Test Install on a Single Server

For a test deployment, you can deploy all of the Ops Manager components to a single server, as described in *Install a Simple Test Ops Manager Installation*. Ensure you *configure the appropriate ulimits* for the deployment.

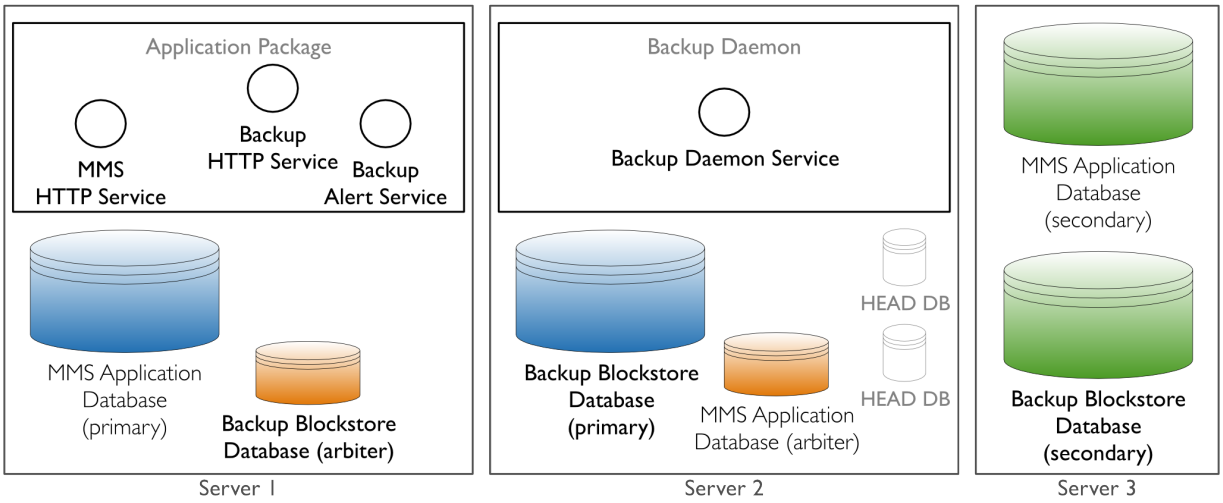


The head databases are dynamically created and maintained by the Backup Daemon. They reside on the disk partition specified in the `conf-daemon.properties` file.

Durable Production Install

The basic deployment provides durability in case of failure by keeping a redundant copy of the application data and snapshots. However, the basic deployment does not provide high availability and cannot accept writes to the backing databases in the event that a replica set memeber is lost. See *Durable, Highly Available Install with Multiple Backup Databases* for a deployment that can continue to accept writes with the loss of a member.

Server 1 must satisfy the combined *hardware and software requirements* for the Ops Manager Application hardware and Ops Manager Application Database hardware.



Server 2 must satisfy the combined hardware and software requirements for the Backup Daemon hardware and Backup Blockstore database hardware. The Backup Daemon automatically creates and maintains the *head* databases. These databases reside on the disk partition specified in the `conf-daemon.properties` file. Do not place the head databases on the same disk partition as the Backup Blockstore database, as this will reduce Backup's performance.

Server 3 hosts replica set members for the Backup Blockstore and Ops Manager Application databases. Replica sets provide data redundancy and are strongly recommended, but are not required for Ops Manager. Server 3 must satisfy the combined hardware and software requirements for the Ops Manager Application database hardware and Backup Blockstore database hardware.

For an example tutorial on installing the minimally viable Ops Manager installation on RHEL 6+ or Amazon Linux, see /tutorial/install-basic-deployment.

Durable, Highly Available Install with Multiple Backup Databases

The following is a highly available deployment that you can scale out to add additional Backup Databases.

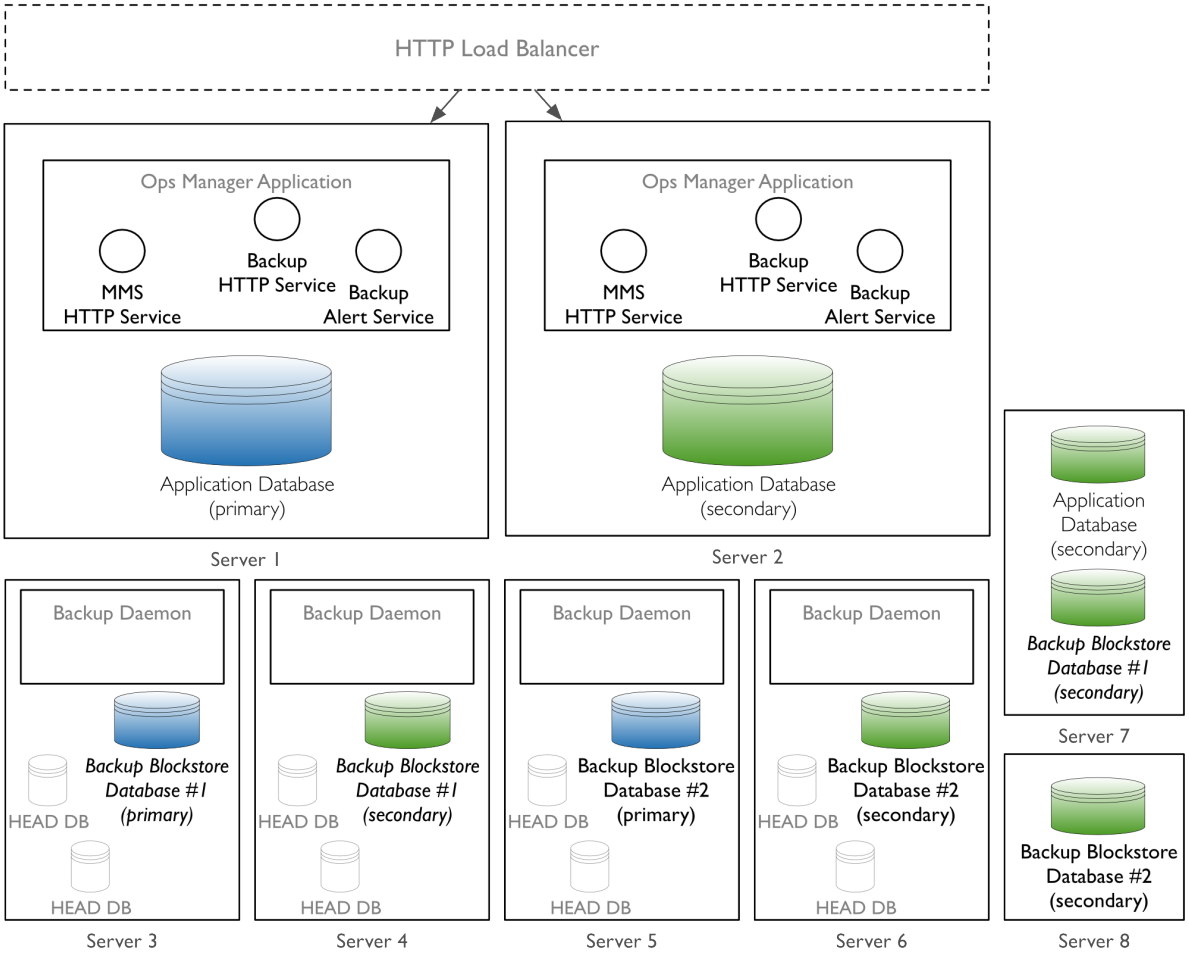
The deployment includes two servers that host the Ops Manager Application and the Ops Manager Application Database, four servers that host two Backup deployments, and an additional server to host the arbiters for each replica set.

Deploy an HTTP Load Balancer to balance the HTTP traffic for the Ops Manager HTTP Service and Backup service. Ops Manager does not supply an HTTP Load Balancer: you must deploy and configure it yourself.

All of the software services need to be able to communicate with the Ops Manager Application databases, and the Backup Blockstore databases. Configure your firewalls to allow traffic between these servers on the *appropriate ports*.

- **Server 1** and **Server 2** must satisfy the combined *hardware and software requirements* for the Ops Manager Application hardware and Ops Manager Application Database hardware.
- **Server 3**, **Server 4**, **Server 5**, and **Server 6** must satisfy the combined hardware and software requirements for the Backup Daemon hardware and Backup Database hardware.

  The Backup Daemon creates and maintains the *head* databases. They reside on the disk partition specified in the `conf-daemon.properties` file. Only the Backup Daemon needs to communicate with the head databases. As such, their `net.bindIp` value is `127.0.0.1` to prevent external communication. `net.bindIp` specifies the IP address that `mongod` and `mongos` listens to for connections coming from applications.


HTTP Load Balancer
Ops Manager Application
MMS
HTTP Service
Backup
HTTP Service
Backup
Alert Service
Application Database
(primary)
Server 1
Ops Manager Application
MMS
HTTP Service
Backup
HTTP Service
Backup
Alert Service
Application Database
(secondary)
Server 2
Application
Database
(secondary)
Backup Blockstore
Database #1
(secondary)
Server 7
Backup Blockstore
Database #2
(secondary)
Server 8
Backup Daemon
Backup Blockstore
Database #1
(primary)
HEAD DB
HEAD DB
Server 3
Backup Daemon
Backup Blockstore
Database #1
(secondary)
HEAD DB
HEAD DB
Server 4
Backup Daemon
Backup Blockstore
Database #2
(primary)
HEAD DB
HEAD DB
Server 5
Backup Daemon
Backup Blockstore
Database #2
(secondary)
HEAD DB
HEAD DB
Server 6

For best performance, each Backup server should have 2 partitions. One for the Backup Blockstore database, and one for the head databases.

- **Server 7** and **Server 8** host secondaries for the Ops Manager Application database, and for the two Backup Blockstore databases. They must satisfy the combined hardware and software requirements for the databases.

To deploy Ops Manager with high availability, see: *Configure a Highly Available Ops Manager Application*.

2.3 Ops Manager Hardware and Software Requirements

On this page

This page describes the hardware and software requirements for the servers that run the *Ops Manager Components*, including the servers that run the backing MongoDB replica sets.

The servers that run the Backup Daemon and the backing replica sets **must** also meet the configuration requirements in the MongoDB Production Notes in addition to the requirements on this page. The Production Notes include information on ulimits, NUMA, Transparent Huge Pages (THP), and other configuration options.

Warning: Failure to configure servers according to the MongoDB Production Notes can lead to production failure.

This page also includes requirements for the EC2 security group used when installing on AWS servers.

Hardware Requirements

Each server must meet the sum of the requirements for all its components.

Ops Manager Monitoring and Automation require servers for the following components:

- *Ops Manager Application*.
- *Ops Manager Application Database* replica set members

Note: Usually the Ops Manager Application and one of the Application database's replica set members run on the same server.

If you run Backup, Ops Manager also requires servers for the following:

- *Backup Daemon*
- *Backup Blockstore database* replica set members

Note: The following requirements are specific to a given component. You must add together the requirements for the components you will install. For example, the requirements for the Ops Manager Application do not cover the Ops Manager Application database.

Ops Manager Application Hardware

The Ops Manager Application requires the hardware listed here.

Number of Monitored Hosts	CPU Cores	RAM
Up to 400 monitored hosts	4+	15 GB
Up to 2000 monitored hosts	8+	15 GB
More than 2000 hosts	Contact MongoDB Account Manager	Contact MongoDB Account Manager

Ops Manager Application Database Hardware

The *Ops Manager Application Database* holds monitoring and other metadata for the *Ops Manager Application*.

The database runs as a three-member replica set. If you cannot allocate space for three data-bearing members, the third member can be an arbiter, but keep in mind that Ops Manager uses `w:2` write concern, which reports a write operation as successful after acknowledgement from the primary and one secondary. If you use a replica set with fewer than 3 data-bearing members, and if you lose one of the data-bearing members, MongoDB blocks write operations, meaning the *Ops Manager Application Database* has durability but not high availability.

Run the replica set on dedicated servers. You can optionally run one member of the replica set on the same physical server as the Ops Manager Application.

For a *test deployment*, you can use a MongoDB standalone in place of a replica set.

Each server that hosts a MongoDB process for the Ops Manager Application database **must** comply with the Production Notes in the MongoDB manual. The Production Notes include important information on ulimits, NUMA, Transparent Huge Pages (THP), and other configuration options.

> **Warning:** Failure to configure servers according to the MongoDB Production Notes can lead to production failure.

Each server also requires the following:

Number of Monitored Hosts	RAM	Disk Space
Up to 400 monitored hosts	8 GB additional RAM beyond the RAM required for the Ops Manager Application	200 GB of storage space
Up to 2000 monitored hosts	15 GB additional RAM beyond the RAM required for the Ops Manager Application	500 GB of storage space
More than 2000 hosts	Contact MongoDB account manager	Contact MongoDB account manager

For the best results use SSD-backed storage.

Ops Manager Backup Daemon Hardware

The Backup Daemon server must meet the requirements in the table below and **also must meet** the configuration requirements in the MongoDB Production Notes. The Production Notes include information on ulimits, NUMA, Transparent Huge Pages (THP), and other options.

Warning: Failure to configure servers according to the MongoDB Production Notes can lead to production failure.

If you wish to install the Backup Daemon on the same physical server as the Ops Manager Application, the server must satisfy these requirements separately from the requirements in *Ops Manager Application Hardware*.

The server running the Backup Daemon acts like a hidden secondary for every replica set assigned to it, receiving the streamed oplog entries each replica set's primary. However, the Backup Daemon differs from a hidden secondary in that the replica set is not aware of it.

The server must have the disk space and write capacity to maintain the replica sets plus the space to store an additional copy of the data to support point-in-time restore. Typically, the Backup Daemon must be able to store 2 to 2.5 times the sum of the size on disk of all the backed-up replica sets, as it also needs space locally to build *point-in-time* restores.

Before installing the Backup Daemon, we recommend contacting your MongoDB Account Manager for assistance in estimating the storage requirements for your Backup Daemon server.

Number of Hosts	CPU Cores	RAM	Disk Space	Storage IOPS/s
Up to 200 hosts	4+ 2Ghz+	15 GB additional RAM	Contact MongoDB Account Manager	Contact MongoDB Account Manager

Ops Manager Backup Blockstore Database Hardware

Blockstore servers store snapshots of MongoDB deployments. Only provision Blockstore servers if you are deploying Ops Manager Backup.

Replica Set for the Blockstore Database

Backup requires a separate, **dedicated** MongoDB replica set to hold snapshot data. This cannot be a replica set used for any purpose other than holding the snapshots.

For durability, the replica set must have at least two data-bearing members. For high availability the replica set must have at least three data-bearing members.

Note: Ops Manager uses `w:2` write concern, which reports a write operation as successful after acknowledgement from the primary and one secondary. If you use a replica set with two data-bearing members and an arbiter, and you lose one of the data-bearing members, write operations will be blocked.

For *testing only* you may use a standalone MongoDB deployment in place of a replica set.

Server Size for the Blockstore Database

Snapshots are compressed and de-duplicated at the block level in the Blockstore database. Typically, depending on data compressibility and change rate, the replica set must run on servers with enough capacity to store 2 to 3 times the total backed-up production data size.

Contact your MongoDB Account Manager for assistance in estimating the use-case and workload-dependent storage requirements for your Blockstore servers.

Configuration Requirements from the MongoDB Production Notes

Each server that hosts a MongoDB process for the Blockstore database **must** comply with the Production Notes in the MongoDB manual. The Production Notes include important information on ulimits, NUMA, Transparent Huge Pages (THP), and other configuration options.

> **Warning:** Failure to configure servers according to the MongoDB Production Notes can lead to production failure.

Other Requirements for the Blockstore Databsase

For each data-bearing member of the replica set member

CPU Cores	RAM	Disk Space	Storage IOPS
4 x 2ghz+	8 GB of RAM for every 1 TB disk of Blockstore to provide good snapshot and restore speed. Ops Manager defines 1 TB of Blockstore as 1024^4 bytes.	Contact your MongoDB Account Manager.	Medium grade HDDs should have enough I/O throughput to handle the load of the Blockstore.

EC2 Security Groups

If you install on AWS servers, you must have at least one EC2 security group configured with the following inbound rules:

- An SSH rule on the `ssh` port, usually port `22`, that allows traffic from all IPs. This is to provide administrative access.
- A custom TCP rule that allows connection on ports `8080` and `8081` on the server that runs the Ops Manager Application. This lets users connect to Ops Manager.
- A custom TCP rule that allows traffic on all MongoDB ports from any member of the security group. This allows communication between the various Ops Manager components. MongoDB usually uses ports between `27000` and `28000`.

Software Requirements

Operating System

The Ops Manager Application and Backup Daemon(s) can run on 64-bit versions of the following operating systems:

- CentOS 5 or later
- Red Hat Enterprise Linux 5 or later
- SUSE 11 or Later
- Amazon Linux AMI (latest version only)
- Ubuntu 12.04 or later
- Windows Server 2008 R2 or later

Warning: Ops Manager supports *Monitoring* and *Backup* on Windows but does not support Automation on Windows.

Ulimits

The Ops Manager packages automatically raise the open file, max user processes, and virtual memory ulimits. On Red Hat, be sure to check for a `/etc/security/limits.d/90-nproc.conf` file that may override the max user processes limit. If the `/etc/security/limits.d/90-nproc.conf` file exists, remove it before continuing.

See MongoDB ulimit Settings for recommended ulimit settings.

Warning: Always refer to the MongoDB Production Notes to ensure healthy server configurations.

Authentication

If you are using LDAP for user authentication to Ops Manager (as described in *Configure Users and Groups with LDAP for Ops Manager*), you must enable LDAP **before setting up** Ops Manager, beyond starting the service.

Important: You **cannot** enable LDAP once you have opened the Ops Manager user interface and registered the first user. You can enable LDAP **only** on a completely blank, no-hosts, no-users installation.

MongoDB

The *Ops Manager Application Database* and *Backup Blockstore Database* must run on MongoDB 2.4.9 or later.

Note: Ops Manager 1.8.0, when released, **will not** support MongoDB 2.4 for the Ops Manager Application database and Backup Blockstore database.

Your **backed-up** sharded cluster deployments must run at least MongoDB 2.4.3 or later. Your backed-up replica set deployments must run at least MongoDB 2.2 or later.

Web Browsers

Ops Manager supports clients using the following browsers:

- Chrome 8 and greater
- Firefox 12 and greater
- IE 9 and greater
- Safari 6 and greater

The Ops Manager Application will display a warning on non-supported browsers.

SMTP

Ops Manager requires email for fundamental server functionality such as password reset and alerts.

Many Linux server-oriented distributions include a local SMTP server by default, for example, Postfix, Exim, or Sendmail. You also may configure Ops Manager to send mail via third party providers, including Gmail and Sendgrid.

SNMP

If your environment includes SNMP, you can configure an SMNP trap receiver with periodic heartbeat traps to monitor the internal health of Ops Manager. Ops Manager uses SNMP v2c.

For more information, see *Configure SNMP Heartbeat Support*.

2.4 Deploy Backing MongoDB Replica Sets

On this page

Overview

Ops Manager uses two dedicated replica sets to store operational data: one replica set to store the *Ops Manager Application Database* and another to store the *Backup Blockstore Database*. If you use multiple application or blockstore databases, set up separate replica sets for each database.

The backing MongoDB replica sets are dedicated to operational data and **must store no other data**.

Replica Sets Requirements

Each replica set that hosts the Ops Manager Application Database or a Backup Database:

- Stores data in support of Ops Manager operations only **and stores no other data**.
- Must run on a server that meets the *server prerequisites below*.
- Must run MongoDB 2.4.9 or later.
- Must use the MMAPv1 storage engine.
- Must have the MongoDB `security.javascriptEnabled` set to `true`, which is the default. The Ops Manager Application uses the `$where` query and requires this setting be enabled on the Ops Manager Application database.
- Must **not** run the MongoDB `notablescan` option.

Server Prerequisites

The servers that run the replica sets must meet the following requirements.

- The hardware requirements described in *Ops Manager Application Database Hardware* or *Ops Manager Backup Blockstore Database Hardware*, depending on which database the server will host. If a server hosts other Ops Manager components in addition to the database, you must sum the hardware requirements for each component to determine the requirements for the server.
- The system requirements in the MongoDB Production Notes. The Production Notes include important information on ulimits, NUMA, Transparent Huge Pages (THP), and other configuration options.
- The MongoDB ports requirements described in *Firewall Configuration*. Each server's firewall rules must allow access to the required ports.
- **RHEL servers only**: if the `/etc/security/limits.d` directory contains the `90-nproc.conf` file, remove the file. The file overrides `limits.conf`, decreasing `ulimit` settings. Issue the following command to remove the file:

```
sudo rm /etc/security/limits.d/90-nproc.conf
```

Procedures

Install MongoDB on Each Server

Warning: Failure to configure servers according to the MongoDB Production Notes can lead to production failure.

Use servers that meet the above *prerequisites*.

The following procedure assumes that you are installing MongoDB on a server running Red Hat Enterprise Linux (RHEL). If you are installing MongoDB on another Linux operating system, or you prefer to use `cURL` rather than `yum`, refer to the installation guides in the MongoDB manual.

Step 1: Create a repository file on each server by issuing the following command:

```
echo "[mongodb-org-3.0]
name=MongoDB Repository
baseurl=http://repo.mongodb.org/yum/redhat/\$releasever/mongodb-org/3.0/x86_64/
gpgcheck=0
enabled=1" | sudo tee -a /etc/yum.repos.d/mongodb-org-3.0.repo
```

Step 2: Install MongoDB on each server by issuing the following command:

```
sudo yum install -y mongodb-org mongodb-org-shell
```

Deploy a Backing Replica Set

This procedure deploys a three-member replica set dedicated to store either the Ops Manager Application database or Backup Blockstore database. Deploy the replica set to three servers that meet the *requirements* for the database.

Repeat this procedure for each backing replica set that your installation requires.

Step 1: Create a data directory on each server.

Create a data directory on each server and set `mongod` as the directory's owner. For example:

```
sudo mkdir -p /data
sudo chown mongod:mongod /data
```

Step 2: Start each MongoDB process.

For each replica set member, start the `mongod` process and specify `mongod` as the user. Start each process on its own dedicated port and point to the data directory you created in the previous step. Specify the same replica set for all three members.

The following command starts a MongoDB process as part of a replica set named `operations` and specifies the `/data` directory.

```
sudo -u mongod mongod --port 27017 --dbpath /data --replSet operations --logpath /
↪data/mongodb.log --fork
```

Step 3: Connect to the MongoDB process on the server that will initially host the database's primary.

For example, the following command connects on port `27017` to a MongoDB process running on `mongodb1.example.net`:

```
mongo --host mongodb1.example.net --port 27017
```

When you connect, the MongoDB shell displays its version as well as the database to which you are connected.

Step 4: Initiate the replica set.

To initiate the replica set, issue the following command:

```
rs.initiate()
```

MongoDB returns `1` upon successful completion, and creates a replica set with the current `mongod` as the initial member.

The server on which you initiate the replica set becomes the initial primary. The `mongo` shell displays the replica set member's state in the prompt.

MongoDB supports automatic failover, so this `mongod` may not always be the primary.

Step 5: Add the remaining members to the replica set.

In a `mongo` shell connected to the primary, use the rs.add() method to add the other two replica set members. For example, the following adds the `mongod` instances running on `mongodb2.example.net:27017` and `mongodb3.example.net:27017` to the replica set:

```
rs.add('mongodb2.example.net:27017')
rs.add('mongodb3.example.net:27017')
```

Step 6: Verify the replica set configuration.

To verify that the configuration includes the three members, issue rs.conf():

```
rs.conf()
```

The method returns output similiar to the following.

```
{
    "_id" : "operations",
    "version" : 3,
    "members" : [
        {
            "_id" : 0,
            "host" : "mongodb1.example.net:27017"
        },
        {
            "_id" : 1,
            "host" : "mongodb2.example.net:27017"
        },
        {
            "_id" : 2,
            "host" : "mongodb3.example.net:27017",
        }
    ]
}
```

Optionally, run rs.status() </reference/method/rs.status to verify the status of the replica set and see the state of each replica set member.

For more information on deploying replica sets, see Deploy a Replica Set in the MongoDB manual.

2.5 Install Ops Manager

Install with DEB Packages Install Ops Manager on Debian and Ubuntu systems.

Install with RPM Packages Install Ops Manager on Red Hat, Fedora, CentOS, and Amazon AMI Linux.

Install from Archives on Linux Install Ops Manager on other Linux systems, without using package management.

Install on Windows Install Ops Manager on Windows.

Install Ops Manager with `deb` Packages

On this page

Overview

To install Ops Manager you install the *Ops Manager Application* and the optional *Backup Daemon*. This tutorial describes how to install both using `deb` packages. The Ops Manager Application monitors MongoDB deployments, and the Backup Daemon creates and stores deployment snapshots.

If you are instead upgrading an existing deployment, please see *Upgrade Ops Manager*.

> **Warning:** To use Ops Manager 1.6 Automation to manage MongoDB Enterprise deployments, you must **first install** the *MongoDB Enterprise dependencies* to each server that runs MongoDB Enterprise. You must install the dependencies to the servers before using Automation.
>
> Note that Automation **does not yet support** use of the MongoDB Enterprise advanced security features (SSL, LDAP, Kerberos, and auditing). Automation will support these features in the next major Ops Manager release.
>
> To run MongoDB Enterprise with advanced security now, deploy MongoDB Enterprise manually (not through Automation) and use Ops Manager only for Monitoring and Backup.

Prerequisites

Deploy Servers

Prior to installation, you must set up servers for the entire Ops Manager deployment, including the Ops Manager Application, the optional Backup Daemon, and the *backing replica sets*. For deployment diagrams, see *Example Installation Diagrams*.

Deploy servers that meet the hardware requirements described in *Ops Manager Hardware and Software Requirements*. Servers for the Backup Daemon and the backing replica sets must also comply with the Production Notes in the MongoDB manual. Configure as many servers as needed for your deployment.

> **Warning:** Failure to configure servers according to the MongoDB Production Notes can lead to production failure.

Deploy MongoDB

Install MongoDB on the servers that will store the *Ops Manager Application Database* and *Backup Blockstore Database*. The Backup Blockstore database is required only if you run the Backup Daemon. The databases require dedicated MongoDB instances. Do **not** use MongoDB installations that store other data.

Install separate MongoDB instances for the two databases and install the instances as replica sets. Ensure that firewall rules on the servers allow access to the *ports* that the instances runs on.

The Ops Manager Application and Backup Daemon must authenticate to their databases as a MongoDB user with appropriate access. The user must have the following roles:

- `readWriteAnyDatabase`
- `dbAdminAnyDatabase`.
- `clusterAdmin` if the database is a sharded cluster, otherwise `clusterMonitor`

Install Procedures

You must have administrative access on the machines to which you install.

Install and Start the Ops Manager Application

Step 1: Download the latest version of the Ops Manager Application package.

1. In a browser, go to http://www.mongodb.com/download.
2. Submit the subscription form.
3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation to contact MongoDB for production installs.
5. On the *MongoDB Ops Manager Downloads* page, copy the link address of the "Monitoring, Automation and Core" *DEB* link.
6. Open a system prompt.
7. Download the Ops Manager Application package by issuing a `curl` command that uses the copied link address:

   ```
   curl -OL <link-address-for-monitoring-automation-core-deb>
   ```

   The downloaded package is named `mongodb-mms-<version>.x86_64.deb`, where `<version>` is the version number.

Step 2: Install the Ops Manager Application package.

Install the `.deb` package by issuing the following command, where `<version>` is the version of the `.deb` package:

```
sudo dpkg --install mongodb-mms_<version>_x86_64.deb
```

When installed, the base directory for the Ops Manager software is `/opt/mongodb/mms/`. The `.deb` package creates a new system user `mongodb-mms` under which the server will run.

Step 3: Configure Monitoring.

Open `<install-directory>/conf/conf-mms.properties` with root privileges and set values for the settings described in this step. For detailed information on each setting, see the *Ops Manager Configuration Files* page.

Set *`mms.centralUrl`* and *`mms.backupCentralUrl`* as follows, where `<host>` is the fully qualified domain name of the server running the Ops Manager Application.

```
mms.centralUrl=http://<host>:8080
mms.backupCentralUrl=http://<host>:8081
```

Set the following *Email Address Settings* as appropriate. Each may be the same or different.

```
mms.fromEmailAddr=<email_address>
mms.replyToEmailAddr=<email_address>
mms.adminFromEmailAddr=<email_address>
mms.adminEmailAddr=<email_address>
mms.bounceEmailAddr=<email_address>
```

Set *mongo.mongoUri* to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

If you use HTTPS to encrypt user connections to Ops Manager, set *mms.https.PEMKeyFile* to a PEM file containing an X509 certificate and private key, and set *mms.https.PEMKeyFilePassword* to the password for the certificate. For example:

```
mms.https.PEMKeyFile=<path_and_name_of_pem_file>
mms.https.PEMKeyFilePassword=<password_for_pem_file>
```

To configure authentication, email, and other optional settings, see *Ops Manager Configuration Files*. To run the Ops Manager application in a highly available configuration, see *Configure a Highly Available Ops Manager Application*.

Step 4: Start the Ops Manager Application.

Issue the following command:

```
sudo service mongodb-mms start
```

Step 5: Open the Ops Manager home page and register the first user.

To open the home page, enter the following URL in a browser, where `<host>` is the fully qualified domain name of the server:

```
http://<host>:8080
```

Click the *Register* link and follow the prompts to register the first user and create the first group. The first user is automatically assigned the *Global Owner* role. When you finish, you are logged into the Ops Manager Application as the new user. For more information on creating and managing users, see *Manage Ops Manager Users*.

Step 6: At the Welcome page, follow the prompts to set up Automation (which includes Monitoring) or just Monitoring alone.

Automation lets you deploy and configure MongoDB processes from Ops Manager as well as monitor and manage them. If you select Automation, Ops Manager prompts you to download the *Automation Agent* and *Monitoring Agent*.

Monitoring lets you manage a MongoDB deployment from Ops Manager. If you select Monitoring, Ops Manager prompts you to download only the *Monitoring Agent*.

Install the Backup Daemon (Optional)

If you use *Backup*, install the *Backup Daemon*.

Step 1: Download the Backup Daemon package.

1. In a browser, go to http://www.mongodb.com/download.
2. Submit the subscription form.
3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation to contact MongoDB for production installs.
5. On the *MongoDB Ops Manager Downloads* page, copy the link address of the "Backup" *DEB* link.
6. Open a system prompt.
7. Download the Backup Daemon package by issuing a `curl` command that uses the copied link address:

   ```
   curl -OL <link-address-for-backup-deb-package>
   ```

   The downloaded package is named `mongodb-mms-backup-daemon-<version>.x86_64.deb`, where `<version>` is replaced by the version number.

Step 2: Install the Backup Daemon package.

Issue `dpkg --install` with root privileges and specify the name of the downloaded package:

```
sudo dpkg --install <downloaded-package>
```

When installed, the base directory for the Ops Manager software is `/opt/mongodb/mms/`. The `.deb` package creates a new system user `mongodb-mms` under which the server will run.

Step 3: Point the Backup Daemon to the Ops Manager Application database.

Open the `/opt/mongodb/mms-backup-daemon/conf/conf-daemon.properties` file with root privileges and set the *`mongo.mongoUri`* value to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

Step 4: Synchronize the `gen.key` file

Synchronize the `/etc/mongodb-mms/gen.key` file from an Ops Manager Application server. This is only required if the Backup Daemon was installed on a different server than the Ops Manager Application.

Step 5: Start the back-end software package.

To start the Backup Daemon run:

```
sudo service mongodb-mms-backup-daemon start
```

If everything worked the following displays:

```
Start Backup Daemon                                        [  OK  ]
```

If you run into any problems, the log files are at `/opt/mongodb/mms-backup-daemon/logs`.

Step 6: Open Ops Manager and access the Backup configuration page.

Open the Ops Manager home page and log in as the user you first registered when installing the Ops Manager Application. (This user is the *global owner*.) Then click the *Admin* link at the top right of the page. Then click the *Backup* tab.

Step 7: Enter configuration information for the Backup database.

Enter the configuration information described below, and then click *Save*. Ops Manager uses this information to create the connection string URI used to connect to the database.

<hostname>:<port>: Enter a comma-separated list of the fully qualified domain names and port numbers for all replica set members for the Backup database.

MongoDD Auth Username and *MongoDB Auth Password*: Enter the user credentials if the database uses authentication.

Encrypted Credentials: Check this if the user credentials use the Ops Manager `credentialstool`. For more information, see *Encrypt MongoDB User Credentials*.

Use SSL: Check this if the MongoDB database uses SSL. If you select this, you must configure SSL settings for both the Ops Manager Application and Backup Daemon. See *Ops Manager Configuration Files*.

Connection Options: To add additional connection options, enter them using the MongoDB Connection String URI Format.

Install Ops Manager with `rpm` Packages

On this page

Overview

To install Ops Manager you install the *Ops Manager Application* and the optional *Backup Daemon*. This tutorial describes how to install both using `rpm` packages. The Ops Manager Application monitors MongoDB deployments, and the Backup Daemon creates and stores deployment snapshots.

If you are instead upgrading an existing deployment, please see *Upgrade Ops Manager*.

Warning: To use Ops Manager 1.6 Automation to manage MongoDB Enterprise deployments, you must **first install** the *MongoDB Enterprise dependencies* to each server that runs MongoDB Enterprise. You must install the dependencies to the servers before using Automation.

Note that Automation **does not yet support** use of the MongoDB Enterprise advanced security features (SSL, LDAP, Kerberos, and auditing). Automation will support these features in the next major Ops Manager release.

To run MongoDB Enterprise with advanced security now, deploy MongoDB Enterprise manually (not through Automation) and use Ops Manager only for Monitoring and Backup.

Prerequisites

Deploy Servers

Prior to installation, you must set up servers for the entire Ops Manager deployment, including the Ops Manager Application, the optional Backup Daemon, and the *backing replica sets*. For deployment diagrams, see *Example Installation Diagrams*.

Deploy servers that meet the hardware requirements described in *Ops Manager Hardware and Software Requirements*. Servers for the Backup Daemon and the backing replica sets must also comply with the Production Notes in the MongoDB manual. Configure as many servers as needed for your deployment.

Warning: Failure to configure servers according to the MongoDB Production Notes can lead to production failure.

Deploy MongoDB

Install MongoDB on the servers that will store the *Ops Manager Application Database* and *Backup Blockstore Database*. The Backup Blockstore database is required only if you run the Backup Daemon. The databases require dedicated MongoDB instances. Do **not** use MongoDB installations that store other data.

Install separate MongoDB instances for the two databases and install the instances as replica sets. Ensure that firewall rules on the servers allow access to the *ports* that the instances runs on.

The Ops Manager Application and Backup Daemon must authenticate to their databases as a MongoDB user with appropriate access. The user must have the following roles:

- `readWriteAnyDatabase`
- `dbAdminAnyDatabase`.
- `clusterAdmin` if the database is a sharded cluster, otherwise `clusterMonitor`

Install Procedures

You must have administrative access on the machines to which you install.

Install and Start the Ops Manager Application

Step 1: Download the latest version of the Ops Manager Application package.

1. In a browser, go to http://www.mongodb.com/download.
2. Submit the subscription form.
3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation for production installs.
5. On the *MongoDB Ops Manager Downloads* page, copy the link address of the "Monitoring, Automation and Core" *RPM* link.
6. Open a system prompt.
7. Download the Ops Manager Application package by issuing a `curl` command that uses the copied link address:

   ```
   curl -OL <link-address-for-monitoring-automation-core-rpm>
   ```

   The downloaded package is named `mongodb-mms-<version>.x86_64.rpm`, where `<version>` is the version number.

Step 2: Install the Ops Manager Application package.

Install the `.rpm` package by issuing the following command, where `<version>` is the version of the `.rpm` package:

```
sudo rpm -ivh mongodb-mms-<version>.x86_64.rpm
```

When installed, the base directory for the Ops Manager software is `/opt/mongodb/mms/`. The RPM package creates a new system user `mongodb-mms` under which the server runs.

Step 3: Configure Monitoring.

Open `<install-directory>/conf/conf-mms.properties` with root privileges and set values for the settings described in this step. For detailed information on each setting, see the *Ops Manager Configuration Files* page.

Set *mms.centralUrl* and *mms.backupCentralUrl* as follows, where `<host>` is the fully qualified domain name of the server running the Ops Manager Application.

```
mms.centralUrl=http://<host>:8080
mms.backupCentralUrl=http://<host>:8081
```

Set the following *Email Address Settings* as appropriate. Each may be the same or different.

```
mms.fromEmailAddr=<email_address>
mms.replyToEmailAddr=<email_address>
mms.adminFromEmailAddr=<email_address>
mms.adminEmailAddr=<email_address>
mms.bounceEmailAddr=<email_address>
```

Set *mongo.mongoUri* to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

If you use HTTPS to encrypt user connections to Ops Manager, set *mms.https.PEMKeyFile* to a PEM file containing an X509 certificate and private key, and set *mms.https.PEMKeyFilePassword* to the password for the certificate. For example:

```
mms.https.PEMKeyFile=<path_and_name_of_pem_file>
mms.https.PEMKeyFilePassword=<password_for_pem_file>
```

To configure authentication, email, and other optional settings, see *Ops Manager Configuration Files*. To run the Ops Manager application in a highly available configuration, see *Configure a Highly Available Ops Manager Application*.

Step 4: Start the Ops Manager Application.

Issue the following command:

```
sudo service mongodb-mms start
```

Step 5: Open the Ops Manager home page and register the first user.

To open the home page, enter the following URL in a browser, where <host> is the fully qualified domain name of the server:

```
http://<host>:8080
```

Click the *Register* link and follow the prompts to register the first user and create the first group. The first user is automatically assigned the *Global Owner* role. When you finish, you are logged into the Ops Manager Application as the new user. For more information on creating and managing users, see *Manage Ops Manager Users*.

Step 6: At the Welcome page, follow the prompts to set up Automation (which includes Monitoring) or just Monitoring alone.

Automation lets you deploy and configure MongoDB processes from Ops Manager as well as monitor and manage them. If you select Automation, Ops Manager prompts you to download the *Automation Agent* and *Monitoring Agent*.

Monitoring lets you manage a MongoDB deployment from Ops Manager. If you select Monitoring, Ops Manager prompts you to download only the *Monitoring Agent*.

Install the Backup Daemon (Optional)

If you use *Backup*, install the *Backup Daemon*.

Step 1: Download the Backup Daemon package.

To download the Backup Daemon package:

1. In a browser, go to http://www.mongodb.com/download.
2. Submit the subscription form.

3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation to contact MongoDB for production installs.
5. On the *MongoDB Ops Manager Downloads* page, copy the link address of the "Backup" *RPM* link.
6. Open a system prompt.
7. Download the Backup Daemon package by issuing a `curl` command that uses the copied link address:

   ```
   curl -OL <link-address-for-backup-rpm>
   ```

   The downloaded package is named `mongodb-mms-backup-daemon-<version>.x86_64.rpm`, where `<version>` is replaced by the version number.

Step 2: Install the Backup Daemon package.

Issue `rpm --install` with root privileges and specify the name of the downloaded package:

```
sudo rpm --install <downloaded-package>
```

The software is installed to `/opt/mongodb/mms-backup-daemon`.

Step 3: Point the Backup Daemon to the Ops Manager Application database.

Open the `/opt/mongodb/mms-backup-daemon/conf/conf-daemon.properties` file with root privileges and set the *`mongo.mongoUri`* value to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

Step 4: Synchronize the `gen.key` file

Synchronize the `/etc/mongodb-mms/gen.key` file from an Ops Manager Application server. This is only required if the Backup Daemon was installed on a different server than the Ops Manager Application.

Step 5: Start the back-end software package.

To start the Backup Daemon run:

```
sudo service mongodb-mms-backup-daemon start
```

If everything worked the following displays:

```
Start Backup Daemon                                        [  OK  ]
```

If you run into any problems, the log files are at `/opt/mongodb/mms-backup-daemon/logs`.

Step 6: Open Ops Manager and access the Backup configuration page.

Open the Ops Manager home page and log in as the user you first registered when installing the Ops Manager Application. (This user is the *global owner*.) Then click the *Admin* link at the top right of the page. Then click the *Backup* tab.

Step 7: Enter configuration information for the Backup database.

Enter the configuration information described below, and then click *Save*. Ops Manager uses this information to create the connection string URI used to connect to the database.

<hostname>:<port>: Enter a comma-separated list of the fully qualified domain names and port numbers for all replica set members for the Backup database.

MongoDD Auth Username and *MongoDB Auth Password*: Enter the user credentials if the database uses authentication.

Encrypted Credentials: Check this if the user credentials use the Ops Manager `credentialstool`. For more information, see *Encrypt MongoDB User Credentials*.

Use SSL: Check this if the MongoDB database uses SSL. If you select this, you must configure SSL settings for both the Ops Manager Application and Backup Daemon. See *Ops Manager Configuration Files*.

Connection Options: To add additional connection options, enter them using the MongoDB Connection String URI Format.

Install Ops Manager from `tar.gz` or `zip` Archives

On this page

Overview

To install Ops Manager you install the *Ops Manager Application* and the optional *Backup Daemon*. This tutorial describes how to install both using `tar.gz` or `zip` packages. The tutorial installs to a Linux OS. The Ops Manager Application monitors MongoDB deployments, and the Backup Daemon creates and stores deployment snapshots.

If you are instead upgrading an existing deployment, please see *Upgrade Ops Manager*.

Warning: To use Ops Manager 1.6 Automation to manage MongoDB Enterprise deployments, you must **first install** the *MongoDB Enterprise dependencies* to each server that runs MongoDB Enterprise. You must install the dependencies to the servers before using Automation.

Note that Automation **does not yet support** use of the MongoDB Enterprise advanced security features (SSL, LDAP, Kerberos, and auditing). Automation will support these features in the next major Ops Manager release.

To run MongoDB Enterprise with advanced security now, deploy MongoDB Enterprise manually (not through Automation) and use Ops Manager only for Monitoring and Backup.

Prerequisites

Deploy Servers

Prior to installation, you must set up servers for the entire Ops Manager deployment, including the Ops Manager Application, the optional Backup Daemon, and the *backing replica sets*. For deployment diagrams, see *Example Installation Diagrams*.

Deploy servers that meet the hardware requirements described in *Ops Manager Hardware and Software Requirements*. Servers for the Backup Daemon and the backing replica sets must also comply with the Production Notes in the MongoDB manual. Configure as many servers as needed for your deployment.

> **Warning:** Failure to configure servers according to the MongoDB Production Notes can lead to production failure.

Deploy MongoDB

Install MongoDB on the servers that will store the *Ops Manager Application Database* and *Backup Blockstore Database*. The Backup Blockstore database is required only if you run the Backup Daemon. The databases require dedicated MongoDB instances. Do **not** use MongoDB installations that store other data.

Install separate MongoDB instances for the two databases and install the instances as replica sets. Ensure that firewall rules on the servers allow access to the *ports* that the instances runs on.

The Ops Manager Application and Backup Daemon must authenticate to their databases as a MongoDB user with appropriate access. The user must have the following roles:

- `readWriteAnyDatabase`
- `dbAdminAnyDatabase`.
- `clusterAdmin` if the database is a sharded cluster, otherwise `clusterMonitor`

Install Procedures

You must have administrative access on the machines to which you install.

Install and Start the Ops Manager Application

Step 1: Download the Ops Manager Application package.

1. In a browser, go to http://www.mongodb.com/download.
2. Submit the subscription form.
3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation to contact MongoDB for production installs.
5. On the *MongoDB Ops Manager Downloads* page, copy the link address of the "Monitoring, Automation and Core" *TAR.GZ* link.

6. Open a system prompt.
7. Download the Ops Manager Application package by issuing a `curl` command that uses the copied link address:

   ```
   curl -OL <link-address-for-monitoring-automation-core-tar.gz>
   ```

   The downloaded package is named `mongodb-mms-<version>.x86_64.tar.gz`, where `<version>` is the version number.

Step 2: Extract the Ops Manager Application package.

Navigate to the directory to which to install the Ops Manager Application. Extract the archive to that directory:

```
tar -zxf mongodb-mms-<version>.x86_64.tar.gz
```

When complete, Ops Manager is installed.

Step 3: Configure Monitoring.

Open `<install-directory>/conf/conf-mms.properties` with root privileges and set values for the settings described in this step. For detailed information on each setting, see the *Ops Manager Configuration Files* page.

Set *`mms.centralUrl`* and *`mms.backupCentralUrl`* as follows, where `<host>` is the fully qualified domain name of the server running the Ops Manager Application.

```
mms.centralUrl=http://<host>:8080
mms.backupCentralUrl=http://<host>:8081
```

Set the following *Email Address Settings* as appropriate. Each may be the same or different.

```
mms.fromEmailAddr=<email_address>
mms.replyToEmailAddr=<email_address>
mms.adminFromEmailAddr=<email_address>
mms.adminEmailAddr=<email_address>
mms.bounceEmailAddr=<email_address>
```

Set *`mongo.mongoUri`* to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

If you use HTTPS to encrypt user connections to Ops Manager, set *`mms.https.PEMKeyFile`* to a PEM file containing an X509 certificate and private key, and set *`mms.https.PEMKeyFilePassword`* to the password for the certificate. For example:

```
mms.https.PEMKeyFile=<path_and_name_of_pem_file>
mms.https.PEMKeyFilePassword=<password_for_pem_file>
```

To configure authentication, email, and other optional settings, see *Ops Manager Configuration Files*. To run the Ops Manager application in a highly available configuration, see *Configure a Highly Available Ops Manager Application*.

Step 4: Start the Ops Manager Application.

To start Ops Manager, issue the following command:

```
<install_dir>/bin/mongodb-mms start
```

Step 5: Open the Ops Manager home page and register the first user.

To open the home page, enter the following URL in a browser, where `<host>` is the fully qualified domain name of the server:

```
http://<host>:8080
```

Click the *Register* link and follow the prompts to register the first user and create the first group. The first user is automatically assigned the *Global Owner* role. When you finish, you are logged into the Ops Manager Application as the new user. For more information on creating and managing users, see *Manage Ops Manager Users*.

Step 6: At the Welcome page, follow the prompts to set up Automation (which includes Monitoring) or just Monitoring alone.

Automation lets you deploy and configure MongoDB processes from Ops Manager as well as monitor and manage them. If you select Automation, Ops Manager prompts you to download the *Automation Agent* and *Monitoring Agent*.

Monitoring lets you manage a MongoDB deployment from Ops Manager. If you select Monitoring, Ops Manager prompts you to download only the *Monitoring Agent*.

Install the Backup Daemon (Optional)

If you use *Backup*, install the *Backup Daemon*.

Step 1: Download the Backup Daemon package.

1. In a browser, go to http://www.mongodb.com/download.
2. Submit the subscription form.
3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation to contact MongoDB for production installs.
5. On the *MongoDB Ops Manager Downloads* page, copy the link address of the "Backup" *TAR.GZ* link.
6. Open a system prompt.
7. Download the Backup Daemon package by issuing a `curl` command that uses the copied link address:

   ```
   curl -OL <link-address-for-backup-tar.gz>
   ```

   The downloaded package is named `mongodb-mms-backup-daemon-<version>.x86_64.tar.gz`, where `<version>` is replaced by the version number.

Step 2: To install the Backup Daemon, extract the downloaded archive file.

```
tar -zxf <downloaded-archive-file>
```

Step 3: Point the Backup Daemon to the Ops Manager Application database.

Open the `<install-dir>/conf/conf-daemon.properties` file and set the *`mongo.mongoUri`* value to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

Additionally, ensure that the file system that holds the *`rootDirectory`* has sufficient space to accommodate the current snapshots of all backed up instances.

Step 4: Synchronize the `mms-gen-key` file.

Synchronize the `<install-dir>/bin/mms-gen-key` file from the Ops Manager Application server. This is required only if the Backup Daemon is installed on a different server than the Ops Manager Application.

Step 5: Start the Backup Daemon.

To start the Backup Daemon run:

```
<install_dir>/bin/mongodb-mms-backup-daemon start
```

If you run into any problems, the log files are at `<install_dir>/logs`.

Step 6: Open Ops Manager and access the Backup configuration page.

Open the Ops Manager home page and log in as the user you first registered when installing the Ops Manager Application. (This user is the *global owner*.) Then click the *Admin* link at the top right of the page. Then click the *Backup* tab.

Step 7: Enter configuration information for the Backup database.

Enter the configuration information described below, and then click *Save*. Ops Manager uses this information to create the connection string URI used to connect to the database.

<hostname>:<port>: Enter a comma-separated list of the fully qualified domain names and port numbers for all replica set members for the Backup database.

MongoDD Auth Username and *MongoDB Auth Password*: Enter the user credentials if the database uses authentication.

Encrypted Credentials: Check this if the user credentials use the Ops Manager `credentialstool`. For more information, see *Encrypt MongoDB User Credentials*.

Use SSL: Check this if the MongoDB database uses SSL. If you select this, you must configure SSL settings for both the Ops Manager Application and Backup Daemon. See *Ops Manager Configuration Files*.

Connection Options: To add additional connection options, enter them using the MongoDB Connection String URI Format.

Install Ops Manager on Windows

On this page

Overview

This tutorial describes how to install the *Ops Manager Application*, which monitors MongoDB deployments, and the optional *Backup Daemon*, which creates and stores deployment snapshots. This tutorial installs to Windows servers.

Ops Manager supports *Monitoring* and *Backup* on Windows but does not support Automation on Windows.

Prerequisites

Prior to installation you must:

- Configure Windows servers that meet the *hardware and software requirements*. Configure as many servers as needed for your deployment. For deployment diagrams, see *Example Installation Diagrams*.
- Deploy the dedicated MongoDB instances that store the *Ops Manager Application Database* and *Backup Blockstore Database*. Do not use MongoDB instances that store other data. Ensure that firewall rules allow access to the *ports* the instances runs on. See *Deploy Backing MongoDB Replica Sets*.
- Optionally install an SMTP email server.

Procedures

You must have administrative access on the machines to which you install.

Install and Start the Ops Manager Application

Step 1: Download Monitoring.

1. In a browser, go to http://www.mongodb.com/download.
2. Submit the subscription form.
3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation to contact MongoDB for production installs.

5. On the *MongoDB Ops Manager Downloads* page, click the "Monitoring and Core" *MSI* link.

Step 2: Install the Ops Manager Application.

Right-click on the `mongodb-mms-<version>.msi` file and select *Install*. Follow the instructions in the Setup Wizard.

During setup, the *Configuration/Log Folder* screen prompts you to specify a folder for configuration and log files. The installation restricts access to the folder to administrators only.

Step 3: Configure the Ops Manager Application.

In the folder you selected for configuration and log files, navigate to `\Server\Config`. For example, if you chose `C:\MMSData` for configuration and log files, navigate to `C:\MMSData\Server\Config`.

Open the `conf-mms.properties` file and configure the required settings below, as well as any additional settings your deployment uses, such as authentication settings. For descriptions of all settings, see *Ops Manager Configuration Files*.

Set *mms.centralUrl* and *mms.backupCentralUrl* as follows, where `<host>` is the fully qualified domain name of the server running the Ops Manager Application.

```
mms.centralUrl=http://<host>:8080
mms.backupCentralUrl=http://<host>:8081
```

Set the following *Email Address Settings* as appropriate. Each can be the same or different values.

```
mms.fromEmailAddr=<email_address>
mms.replyToEmailAddr=<email_address>
mms.adminFromEmailAddr=<email_address>
mms.adminEmailAddr=<email_address>
mms.bounceEmailAddr=<email_address>
```

Set the *mongo.mongoUri* option to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

Step 4: Start the `MongoDB Ops Manager HTTP Service`.

Before starting the service, make sure the MongoDB instances that store the *Ops Manager Application Database* are running and that they are reachable from the Ops Manager Application's host machine. Ensure that firewall rules allow access to the ports the MongoDB instances runs on.

To start the service, open `Control Panel`, then `System and Security`, then `Administrative Tools`, and then `Services`.

In the `Services` list, right-click on the `MongoDB Ops Manager HTTP Service` and select `Start`.

Step 5: If you will also run MMS Backup, start the two Backup services.

In the `Services` list, right-click on the following services and select `Start`:

- `MMS Backup HTTP Service`
- `MMS Backup Alert Service`

Step 6: Open the Ops Manager home page and register the first user.

To open the home page, enter the following URL in a browser, where `<host>` is the fully qualified domain name of the server:

```
http://<host>:8080
```

Click the *Register* link and follow the prompts to register the first user and create the first group. The first user is automatically assigned the *Global Owner* role. When you finish, you are logged into the Ops Manager Application as the new user. For more information on creating and managing users, see *Manage Ops Manager Users*.

Step 7: At the Welcome page, follow the prompts to set up Monitoring.

Ops Manager prompts you to download only the *Monitoring Agent*.

Install the Backup Daemon (Optional)

If you use Backup, install the *Backup Daemon*.

Step 1: Download the Backup Daemon package.

1. In a browser, go to http://www.mongodb.com/download.
2. Submit the subscription form.
3. On the *MongoDB Enterprise Downloads* page, go to the *MongoDB Ops Manager* section and click the *here* link.
4. On the *Ops Manager Download* page, acknowledge the recommendation to contact MongoDB for production installs.
5. On the *MongoDB Ops Manager Downloads* page, copy the link address of the "Backup" *MSI* link.

Step 2: Install the Backup Daemon.

Right-click on the `mongodb-mms-backup-daemon-<version>.msi` file and select *Install*. Follow the instructions in the Setup Wizard.

During setup, the *Daemon Paths* screen prompts you to specify the following folders. The installer will restrict access to these folders to administrators only:

- *Configuration/Log Path*. The location of the Backup Daemon's configuration and log files.
- *Backup Data Root Path*. The path where the Backup Daemon stores the local copies of the backed-up databases. This location must have enough storage to hold a full copy of each database being backed up.
- *MongoDB Releases Path*. The location of the MongoDB software releases required to replicate the backed up databases. These releases will be downloaded from `mongodb.org` by default.

Step 3: Configure the Backup Daemon.

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

In the folder you selected for storing configuration and log files, navigate to `\BackupDaemon\Config`. For example, if you chose `C:\MMSData`, navigate to `C:\MMSData\BackupDaemon\Config`.

Open the `conf-daemon.properties` file and configure the *mongo.mongoUri* property to point the Backup Daemon to the servers and ports hosting the Ops Manager Application database. For example:

Step 4: Copy the `gen-key` file from the Ops Manager Application server to the Backup Daemon server.

Important: You must copy the file as a whole. Do not open the file and copy its content.

Copy the `gen.key` file from the `C:\MMSData\Secrets` folder on the Ops Manager Application server to the empty `C:\MMSData\Secrets` folder on the Backup Daemon server.

Step 5: If you have not already done so, start the Backup services on the Ops Manager Application server.

On the Ops Manager Application server, open `Control Panel`, then `System and Security`, then `Administrative Tools`, and then `Services`. Right-click on the following services and select `Start`:

- `MMS Backup HTTP Service`
- `MMS Backup Alert Service`

Step 6: Start the Backup Daemon.

On the **Backup Daemon server**, open `Control Panel`, then `System and Security`, then `Administrative Tools`, and then `Services`. Right-click on the `MMS Backup Daemon Service` and select `Start`.

Step 7: Open Ops Manager and access the Backup configuration page.

Open the Ops Manager home page and log in as the user you registered when installing the Ops Manager Application. Then click the *Admin* link at the top right of the page. Then click the *Backup* tab.

Step 8: Enter configuration information for the Backup database.

Enter the configuration information described here, and then click *Save*. Ops Manager uses this information to create the connection string URI used to connect to the database.

<hostname>:<port>: Enter a comma-separated list of the fully qualified domain names and port numbers for all replica set members for the Backup database. For test deployments, you can use a standalone MongoDB instance for the database.

MongoDD Auth Username and *MongoDB Auth Password*: Enter the user credentials if the database uses authentication.

Encrypted Credentials: Check this if the user credentials use the Ops Manager `credentialstool`. For more information, see *Encrypt MongoDB User Credentials*.

Use SSL: Check this if the MongoDB database uses SSL. If you select this, you must configure SSL settings for both the Ops Manager Application and Backup Daemon. See *Ops Manager Configuration Files*.

Connection Options: To add additional connection options, enter them using the MongoDB Connection String URI Format.

Next Step

Set up security for your Ops Manager servers, Ops Manager agents, and MongoDB deployments.

Note: To set up a deployment for a test environment, see *Test Ops Manager Monitoring*. The tutorial populates the replica set with test data, registers a user, and installs the Monitoring and Backup Agents on a client machine in order to monitor the test replica set.

2.6 Upgrade Ops Manager

Upgrade with DEB Packages Upgrade Ops Manager on Debian and Ubuntu systems.

Upgrade with RPM Packages Upgrade Ops Manager on Red Hat, Fedora, CentOS, and Amazon AMI Linux.

Upgrade from Archives on Linux Upgrade Ops Manager on other Linux systems, without using package management.

Upgrade from Version 1.2 and Earlier Upgrade from a version before 1.3.

Upgrade Ops Manager with `deb` Packages

On this page

Warning: To use Ops Manager 1.6 Automation to manage MongoDB Enterprise deployments, you must **first install** the *MongoDB Enterprise dependencies* to each server that runs MongoDB Enterprise. You must install the dependencies to the servers before using Automation.

Note that Automation **does not yet support** use of the MongoDB Enterprise advanced security features (SSL, LDAP, Kerberos, and auditing). Automation will support these features in the next major Ops Manager release.

To run MongoDB Enterprise with advanced security now, deploy MongoDB Enterprise manually (not through Automation) and use Ops Manager only for Monitoring and Backup.

Overview

This tutorial describes how to upgrade an existing *Ops Manager Application* and *Backup Daemon* using `deb` packages.

Prerequisite

You must have administrative access on the machines on which you perform the upgrade.

You must have the download link available on the customer downloads page provided to you by MongoDB. If you do not have this link, you can access the download page for evaluation at http://www.mongodb.com/download.

Procedures

If your version is earlier than 1.3, please see instead *Upgrade from Version 1.2 and Earlier*.

Upgrade the Ops Manager Application from Version 1.3 and Later

If you have an existing Ops Manager Application, use the following procedure to upgrade to the latest release. There are no supported downgrade paths for Ops Manager.

Step 1: *Recommended*. Take a full backup of the Ops Manager database before beginning the upgrade procedure.

Step 2: Shut down Ops Manager.

For example:

```
sudo service mongodb-mms stop
```

Step 3: If you are running Ops Manager Backup, shutdown the Ops Manager Backup Daemon.

The daemon may be installed on a different server. It is *critical* that this is also shut down. To shut down, issue a command similar to the following:

```
sudo service mongodb-mms-backup-daemon stop
```

Step 4: Save a copy of your previous configuration file.

For example:

```
sudo cp /opt/mongodb/mms/conf/conf-mms.properties ~/.
```

Step 5: Upgrade the package.

For example:

```
sudo dpkg -i mongodb-mms_<version>_x86_64.deb
```

Step 6: Edit the new configuration file.

Fill in the new configuration file at `/opt/mongodb/mms/conf/conf-mms.properties` using your old file as a reference point.

Step 7: Start Ops Manager.

For example:

```
sudo service mongodb-mms start
```

Step 8: Update all Monitoring Agents.

See *Install Monitoring Agent* for more information.

Step 9: Update the Backup Daemon and any Backup Agent, as appropriate.

If you are running Backup, update the Backup Daemon package and any Backup Agent.

See *Install Backup Agent* for more information.

Upgrade the Backup Daemon

Step 1: Stop the currently running instance.

```
sudo service mongodb-mms-backup-daemon stop
```

Step 2: Download the latest version of the Backup Daemon.

Download the new version of the Backup Daemon package by issuing a `curl` command with the download link available on the customer downloads page provided to you by MongoDB.

```
curl -OL <link-address-for-backup-deb-package>
```

Step 3: Point the Backup Daemon to the Ops Manager Application database.

Open the `/opt/mongodb/mms-backup-daemon/conf/conf-daemon.properties` file with root privileges and set the *`mongo.mongoUri`* value to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

Step 4: Synchronize the `gen.key` file

Synchronize the `/etc/mongodb-mms/gen.key` file from an Ops Manager Application server. This is only required if the Backup Daemon was installed on a different server than the Ops Manager Application.

Step 5: Start the back-end software package.

To start the Backup Daemon run:

```
sudo service mongodb-mms-backup-daemon start
```

If everything worked the following displays:

```
Start Backup Daemon                                        [  OK  ]
```

If you run into any problems, the log files are at `/opt/mongodb/mms-backup-daemon/logs`.

Step 6: Open Ops Manager and access the Backup configuration page.

Open the Ops Manager home page and log in as the user you first registered when installing the Ops Manager Application. (This user is the *global owner*.) Then click the *Admin* link at the top right of the page. Then click the *Backup* tab.

Step 7: Enter configuration information for the Backup database.

Enter the configuration information described below, and then click *Save*. Ops Manager uses this information to create the connection string URI used to connect to the database.

<hostname>:<port>: Enter a comma-separated list of the fully qualified domain names and port numbers for all replica set members for the Backup database.

MongoDD Auth Username and *MongoDB Auth Password*: Enter the user credentials if the database uses authentication.

Encrypted Credentials: Check this if the user credentials use the Ops Manager `credentialstool`. For more information, see *Encrypt MongoDB User Credentials*.

Use SSL: Check this if the MongoDB database uses SSL. If you select this, you must configure SSL settings for both the Ops Manager Application and Backup Daemon. See *Ops Manager Configuration Files*.

Connection Options: To add additional connection options, enter them using the MongoDB Connection String URI Format.

Upgrade Ops Manager with `rpm` Packages

On this page

Warning: To use Ops Manager 1.6 Automation to manage MongoDB Enterprise deployments, you must **first install** the *MongoDB Enterprise dependencies* to each server that runs MongoDB Enterprise. You must install the dependencies to the servers before using Automation.

Note that Automation **does not yet support** use of the MongoDB Enterprise advanced security features (SSL, LDAP, Kerberos, and auditing). Automation will support these features in the next major Ops Manager release.

To run MongoDB Enterprise with advanced security now, deploy MongoDB Enterprise manually (not through Automation) and use Ops Manager only for Monitoring and Backup.

Overview

This tutorial describes how to upgrade an existing *Ops Manager Application* and *Backup Daemon* using `rpm` packages.

Prerequisite

You must have administrative access on the machines on which you perform the upgrade.

You must have the download link available on the customer downloads page provided to you by MongoDB. If you do not have this link, you can access the download page for evaluation at http://www.mongodb.com/download.

Procedures

If your version is earlier than 1.3, please see instead *Upgrade from Version 1.2 and Earlier*.

Upgrade the Ops Manager Application from Version 1.3 and Later

If you have an existing Ops Manager Application, use the following procedure to upgrade to the latest release. There are no supported downgrade paths for Ops Manager.

Step 1: *Recommended*. Take a full backup of the Ops Manager database before beginning the upgrade procedure.

Step 2: Shut down Ops Manager.

For example:

```
sudo service mongodb-mms stop
```

Step 3: If you are running Ops Manager Backup, shutdown the Ops Manager Backup Daemon.

The daemon may be installed on a different server. It is *critical* that this is also shut down. To shut down, issue a command similar to the following:

```
sudo service mongodb-mms-backup-daemon stop
```

Step 4: Save a copy of your previous configuration file.

For example:

```
sudo cp /opt/mongodb/mms/conf/conf-mms.properties ~/.
```

Step 5: Upgrade the package.

For example:

```
sudo rpm -U mongodb-mms-<version>.x86_64.rpm
```

Step 6: Move the new version of the configuration file into place.

Move the `conf-mms.properties` configuration file to the following location:

```
/opt/mongodb/mms/conf/conf-mms.properties
```

Step 7: Edit the new configuration file.

Fill in the new configuration file at `/opt/mongodb/mms/conf/conf-mms.properties` using your old file as a reference point.

Step 8: Start Ops Manager.

For example:

```
sudo service mongodb-mms start
```

Step 9: Update all Monitoring Agents.

See *Install Monitoring Agent* for more information.

Step 10: Update the Backup Daemon and any Backup Agent, as appropriate.

If you are running Backup, update the Backup Daemon package and any Backup Agent.

See *Install Backup Agent* for more information.

Upgrade the Backup Daemon

Step 1: Stop the currently running instance.

```
sudo service mongodb-mms-backup-daemon stop
```

Step 2: Download the latest version of the Backup Daemon.

Download the new version of the Backup Daemon package by issuing a `curl` command with the download link available on the customer downloads page provided to you by MongoDB.

```
curl -OL <link-address-for-backup-rpm>
```

Step 3: Point the Backup Daemon to the Ops Manager Application database.

Open the `/opt/mongodb/mms-backup-daemon/conf/conf-daemon.properties` file with root privileges and set the *mongo.mongoUri* value to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

Step 4: Synchronize the `gen.key` file

Synchronize the `/etc/mongodb-mms/gen.key` file from an Ops Manager Application server. This is only required if the Backup Daemon was installed on a different server than the Ops Manager Application.

Step 5: Start the back-end software package.

To start the Backup Daemon run:

```
sudo service mongodb-mms-backup-daemon start
```

If everything worked the following displays:

```
Start Backup Daemon                                        [  OK  ]
```

If you run into any problems, the log files are at `/opt/mongodb/mms-backup-daemon/logs`.

Step 6: Open Ops Manager and access the Backup configuration page.

Open the Ops Manager home page and log in as the user you first registered when installing the Ops Manager Application. (This user is the *global owner*.) Then click the *Admin* link at the top right of the page. Then click the *Backup* tab.

Step 7: Enter configuration information for the Backup database.

Enter the configuration information described below, and then click *Save*. Ops Manager uses this information to create the connection string URI used to connect to the database.

<hostname>:<port>: Enter a comma-separated list of the fully qualified domain names and port numbers for all replica set members for the Backup database.

MongoDD Auth Username and *MongoDB Auth Password*: Enter the user credentials if the database uses authentication.

Encrypted Credentials: Check this if the user credentials use the Ops Manager `credentialstool`. For more information, see *Encrypt MongoDB User Credentials*.

Use SSL: Check this if the MongoDB database uses SSL. If you select this, you must configure SSL settings for both the Ops Manager Application and Backup Daemon. See *Ops Manager Configuration Files*.

Connection Options: To add additional connection options, enter them using the MongoDB Connection String URI Format.

Upgrade Ops Manager from `tar.gz` or `zip` Archives

On this page

Warning: To use Ops Manager 1.6 Automation to manage MongoDB Enterprise deployments, you must **first install** the *MongoDB Enterprise dependencies* to each server that runs MongoDB Enterprise. You must install the dependencies to the servers before using Automation.

Note that Automation **does not yet support** use of the MongoDB Enterprise advanced security features (SSL, LDAP, Kerberos, and auditing). Automation will support these features in the next major Ops Manager release.

To run MongoDB Enterprise with advanced security now, deploy MongoDB Enterprise manually (not through Automation) and use Ops Manager only for Monitoring and Backup.

Overview

This tutorial describes how to upgrade an existing *Ops Manager Application* and *Backup Daemon* using tar.gz or zip files.

Prerequisite

You must have administrative access on the machines on which you perform the upgrade.

You must have the download link available on the customer downloads page provided to you by MongoDB. If you do not have this link, you can access the download page for evaluation at http://www.mongodb.com/download.

Procedures

If your version is earlier than 1.3, please see instead *Upgrade from Version 1.2 and Earlier*.

Upgrade the Ops Manager Application from Version 1.3 and Later

If you have an existing Ops Manager Application, use the following procedure to upgrade to the latest release. There are no supported downgrade paths for Ops Manager.

To upgrade a tarball installation, backup the configuration file and logs, and then re-install the Ops Manager server.

Important: It is crucial that you back up the existing configuration because the upgrade process will delete existing data.

In more detail:

Step 1: Shutdown the Ops Manager server and take a backup of your existing configuration and logs.

For example:

```
<install_dir>/bin/mongodb-mms stop
cp -a <install_dir>/conf ~/mms_conf.backup
cp -a <install_dir>/logs ~/mms_logs.backup
```

Step 2: If you are running Ops Manager Backup, shutdown the Ops Manager Backup Daemon.

The daemon may be installed on a different server. It is *critical* that this is also shut down. To shut down, issue a command similar to the following:

```
<install_dir>/bin/mongodb-mms-backup-daemon stop
```

Step 3: Remove your existing Ops Manager server installation entirely and extract the latest release in its place.

For example:

```
cd <install_dir>/../
rm -rf <install_dir>
tar -zxf -C . /path/to/mongodb-mms-<version>.x86_64.tar.gz
```

Step 4: Compare and reconcile any changes in configuration between versions.

For example:

```
diff -u ~/mms_conf.backup/conf-mms.properties <install_dir>/conf/conf-mms.properties
diff -u ~/mms_conf.backup/mms.conf <install_dir>/conf/mms.conf
```

Step 5: Edit your configuration to resolve any conflicts between the old and new versions.

Make *configuration* changes as appropriate. Changes to `mms.centralUri`, email addresses, and MongoDB are the most common configuration changes.

Step 6: Restart the Ops Manager server.

For example:

```
<install_dir>/bin/mongodb-mms start
```

Step 7: Update all Monitoring Agents.

See *Install Monitoring Agent* for more information.

Step 8: Update the Backup Daemon and any Backup Agent, as appropriate.

If you are running Backup, update the Backup Daemon package and any Backup Agent.

See *Install Backup Agent* for more information.

Upgrade the Backup Daemon

Step 1: Stop the currently running instance.

```
<install_dir>/bin/mongodb-mms-backup-daemon stop
```

Step 2: Download the latest version of the Backup Daemon.

Download the new version of the Backup Daemon archive by issuing a `curl` command with the download link available on the customer downloads page provided to you by MongoDB.

```
curl -OL <link-address-for-backup-tar.gz>
```

Step 3: To install the Backup Daemon, extract the downloaded archive file.

```
tar -zxf <downloaded-archive-file>
```

Step 4: Point the Backup Daemon to the Ops Manager Application database.

Open the `<install-dir>/conf/conf-daemon.properties` file and set the *`mongo.mongoUri`* value to the servers and ports hosting the Ops Manager Application database. For example:

```
mongo.mongoUri=mongodb://mongodb1.example.net:27017,mongodb2.example.net:27017,
↪mongodb3.example.net:27017
```

Additionally, ensure that the file system that holds the `rootDirectory` has sufficient space to accommodate the current snapshots of all backed up instances.

Step 5: Synchronize the `mms-gen-key` file.

Synchronize the `<install-dir>/bin/mms-gen-key` file from the Ops Manager Application server. This is required only if the Backup Daemon is installed on a different server than the Ops Manager Application.

Step 6: Start the Backup Daemon.

To start the Backup Daemon run:

```
<install_dir>/bin/mongodb-mms-backup-daemon start
```

If you run into any problems, the log files are at `<install_dir>/logs`.

Step 7: Open Ops Manager and access the Backup configuration page.

Open the Ops Manager home page and log in as the user you first registered when installing the Ops Manager Application. (This user is the *global owner*.) Then click the *Admin* link at the top right of the page. Then click the *Backup* tab.

Step 8: Enter configuration information for the Backup database.

Enter the configuration information described below, and then click *Save*. Ops Manager uses this information to create the connection string URI used to connect to the database.

<hostname>:<port>: Enter a comma-separated list of the fully qualified domain names and port numbers for all replica set members for the Backup database.

MongoDD Auth Username and *MongoDB Auth Password*: Enter the user credentials if the database uses authentication.

Encrypted Credentials: Check this if the user credentials use the Ops Manager `credentialstool`. For more information, see *Encrypt MongoDB User Credentials*.

Use SSL: Check this if the MongoDB database uses SSL. If you select this, you must configure SSL settings for both the Ops Manager Application and Backup Daemon. See *Ops Manager Configuration Files*.

Connection Options: To add additional connection options, enter them using the MongoDB Connection String URI Format.

Upgrade from Version 1.2 and Earlier

On this page

Warning: To use Ops Manager 1.6 Automation to manage MongoDB Enterprise deployments, you must **first install** the *MongoDB Enterprise dependencies* to each server that runs MongoDB Enterprise. You must install the dependencies to the servers before using Automation.

Note that Automation **does not yet support** use of the MongoDB Enterprise advanced security features (SSL, LDAP, Kerberos, and auditing). Automation will support these features in the next major Ops Manager release.

To run MongoDB Enterprise with advanced security now, deploy MongoDB Enterprise manually (not through Automation) and use Ops Manager only for Monitoring and Backup.

Overview

Because of a company name change, the name of the Ops Manager package changed between versions 1.2 and 1.3. Therefore, to upgrade from *any* version before 1.3, use the following procedure:

Procedure

1. *Recommended.* Take a full backup of the MMS database before beginning the upgrade procedure.
2. Shut down MMS, using the following command:

   ```
   /etc/init.d/10gen-mms stop
   ```

3. Download the latest Ops Manager package from the downloads page and proceed with the instructions for a fresh install. Do not attempt to use your package manager to do an upgrade.
4. Follow *the procedure for a new install*, including steps to configure the `conf-mms.properties` file. If you used encrypted authentication credentials you will need to regenerate these manually. *Do not copy the credentials from your old properties file. Old credentials will not work.*
5. Start Ops Manager using the new package name.

   For upgrades using rpm or deb packages, issue:

   ```
   sudo /etc/init.d/mongodb-mms start
   ```

   For upgrades using tar.gz or zip archives, issue:

   ```
   <install_dir>/bin/mongodb-mms start
   ```

6. Update the Monitoring Agent. See *Install Monitoring Agent* for more information.

2.7 Configure Local Mode if Ops Manager has No Internet Access

On this page

Overview

The Automation Agent requires access to MongoDB binaries in order to install MongoDB on new deployments or change MongoDB versions on existing ones. In a default configuration, the agents access the binaries over the internet from MongoDB Inc. If you deploy MongoDB on servers that have no internet access, you can run Automation by configuring Ops Manager to run in "local" mode, in which case the Automation Agents access the binaries from a directory on the Ops Manager Application server.

Specify the directory in the *conf-mms.properties* configuration file and then place `.tgz` archives of the binaries in that directory. The Automation Agents will use these archives for all MongoDB installs. The "mongodb-mms" user must possess the permissions to read the `.tgz` files in the "binaries" directory.

The following shows the `ls -l` output of the "binaries" directory for an example Ops Manager install that deploys only the versions of MongoDB listed:

```
$ cd /opt/mongodb/mms/mongodb-releases
$ ls -l
total 355032
-rw-r----- 1 mongodb-mms staff 116513825 Apr 27 15:06 mongodb-linux-x86_64-2.6.9.tgz
-rw-r----- 1 mongodb-mms staff  50194275 Apr 27 15:05 mongodb-linux-x86_64-3.0.2.tgz
-rw-r----- 1 mongodb-mms staff  95800685 Apr 27 15:05 mongodb-linux-x86_64-enterprise-
↪amzn64-2.6.9.tgz
-rw-r----- 1 mongodb-mms staff  50594134 Apr 27 15:04 mongodb-linux-x86_64-enterprise-
↪amzn64-3.0.2.tgz
-rw-r----- 1 mongodb-mms staff  50438645 Apr 27 15:04 mongodb-linux-x86_64-enterprise-
↪suse11-3.0.2.tgz
```

Prerequisites

Download Binaries Before Importing a Deployment

Populate the binaries directory with all required MongoDB versions **before you import the deployment**. If a version is missing, the Automation Agents will not be able to take control of the deployment.

Determine Which Binaries to Store

Your binaries directory will require archives of following versions:

- versions used by existing deployments that you will import
- versions you will use to create new deployments
- versions you will use during an intermediary step in an upgrade. For example, if you will import an existing MongoDB 2.6 Community deployment and upgrade it first to MongoDB 3.0 Community and then to MongoDB 3.0 Enterprise, you must include all those editions and versions.

If you use both the MongoDB Community edition and the MongoDB Enterprise subscription edition, you must include the required versions of both.

The following table describes the archives required for specific versions:

Edition	Version	Archive
Community	2.6+, 2.4+	Linux archive at http://www.mongodb.org/downloads.
Community	3.0+	`Linux 64-bit legacy` version at http://www.mongodb.org/downloads.
Enterprise	3.0+, 2.6+, 2.4+	Platform-specific archive available from http://mongodb.com/download.

Warning: If you run Ops Manager 1.6 and use Automation to deploy MongoDB Enterprise, then you must **pre-install** the *MongoDB Enterprise Dependencies* on your servers.

If your MongoDB Enterprise deployments use Enterprise's advanced security features (SSL, LDAP, Kerberos, and auditing), you can use Ops Manager for Monitoring and Backup only. You cannot currently use Automation with the advanced security features. Automation will support these features in the next major Ops Manager release.

Prerequisite Packages Required for MongoDB Enterprise

If you use MongoDB Enterprise you must install the following prerequisite packages to each server that will run MongoDB Enterprise:

- `net-snmp`
- `net-snmp-libs`
- `openssl`
- `net-snmp-utils`
- `cyrus-sasl`
- `cyrus-sasl-lib`
- `cyrus-sasl-devel`
- `cyrus-sasl-gssapi`

To install these packages on RHEL, CentOS and Amazon Linux, you can issue the following command:

```
sudo yum install openssl net-snmp net-snmp-libs net-snmp-utils cyrus-sasl cyrus-sasl-
↪lib cyrus-sasl-devel cyrus-sasl-gssapi
```

Version Manifest

When you run in local mode, you must provide Ops Manager with the MongoDB version manifest, which lists all released MongoDB versions. You provide the manifest to make Ops Manager aware of MongoDB versions, but to make a version available to the Automation Agent, you must install the version's archive on the Ops Manager Application server and then select it for use in the associated group's Version Manager.

To provide Ops Manager with the manifest, , in the Ops Manager web interface, click *Admin* in the upper right, then click *General*, and then click *Version Manifest*. Here, you can update the manifest directly, if your browser has internet access. Otherwise, copy the manifest from https://opsmanager.mongodb.com/static/version_manifest/1.6.json and click the link for pasting in the manifest.

When MongoDB releases new versions, update the manifest.

Required Access

You must have *Global Automation Admin* or *Global Owner* access to perform this procedure.

Procedure

Step 1: Stop the Ops Manager Application if not yet running in local mode.

Use the command appropriate to your operating system.

On a Linux system installed with a package manager:

```
sudo service mongodb-mms stop
```

On a Linux system installed with a .tar file:

```
<install_dir>/bin/mongodb-mms stop
```

Step 2: Edit the `conf-mms.properties` configuration file to enable local mode and to specify the local directory for MongoDB binaries.

Open *conf-mms.properties* with root privileges and set the following *automation.versions values*:

Set the *`automation.versions.source`* setting to the value `local`:

```
automation.versions.source=local
```

Set *`automation.versions.directory`* to the directory on the Ops Manager Application server where you will store `.tgz` archives of the MongoDB binaries for access by the Automation Agent.

For example:

```
automation.versions.directory=/opt/mongodb/mms/mongodb-releases/
```

Step 3: Start the Ops Manager Application.

Use the command appropriate to your operating system.

On a Linux system installed with a package manager:

```
sudo service mongodb-mms start
```

On a Linux system installed with a .tar file:

```
<install_dir>/bin/mongodb-mms start
```

Step 4: Populate the Ops Manager Application server directory with the `.tgz` files for the MongoDB binaries.

Populate the directory you specified in the `automation.versions.directory` setting with the necessary versions of MongoDB as determined by the *Determine Which Binaries to Store* topic on this page.

Important: If you have not yet read the *Determine Which Binaries to Store* topic on this page, please do so before continuing with this procedure.

For example, to download MongoDB Enterprise 3.0 on Amazon Linux, issue a command similar to the following, replacing `<download-url.tgz>` with the download url for the archive:

```
sudo curl -OL <download-url.tgz>
```

Step 5: Ensure that the "mongodb-mms" user can read the MongoDB binaries.

The "mongodb-mms" user must be able to read the `.tgz` files placed in the directory you specified in the `automation.versions.directory`.

For example, if on a Linux platform you place the `.tgz` files in the `/opt/mongodb/mms/mongodb-releases/` directory, you could use the following sequence of commands to change ownership for **all** files in that directory to "mongodb-mms":

```
cd /opt/mongodb/mms/mongodb-releases/
sudo chown mongodb-mms:mongodb-mms ./*
```

Step 6: Open Ops Manager.

If you have not yet registered a user, click the *Register* link and follow the prompts to register a user and create the first group. The first registered user is automatically assigned the *Global Owner* role.

Step 7: Copy the version manifest to Ops Manager.

1. If you have not already done so, copy the *version manifest* from https://opsmanager.mongodb.com/static/version_manifest/1.6.json.
2. Click the *Admin* link in the upper right to go to the system-wide Administration settings.
3. Click the *General* tab and then click *Version Manifest*.
4. Do one of the following:
   - If your browser has internet access, click *Update the MongoDB Version Manifest* and paste in the manifest.
   - If your browser does not have internet access, follow the instructions on the page.

Step 8: Specify which versions are available for download by Automation Agents associated with each group.

1. Click *Ops Manager* in the upper left to leave the system-wide Administration settings.
2. Click *Deployment* and then click *Version Manager*.
3. Select the checkboxes for the versions of MongoDB that you have made available on the Ops Manager Application server.
4. Click *Review & Deploy* at the top of the page.
5. Click *Confirm & Deploy*.

Step 9: Install the Automation Agent on each server on which you will manage MongoDB processes.

1. Click *Administration* and then *Agents*.
2. In the *Automation* section of the page, click the link for the operating system to which you will install. Following the installation instructions.
3. Pre-install the *MongoDB Enterprise Dependencies* if deploying MongoDB Enterprise.

2.8 Configure High Availability

Application High Availability Outlines the process for achieving a highly available Ops Manager deployment.

Backup High Availability Make the Backup system highly available.

Configure a Highly Available Ops Manager Application

On this page

Overview

The Ops Manager Application provides high availability through horizontal scaling and through use of a replica set for the *backing MongoDB instance* that hosts the *Ops Manager Application Database*.

Horizontal Scaling

The *components* of the Ops Manager Application are stateless between requests. Any Ops Manager Application server can handle requests as long as all the servers read from the same backing MongoDB instance. If one Ops Manager Application becomes unavailable, another fills requests.

To take advantage of this for high availability, configure a load balancer to balance between the pool of Ops Manager Application servers. Use the load balancer of your choice. Configure each application server's `conf-mms.properties` file to point the *`mms.centralUrl`* and *`mms.backupCentralUrl`* properties to the load balancer. For more information, see *Ops Manager Configuration Files*.

The *`mms.remoteIp.header`* property should reflect the HTTP header set by the load balancer that contains the original client's IP address, i.e. `X-Forwarded-For`. The load balancer then manages the *Ops Manager HTTP Service* and *Backup HTTP Service* each application server provides.

The Ops Manager Application uses the client's IP address for auditing, logging, and white listing for the API.

Replica Set for the Backing Instance

Deploy a replica set rather than a standalone as the *backing MongoDB instance* for monitoring. Replica sets have automatic failover if the primary becomes unavailable.

When deploying a replica set with members in multiple facilities, ensure that a single facility has enough votes to elect a primary if needed. Choose the facility that hosts the core application systems. Place a majority of voting members and all the members that can become primary in this facility. Otherwise, network partitions could prevent the set from being able to form a majority. For details on how replica sets elect primaries, see Replica Set Elections.

To deploy a replica set, see Deploy a Replica Set.

You can create backups of the replica set using file system snapshots. File system snapshots use system-level tools to create copies of the device that holds replica set's data files.

Prerequisites

Deploy a replica set for the *backing instance* for the *Ops Manager Application Database*. To deploy a replica set, see Deploy a Replica Set.

Procedure

To configure multiple application servers with load balancing:

Step 1: Configure a load balancer with the pool of Ops Manager Application servers.

This configuration depends on the general configuration of your load balancer and environment.

Step 2: Update each Ops Manager Application server with the load balanced URL.

On each server, edit the `conf-mms.properties` file to configure the `mms.centralUrl` and `mms.backupCentralUrl` properties to point to the load balancer URL.

The `conf-mms.properties` file is located in the `<install_dir>/conf/` directory. See *Ops Manager Configuration Files* for more information.

Step 3: Update each Ops Manager Application server with the replication hosts information.

On each server, edit the `conf-mms.properties` file to set the *`mongo.mongoUri`* property to the connection string of the *Ops Manager Application Database*. You *must* specify at least **3** hosts in the *`mongo.mongoUri`* connection string. For example:

```
mongo.mongoUri=mongodb://<mms0.example.net>:<27017>,<mms1.example.net>:<27017>,<mms2.
↪example.net>:<27017>/?maxPoolSize=100
```

Step 4: Synchronize the `gen.key` file across all the Ops Manager Application servers.

Synchronize the `/etc/mongodb-mms/gen.key` file across all Application Servers. The Ops Manager Application server uses this file to encrypt sensitive information before storing the data in a database.

Additional Information

For information on making Ops Manager Backup highly available, see *Configure a Highly Available Ops Manager Backup Service*.

Configure a Highly Available Ops Manager Backup Service

On this page

Overview

The *Backup Daemon* maintains copies of the data from your backed up `mongod` instances and creates snapshots used for restoring data. The file system that the Backup Daemon uses must have sufficient disk space and write capacity to store the backed up instances.

For replica sets, the local copy is equivalent to an additional secondary replica set member. For sharded clusters the daemon maintains a local copy of each shard as well as a copy of the config database.

To configure high availability

- scale your deployment horizontally by using multiple *backup daemons*, and
- provide failover for your Ops Manager Application Database and Backup Database by deploying replica sets for the dedicated MongoDB processes that host the databases.

Multiple Backup Daemons

To increase your storage and to scale horizontally, you can run multiple instances of the Backup Daemon. With multiple daemons, Ops Manager binds each backed up replica set or shard to a particular Backup Daemon. For example, if you run two Backup Daemons for a cluster that has three shards, and if Ops Manager binds two shards to the first daemon, then that daemon's server replicates only the data of those two shards. The server running the second daemon replicates the data of the remaining shard.

Multiple Backup Daemons allow for **manual** failover should one daemon become unavailable. You can instruct Ops Manager to transfer the daemon's backup responsibilities to another Backup Daemon. Ops Manager reconstructs the data on the new daemon's server and binds the associated replica sets or shards to the new daemon. See *Move Jobs from a Lost Backup Service to another Backup Service* for a description of this process.

Ops Manager reconstructs the data using a snapshot and the oplog from the *Backup Blockstore Database*. Installing the Backup Daemon is part of the procedure to *Install Ops Manager*. Select the procedure specific to your operation system.

Replica Sets for Application and Backup Data

Deploy replica sets rather than standalones for the *dedicated MongoDB processes* that host the Ops Manager Application Database and Backup Database. Replica sets provide automatic failover should the primary become unavailable.

When deploying a replica set with members in multiple facilities, ensure that a single facility has enough votes to elect a primary if needed. Choose the facility that hosts the core application systems. Place a majority of voting members and all the members that can become primary in this facility. Otherwise, network partitions could prevent the set from being able to form a majority. For details on how replica sets elect primaries, see Replica Set Elections.

To deploy a replica set, see Deploy a Replica Set.

Additional Information

To move jobs from a lost Backup server to another Backup server, see *Move Jobs from a Lost Backup Service to another Backup Service*.

For information on making the Ops Manager Application highly available, see *Configure a Highly Available Ops Manager Application*.

2.9 Configure Backup Jobs and Storage

Backup Data Locality Use multiple Backup daemons and blockstore instances to improve backup data locality.

Manage Backup Daemon Jobs Manage job assignments among the backup daemon

Configure Multiple Blockstores in Multiple Data Centers

On this page

Overview

The *Backup Blockstore databases* are the primary storage systems for the backup data of your MongoDB deployments. You can *add new blockstores* to your data center if existing ones have reached capacity.

If needed, you can also deploy blockstores in multiple data centers and assign backups of particular MongoDB deployments to particular data centers, as described on this page. You assign backups to data centers by attaching specific Ops Manager groups to specific blockstores.

Deploy blockstores to multiple data centers when:

- Two sets of backed up data cannot have co-located storage for regulatory reasons.
- You have multiple data centers and want to reduce cross-data center network traffic by keeping each blockstore in the data center it backs.

This tutorial sets up two blockstores in two separate data centers and attaches a separate group to each.

Prerequisites

Each data center hosts a *Backup Blockstore database* and requires its own *Ops Manager Application*, *Backup Daemon*, and *Backup Agent*

The two Ops Manager Application instances must share a single dedicated *Ops Manager Application Database*. You can put members of the Ops Manager Application database replica set in each data center.

Configure each Backup Agent to use the URL for its local Ops Manager Application. You can configure each Ops Manager Application to use a different hostname, or you can use split-horizon DNS to point each agent to its local Ops Manager Application.

The Ops Manager Application database and the Backup Blockstore databases are MongoDB databases and can run as standalones or replica sets. For production deployments, use replica sets to provide database high availability.

Procedures

Provision Servers in Each Data Center

Each server must meet the cumulative hardware and software requirements for the components it runs. See *Ops Manager Hardware and Software Requirements*.

Servers that run the Backup Damon, Ops Manager Application database, and the Backup Blockstore databases all run MongoDB. They must meet the configuration requirements in the MongoDB Production Notes.

Install MongoDB

Install MongoDB on the servers that host the:

- Backup Daemon
- Ops Manager Application database
- Backup Blockstore databases

See Install MongoDB in the MongoDB manual to find the correct install procedure for your operating system. To run replica sets for the Ops Manager Application database and Backup Blockstore databases, see Deploy a Replica Set in the MongoDB manual.

Install the Ops Manager Application

Install the Ops Manager Application in each data center but **do not** perform the step to *start* the service. See *Install Ops Manager* to find the procedure for your operating system.

In the step for configuring the `conf-mms.properties` file, set the same Ops Manager Application URL in both data centers. For example, in both data centers, set *`mms.centralUrl`* and `mms.centralBackupUrl` to point to the server in Data Center 1:

```
mms.centralUrl = <application-server-in-data-center-1>:<8080>
mms.centralBackupUrl = <application-server-in-data-center-1>:<8081>
```

Start the Ops Manager Application

The Ops Manager Application creates a `gen.key` file on initial startup. You must start the Ops Manager Application in one data center and copy its `gen.key` file **before** starting the other Ops Manager Application. Ops Manager uses the same `gen.key` file for **all** servers in both data centers.

The `gen.key` file is binary. You cannot copy the contents: you must copy the file. For example, use SCP.

Step 1: Start the Ops Manager Application in Data Center 1.

Issue the following:

```
service mongodb-mms start
```

Step 2: Copy the `gen.key` file.

Copy the `/etc/mongodb-mms/gen.key` file from the Ops Manager Application server in Data Center 1 to the:

- `/etc/mongodb-mms` directory on the Ops Manager Application server in Data Center 2.
- `/etc/mongodb-mms` directory of each Backup Daemon server in each data center.

Step 3: Start the Ops Manager Application server in Data Center 2.

Issue the following:

```
service mongodb-mms start
```

Install the Backup Daemon

Install and start the Backup Daemon in each data center. See *Install Ops Manager* for instructions for your operating system.

Bind Groups to Backup Resources

Step 1: In a web browser, open Ops Manager.

Step 2: Create a new Ops Manager group for the first data center.

To create a group, select the *Administration* tab and open the *My Groups* page. Then, click *Add Group* and specify the group name."

Step 3: Create a second Ops Manager group for the second data center.

Step 4: Open the Admin interface.

In Ops Manager, select the *Admin* link in the upper right.

Step 5: Configure backup resources.

Select the *Backup* tab and do the following:

- Select the *Daemons* page and ensure there are two daemons listed.
- Select the *Blockstores* page. Add a blockstore with the hostname and port of the blockstore for the second data center and click *Save*.
- Select the *Sync Stores* page. Add a sync store with the hostname and port of the blockstore for the second data center and click *Save*.
- Select the *Oplog Stores* page. Add an oplog store with the hostname and port of the blockstore for the second data center and click *Save*.

Step 6: Assign resources to the data centers.

Open the *General* tab, then the *Groups* page. Select the group you will house in the first data center, and then select the *View* link for the *Backup Configuration*.

For each of the following, click the drop-down box and select the local option for the group:

- *Backup Daemons*
- *Sync Stores*
- *Oplog Stores*
- *Block Stores*

Repeat the above steps for the second group.

Step 7: Install agents.

If you are using Automation, *install the Automation Agent* for the group in Data Center 1 on each server in Data Center 1. Install the Automation Agent for Data Center 2 on each server in Data Center 2. The Automation Agent will then install Monitoring and Backup agents as needed.

If you are not using Automation, download and install the Monitoring and Backup agents for the group assigned to Data Center 1 by navigating to the *Administration* and then *Agents* page while viewing that group in Ops Manager.

Then, switch to the group in Data Center 2 by choosing it from the drop-down menu in the top navigation bar in Ops Manager, and download and install its Monitoring and Backup agents.

See the following pages for the procedures for installing the agents manually:

- *Install Monitoring Agent*
- *Install Backup Agent*

Move Jobs from a Lost Backup Service to another Backup Service

On this page

Overview

If the server running a *Backup Daemon* fails, and if you *run multiple Backup Daemons*, then an administrator with the `global owner` or `global backup admin` *role* can move all the daemon's jobs to another Backup Daemon. The new daemon takes over the responsibility to back up the associated shards and replica sets.

When you move jobs, the destination daemon reconstructs the data using a snapshot and the oplog from the *Backup Blockstore database*. Reconstruction of data takes time, depending on the size of the databases on the source.

During the time it takes to reconstruct the data and reassign the backups to the new Backup Daemon:

- Ops Manager Backup does not take new snapshots of the jobs that are moving until the move is complete. Jobs that are not moving are not affected.
- Ops Manager Backup *does* save incoming oplog data. Once the jobs are on the new Backup Daemon's server, Ops Manager Backup takes the missed snapshots at the regular snapshot intervals.
- Restores of previous snapshots are still available.
- Ops Manager can produce restore artifacts using existing snapshots with *point-in-time recovery* for replica sets or *checkpoints* for sharded clusters.

Procedure

With administrative privileges, you can move jobs between Backup daemons using the following procedure:

Step 1: Click the *Admin* link at the top of Ops Manager.

Step 2: Select *Backup* and then select *Daemons*.

The *Daemons page* lists all active Backup Daemons.

Step 3: Locate the failed Backup Daemon and click the *Move all heads* link.

Ops Manager displays a drop-down list from which to choose the destination daemon. The list displays only those daemons with more free space than there is used space on the source daemon.

Step 4: Move the jobs to the new daemon.

Select the destination daemon and click the *Move all heads* button.

2.10 Test Ops Manager Monitoring

On this page

Overview

The following procedure creates a MongoDB replica set and sets up a database populated with random data for use in testing an Ops Manager installation. Create the replica set on a separate machine from Ops Manager.

These instructions create the replica set on a server running **RHEL 6+** or **Amazon Linux**. The procedure installs all the members to one server.

Procedure

Step 1: Increase each server's default `ulimit` settings.

If you are installing to RHEL, check whether the `/etc/security/limits.d` directory contains the `90-nproc.conf` file. If the file exists, remove it. (The `90-nproc.conf` file overrides `limits.conf`.) Issue the following command to remove the file:

```
sudo rm /etc/security/limits.d/90-nproc.conf
```

For more information, see UNIX ulimit Settings in the MongoDB manual.

Step 2: Install MongoDB on each server.

First, set up a repository definition by issuing the following command:

```
echo "[mongodb-org-3.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.0/x86_64/
gpgcheck=0
enabled=1" | sudo tee -a /etc/yum.repos.d/mongodb-org-3.0.repo
```

Second, install MongoDB by issuing the following command:

```
sudo yum install -y mongodb-org mongodb-org-shell
```

Step 3: Create the data directories for the replica set.

Create a data directory for each replica set member and set `mongod.mongod` as each data directory's owner.

For example, the following command creates the directory `/data` and then creates a data directory for each member of the replica set. You can use different directory names:

```
sudo mkdir -p /data /data/mdb1 /data/mdb2 /data/mdb3
```

The following command sets `mongod.mongod` as owner of the new directories:

```
sudo chown mongod:mongod /data /data/mdb1 /data/mdb2 /data/mdb3
```

Step 4: Start a separate MongoDB instance for each replica set member.

Start each `mongod` instance on its own dedicated port number and with the data directory you created in the last step. For each instance, specify `mongod` as the user. Start each instance with the replSet command-line option specifying the name of the replica set.

For example, the following three commands start three members of a new replica set named `rs0`:

```
sudo -u mongod mongod --port 27017 --dbpath /data/mdb1 --replSet rs0 --logpath /data/
↪mdb1/mongodb.log --fork

sudo -u mongod mongod --port 27018 --dbpath /data/mdb2 --replSet rs0 --logpath /data/
↪mdb2/mongodb.log --fork

sudo -u mongod mongod --port 27019 --dbpath /data/mdb3 --replSet rs0 --logpath /data/
↪mdb3/mongodb.log --fork
```

Step 5: Connect to one of the members.

For example, the following command connects to the member running on port `27017`:

```
mongo --port 27017
```

Step 6: Initiate the replica set and add members.

In the `mongo` shell, issue the rs.initiate() and rs.add() methods, as shown in the following example. **Replace the hostnames in the example** with the hostnames of your servers:

```
rs.initiate()
rs.add("mdb.example.net:27018")
rs.add("mdb.example.net:27019")
```

Step 7: Verify the replica set configuration.

Issue the rs.conf() method and verify that the `members` array lists the three members:

```
rs.conf()
```

Step 8: Add data to the replica set.

Issue the following `for` loop to create a collection titled `testData` and populate it with 25,000 documents, each with an `_id` field and a field `x` set to a random string.

```
for (var i = 1; i <= 25000; i++) {
  db.testData.insert( { x : Math.random().toString(36).substr(2, 15) } );
  sleep(0.1);
}
```

Step 9: Confirm data entry.

After the script completes, you can view a document in the `testData` collection by issuing the following:

```
db.testData.findOne()
```

To confirm that the script inserted 25,000 documents into the collection, issue the following:

```
db.testData.count()
```

Step 10: Open the Ops Manager home page in a web browser and register the first user.

The first user created for Ops Manager is automatically assigned the *Global Owner* role.

Enter the following URL in a web browser, where `<ip_address>` is the IP address of the server:

```
http://<ip_address>:8080
```

Click the *Register* link and enter information for the new *Global Owner* user. When you finish, you are logged into the Ops Manager Application as that user. For more information on creating and managing users, see *Manage Ops Manager Users*.

Step 11: Set up monitoring for the replica set.

If you have installed the *Backup Daemon*, click the *Get Started* button for *Backup* and follow the instructions. This will set up both monitoring and backup. Otherwise click the *Get Started* button for *Monitoring* and follow instructions.

When prompted to add a host, enter the hostname and port of one of the replica set members in the form `<hostname>:<port>`. For example: `mdb.example.net:27018`

When you finish the instructions, Ops Manager is running and monitoring the replica set.

3 Create a New MongoDB Deployment

Add Servers for Use by Automation Add servers to Ops Manager.

Deploy a Replica Set Use Ops Manager to deploy a managed replica set.

Deploy a Sharded Cluster Use Ops Manager to deploy a managed sharded cluster.

Deploy a Standalone For testing and deployment, create a new standalone MongoDB instance.

Connect to a MongoDB Process Connect to a MongoDB deployment managed by Ops Manager.

3.1 Add Servers for Use by Automation

This section describes how to add servers for use by Ops Manager Automation.

Overview How to add servers to Ops Manager.

Add Existing Servers to Ops Manager Add your existing servers to Ops Manager.

Overview

On this page

Add Servers for Use by Automation

You can add servers to *Automation* in the following ways:

- Provision existing systems and infrastructure. Ops Manager can deploy and manage MongoDB on your existing servers. To use your existing hardware, you must deploy the Automation Agent to each server on which Ops Manager will deploy MongoDB. See *Add Existing Servers to Ops Manager*.
- Use your local system for testing. Ops Manager can deploy MongoDB to your laptop or desktop system. Do not use local systems for production deployments. To use your local system, you must deploy the Automation Agent.

Server Requirements

The following are the minimum requirements for the servers that will host your MongoDB deployments:

Hardware:

- At least 10 GB of free disk space plus whatever space is necessary to hold your MongoDB data.
- At least 4 GB of RAM.
- If you use Amazon Web Services (AWS) EC2 instances, we recommend at least an `m3.medium` instance.

Networking:

- For each server that hosts a MongoDB process managed by Ops Manager, the output of `hostname -f` must generate a hostname that is resolvable from all other servers in the MongoDB deployment.

Software:

If you provision your own servers and if you use MongoDB Enterprise, you must install the *prerequisite packages* on the servers before deploying MongoDB Enterprise to the servers.

Add Existing Servers to Ops Manager

On this page

Overview

You can allow Ops Manager to install, manage, and discover MongoDB processes on your existing servers. To do so, you install the *Automation Agent* on each server.

Prerequisites

Ensure that the *directories used by the Automation Agent* have appropriate permissions for the user running the agent.

You can install the Automation Agent on the operating systems listed in Ops Manager on the *Administration* tab's *Agents* page.

To install the agent using `rpm` or `deb` packages, you must have root access.

Important: If a server runs MongoDB Enterprise, you must install the *prerequisite packages* on the server before deploying MongoDB Enterprise on the servers.

Procedure

Install the Automation Agent on each each server that you want Ops Manager to manage. The following procedure applies to all operating systems. For instructions for a specific operating system, see *Install the Automation Agent*.

Step 1: Select the *Administration* tab and then select *Agents*.

Step 2: Under *Automation* at the bottom of the page, click your operation system and follow the instructions to install and run the agent.

If the install file is a `tar.gz` archive file, make sure to extract the archive after download.

Step 3: Ensure that the directories used by the Automation Agent have appropriate permissions for the user that runs the agent.

Set the required permissions described in *Directory and File Permissions*.

Once you have installed the agent to all your servers, you can deploy your first *replica set*, *cluster*, or *standalone*.

3.2 Deploy a Replica Set

On this page

Overview

A replica set is a group of MongoDB deployments that maintain the same data set. Replica sets provide redundancy and high availability and are the basis for all production deployments. See the Replication Introduction in the MongoDB manual for more information about replica sets.

Use this procedure to deploy a new replica set managed by Ops Manager. After deployment, use Ops Manager to manage the replica set, including such operations as adding, removing, and reconfiguring members.

Consideration

Use unique replica set names for different replica sets within an Ops Manager group. Do not give different replica sets the same name. Ops Manager uses the replica set name to identify which set a member belongs to.

Prerequisites

You must have an existing set of servers to which to deploy, and Ops Manager must have access to the servers.

The servers can exist on your own system or on Amazon Web Services (AWS). To give Ops Manager access to servers on your system, install the Automation Agent to each server.

Important: If you provision your own servers and if you use MongoDB Enterprise, you must install the *prerequisite packages* on the servers before deploying MongoDB Enterprise on the servers.

Procedure

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click *Add* and select *Create New Replica Set*.

Step 3: Configure the replica set.

Enter information as required and click *Apply*.

To select specific servers to use for the deployment, enter the prefix of the servers in the *Eligible Server RegExp* field. You can use regular expressions. To use all provisioned servers, enter a period (“.”). To run the deployment on your local machine, enter the name of the machine.

For information on replica set options in the *Member Options* box, see Replica Set Members in the MongoDB Manual. The *votes* field applies only to pre-2.6 versions of MongoDB.

To configure additional `mongod` runtime options, such as specifying the oplog size, or modifying journaling settings, click *Advanced Options*. For option descriptions, see *Advanced Options for MongoDB Deployments*.

Step 4: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 5: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

3.3 Deploy a Sharded Cluster

On this page

Overview

Sharded clusters provide horizontal scaling for large data sets and enable high throughput operations by distributing the data set across a group of servers. See the Sharding Introduction in the MongoDB manual for more information.

Use this procedure to deploy a new sharded cluster managed by Ops Manager. Later, you can use Ops Manager to add shards and perform other maintenance operations on the cluster.

Prerequisites

You must have an existing set of servers to which to deploy, and Ops Manager must have access to the servers.

The servers can exist on your own system or on Amazon Web Services (AWS). To give Ops Manager access to servers on your system, install the Automation Agent to each server.

Important: If you provision your own servers and if you use MongoDB Enterprise, you must install the *prerequisite packages* on the servers before deploying MongoDB Enterprise on the servers.

Procedure

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click *Add* and select *Create New Cluster*.

Step 3: Configure the sharded cluster.

Enter information as required and click *Apply*.

To select specific servers to use for the deployment, enter the prefix of the servers in the *Eligible Server RegExp* field. You can use regular expressions. To use all provisioned servers, enter a period (“.”). To run the deployment on your local machine, enter the name of the machine.

For information on replica set options in the *Member Options* box, see Replica Set Members in the MongoDB Manual. The *votes* field applies only to pre-2.6 versions of MongoDB.

To configure additional `mongod` or `mongos` options, such as specifying the oplog size, or modifying journaling settings, click *Advanced Options*. For option descriptions, see *Advanced Options for MongoDB Deployments*.

Step 4: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 5: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

3.4 Deploy a Standalone MongoDB Instance

On this page

Overview

You can deploy a standalone MongoDB instance managed by Ops Manager. Use standalone instances for testing and development. **Do not** use these deployments, which lack replication and high availability, for production systems. For all production deployments use replica sets. See *Deploy a Replica Set* for production deployments.

Prerequisites

You must have an existing server to which to deploy. For testing purposes, you can use your localhost, or another machine to which you have access.

Important: If you provision your own server and if you use MongoDB Enterprise, you must install the *prerequisite packages* to the server before deploying MongoDB Enterprise on it.

Procedure

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and click *Add* and select *Create New Standalone*.

Step 3: Configure the standalone MongoDB instance.

Enter information as required and click *Apply*. For descriptions of *Advanced Options*, see *Advanced Options for MongoDB Deployments*.

Step 4: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 5: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

3.5 Connect to a MongoDB Process

On this page

Overview

To connect to a MongoDB instance, retrieve the hostname and port information from the Ops Manager console and then use a MongoDB client, such as the `mongo` shell or a MongoDB driver, to connect to the instance. You can connect to a cluster, replica set, or standalone.

Firewall Rules

Firewall rules and user authentication affect your access to MongoDB. You must have access to the server and port of the MongoDB process.

If your MongoDB instance runs on Amazon Web Services (AWS), then the security group associated with the AWS servers also affects access. AWS security groups control inbound and outbound traffic to their associated servers.

Procedures

Get the Connection Information for the MongoDB Instance

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the process.

For a replica set, click the primary. For a sharded cluster, click the `mongod`.

Ops Manager displays the hostname and port of the process at the top of the charts page.

Connect to a Deployment Using the Mongo Shell

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the process.

For a replica set, click the primary. For a sharded cluster, click the `mongod`.

Ops Manager displays the hostname and port of the process above the status charts.

Step 4: On a system shell, run `mongo` and specify the host and port of the deployment.

Issue a command in the following form:

```
mongo --username <user> --password <pass> --host <host> --port <port>
```

Connect to a Deployment Using a MongoDB Driver

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the process.

For a replica set, click the primary. For a sharded cluster, click the `mongod`.

Ops Manager displays the hostname and port of the process at the top of the charts on the page.

Step 4: Connect from your driver.

Use your driver to create a connection string that specifies the hostname and port of the deployment. The connection string for your driver will resemble the following:

```
mongodb://[<username>:<password>@]hostname0<:port>[,hostname1:<port1>][,hostname2:
↪<port2>][...][,hostnameN:<portN>]
```

If you specify a seed list of all hosts in a replica set in the connection string, your driver will automatically connect to the term:*primary*.

For standalone deployments, you will only specify a single host. For sharded clusters, only specify a single `mongos` instance.

Retrieve the Command to Connect Directly from the Process's Server

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the instance's gear icon and select *Connect to this Instance*.

Ops Manager provides a `mongo` shell command that you can use to connect to the MongoDB process if you are connecting from the system where the deployment runs.

4 Import an Existing MongoDB Deployment

Add Existing Processes to Monitoring Add existing MongoDB processes to Ops Manager Monitoring.

Add Monitored Processes to Automation Add an existing MongoDB deployment to be managed through Ops Manager Automation.

Reactivate Monitoring for a Process Reactivate a deactivated MongoDB process.

Remove Hosts Remove processes you no longer use from monitoring.

4.1 Add Existing MongoDB Processes to Monitoring

On this page

Overview

You can monitor existing MongoDB processes in Ops Manager by adding the hostnames and ports of the processes. Ops Manager will start monitoring the `mongod` and `mongos` instances.

If you add processes from an environment that uses authentication, you must add each `mongod` instance separately and explicitly set the authentication credentials on each.

If you add processes in an environment that does *not* use authentication, you can manually add one process from a replica set or a sharded cluster as a seed. Once the Monitoring Agent has the seed, it automatically discovers all the other nodes in the replica set or sharded cluster.

Unique Replica Set Names

Do not add two different replica sets with the same name. Ops Manager uses the replica set name to identify which set a member belongs to.

Preferred Hostnames

If the MongoDB process is accessible only by specific hostname or IP address, or if you need to specify the hostname to use for servers with multiple aliases, set up a preferred hostname. For details, see the *Preferred Hostnames* setting in *Group Settings*.

Prerequisite

You must *install a Monitoring Agent* to your infrastructure.

To monitor or back up MongoDB 3.0 deployments, you must install Ops Manager 1.6 or higher. To monitor a MongoDB 3.0 deployment, you must also run Monitoring Agent version 2.7.0 or higher.

Important: If you provision your own servers and if you use MongoDB Enterprise, you must install the *prerequisite packages* on the servers before deploying MongoDB Enterprise on the servers.

Add MongoDB Processes

If your deployments use authentication, perform this procedure for each process. If your deployment does not use authentication, add one process from a replica set or sharded cluster and Ops Manager will discover the other nodes in the replica set or sharded cluster.

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Select *view mode* and then select *Add Host*.

Step 3: Enter information for the MongoDB process.

Enter the following information, as appropriate:

Host Type	The type of MongoDB deployment.
Internal Hostname	The hostname of the MongoDB instance as seen from the Monitoring Agent.
Port	The port on which the MongoDB instance runs.
Auth Mechanism	The authentication mechanism used by the host. • *MONGODB-CR*, • *LDAP (PLAIN)*, or • *Kerberos(GSSAPI)*. See *Add Monitoring Agent User for MONGODB-CR*, *Configure Monitoring Agent for LDAP*, or *Configure the Monitoring Agent for Kerberos* for setting up user credentials.
DB Username	If the authentication mechanism is MONGODB-CR or LDAP, the username used to authenticate the Monitoring Agent to the MongoDB deployment.
DB Password	If the authentication mechanism is MONGODB-CR or LDAP, the password used to authenticate the Monitoring Agent to the MongoDB deployment.
My deployment supports SSL for MongoDB connections	If checked, the Monitoring Agent must have a trusted CA certificate in order to connect to the MongoDB instances. See *Configure Monitoring Agent for SSL*.

Step 4: Click *Add*.

To view agent output logs, click the *Administration* tab, then *Agents*, and then *view logs* for the agent.

To view process logs, click the *Deployment* tab, then the *Deployment* page, then the process, and then the *Logs* tab.

For more information on logs, see *View Logs*.

4.2 Add Monitored Processes to Automation

On this page

Overview

Ops Manager can automate operations for your monitored MongoDB processes. Adding your processes to Automation lets you reconfigure, stop, and restart MongoDB through the Ops Manager interface.

Adding monitored processes involves two steps. First, install the Automation Agent on each server hosting a monitored MongoDB process. Second, add the processes to Automation through the Ops Manager interface.

Prerequisites

Automation Agents can run only on 64-bit architectures.

Automation supports most but not all available MongoDB options. Automation supports the options described in *Supported MongoDB Options for Automation*.

The user running the Automation Agent must the same as the user running the MongoDB process to be managed.

The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (retrieved on each server by issuing `hostname -f`). Each server must be able to reach every other server through the FQDN.

Ensure that your network configuration allows each Automation Agent to connect to every MongoDB process listed on the *Deployment* tab. Ensure that the network and security systems, including all interfaces and firewalls, allow these connections.

Ops Manager must be currently monitoring the MongoDB processes, and the Monitoring Agent must be running. The processes must appear in the Ops Manager *Deployment* tab.

Procedures

Install the Automation Agent on Each Server

Install the Automation Agent on each server that hosts a monitored MongoDB process. Ensure that the Automation Agent has adequate permissions to stop and restart the existing Monitoring Agent so that it can update the Monitoring Agent as new versions are released.

On each server, you must download the agent, create the necessary directories, and configure the agent's `local.config` file with the Group ID and API key.

Step 1: In Ops Manager, select the *Administration* tab and then select *Agents*.

Step 2: Under *Automation* at the bottom of the page, click your operation system and follow the instructions to install and run the agent.

For additional information on installing the agent, see *Install the Automation Agent*.

Add the Processes to Automation

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click the *Add* button and select *Attach to Existing*.

Step 3: Select the MongoDB processes.

Click the *Deployment Item* field to display your currently monitored processes. Select the cluster, replica set or standalone to add.

Step 4: Click *Start Import*.

Ops Manager displays the progress of the import for each MongoDB process, including any errors. If you need to correct errors, click *Stop Import*, correct them, and restart this procedure.

Step 5: Click *Confirm Import*.

Step 6: Click *Review & Deploy*.

Step 7: Click *Confirm & Deploy*.

Ops Manager Automation takes over the management of the processes and peforms a rolling restart. To view progress, click *View Agent Logs*.

If you diagnose an error that causes Automation to fail to complete the deployment, click *Edit Configuration* to correct the error.

4.3 Reactivate Monitoring for a Process

On this page

Overview

If the Monitoring Agent cannot collect information from a MongoDB process, Ops Manager stops monitoring the process. By default, Ops Manager stops monitoring a `mongos` that is unreachable for 24 hours and a `mongod` that is unreachable for 7 days. Your group might have different default behavior. Ask your system administrator.

When the system stops monitoring a process, the *Deployment* page marks the process with an `x` in the *Last Ping* column. If the instance is a `mongod`, Ops Manager displays a caution icon at the top of each *Deployment* page.

You can reactivate monitoring for the process whether or not the process is running. When you reactivate monitoring, the Monitoring Agent has an hour to reestablish contact and provide a ping to Ops Manager. If a process is running and reachable, it appears marked with a green circle in the *Last Ping* column. If it is unavailable, it appears marked with a red square. If it remains unreachable for an hour, Ops Manager again stops monitoring the process.

You can optionally remove a process that you are no longer using. Removed processes are permanently hidden from Ops Manager. For more information, see *Remove Hosts*.

Procedure

To reactivate monitoring for a process:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click the warning icon at the top of the page.

Step 3: Click *Reactivate ALL hosts*.

The processes that are now running and reachable by the Monitoring Agent will appear marked with green circles in the *Last Ping* column.

The processes that are unavailable or unreachable will appear marked with a red square. If a process does not send a ping within an hour after reactivation, it is deactivated again.

Step 4: Add the `mongos` instances.

To activate the `mongos` instances, click the *Add Host* button and enter the hostname, port, and optionally an `admin` database username and password. Then click *Add*.

4.4 Remove Hosts

On this page

Overview

You can remove hosts that you no longer use, but when you do they are hidden permanently. If you run the instance again, Ops Manager will not discover it. If you choose to *add* the host again, Ops Manager will **not** display it.

Only a global administrator can undelete the host so that it will again display if added. The administrator can add the host back through the *Deleted Hosts* page in the Ops Manager *Admin* interface.

Instead of removing a host, you can optionally disable alerts for the host, which does not remove it from the *Deployment* pages. See *Manage Host Alerts*.

Procedure

To remove a host from Ops Manager:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: On the line listing the process, click the gear icon and select *Remove Host*.

Step 3: Click *Delete*.

Step 4: If prompted for a two-factor authentication code, enter it, click *Verify*, and then click *Delete* again.

5 Manage Deployments

Edit a Replica Set Add hosts to, remove hosts from, or modify the configuration of hosts in an Ops Manager managed replica set. - 'cloud' - 'onprem'

Migrate a Replica Set Member to a New Server Migrate replica sets to new underlying systems by adding members to the set and decommissioning existing members.

Move or Add a Monitoring or Backup Agent Migrate a backup and monitoring agents to different servers.

Change the Version of MongoDB Upgrade or downgrade MongoDB deployments managed by Ops Manager.

Restart a MongoDB Process Restart MongoDB deployments managed by Ops Manager.

Shut Down MongoDB Processes Shut down MongoDB deployments managed by Ops Manager.

Remove Processes from Monitoring Remove MongoDB deployments from management by Ops Manager.

Alerts Set up and manage alert configurations.

Monitoring Metrics Interpreting the metrics.

Logs View host and agent logs.

5.1 Edit a Replica Set

On this page

Overview

You can add, remove, and reconfigure members in a replica set directly in the Ops Manager console.

Procedures

Add a Replica Set Member

You must have an existing server to which to deploy the new replica set member. To add a member to an existing replica set, increasing the size of the set:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Edit the replica set.

Click *edit mode*. Then click the arrow to the right of the replica set and click the *Edit* button.

Step 3: Add the member.

In the *MongoDs Per Replica Set* field, click + to increase the number of members for the replica set.

Configure the new members as desired in the *Member Options* box.

Then click *Apply*.

Step 4: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 5: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

Edit a Replica Set Member

Use this procedure to:

- Reconfigure a member as hidden
- Reconfigure a member as delayed
- Reset a member's priority level in elections
- Reset a member's votes (for pre-2.6 versions of MongoDB)

To reconfigure a member as an arbiter, see *Replace a Member with an Arbiter*.

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click the arrow to the right of the replica set and click the *Edit* button.

Step 3: In the *Member Options* box, configure each member as needed.

For information on member options, see the following in the MongoDB manual:

- Hidden Members
- Delayed Members
- Elections, which describes priority levels.

The *Votes* field applies to pre-2.6 versions of MongoDB.

Step 4: Click *Apply*.

Step 5: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 6: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

Replace a Member with an Arbiter

You cannot directly reconfigure a member as an arbiter. Instead, you must must add a new member to the replica set as an arbiter. Then you must shut down an existing secondary.

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*.

Step 3: Click the arrow to the right of the replica set and then click the *Edit* button.

Step 4: Add an arbiter.

In the *MongoDs Per Replica Set* field, increase the number of members by 1.

In the *Member Options* box, click the member's drop-down arrow and select *Arbiter*.

Click *Apply*.

Step 5: Click *Review & Deploy*.

Step 6: Click *Confirm & Deploy*.

Step 7: Remove the secondary.

When the deployment completes, click the arrow to the right of the secondary to remove. Click the gear icon drop-down list, and select *Remove Member*."

Step 8: Click *Review & Deploy*.

Step 9: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

Upon completion, Ops Manager removes the member from the replica set, but it will continue to run as a standalone MongoDB instance. To shut down the standalone, see *Shut Down MongoDB Processes*.

Remove a Replica Set Member

Removing a member from a replica set does not shut down the member or remove it from Ops Manager. Ops Manager still monitors the `mongod` as as standalone instance. To remove a member:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click the arrow to the right of the member to be removed.

Step 3: Click the gear icon drop-down list and select *Remove Member*.

Step 4: Click *Remove* to confirm.

Step 5: Click *Review & Deploy*.

Step 6: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

Upon completion, Ops Manager removes the member from the replica set, but it will continue to run as a standalone MongoDB instance. To shut down the standalone, see *Shut Down MongoDB Processes*.

Additional Information

To view data from all replica set members at once, see *Replica Set Statistics*.

For more information on replica set configuration options, see, Replica Set Configuration in the MongoDB manual.

5.2 Migrate a Replica Set Member to a New Server

On this page

Overview

For Ops Manager managed replica sets, you can replace one member of a replica set with another new member from the Ops Manager console. Use this process to migrate members of replica sets to new underlying servers. From a high level, this procedure requires that you: you add a member to the replica set on the new server and then shut down the existing member on the old server. Specifically, you will

1. Provision the new server.
2. Add an extra member to the replica set.
3. Shut down old member of the replica set.
4. Un-manage the old member (Optional).

Considerations

Initial Sync

When you add a new replica set member, the member must perform an initial sync, which takes time to complete, depending on the size of your data set. For more information on initial sync, see Replica Set Data Synchronization.

Migrating Multiple Members

If you are moving multiple members to new servers, migrate each member separately to keep the replica set available.

Procedure

Perform this procedure separately for each member of a replica set to migrate.

Step 1: Provision the new server.

See *Add Existing Servers to Ops Manager*.

Step 2: Select the *Deployment* tab and then the *Deployment* page.

Step 3: Click *edit mode*, and then click the arrow to the right of the replica set and click the *Edit* button.

Step 4: Add a member to the replica set.

In the *Nodes Per Replica Set* field, increase the number of members by 1, and then click *Apply*.

Step 5: Verify changes.

Verify the server to which Ops Manager will deploy the new replica set member. If necessary, select a different server.

The *Deployment* page's *Topology View* lists the new replica set member and indicates the server to which Ops Manager will deploy it.

If Ops Manager has not chosen the server you intended, click the *Deployment* page's *Server View* to display your available servers and their processes. Processes not yet deployed can be dragged to different servers. Drag the new replica set member to the server to which to deploy it.

Step 6: Click *Review & Deploy*.

Step 7: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

Step 8: Verify that the new member has synchronized.

Select the *Deployment* page's *view mode* to view the new member's status. Verify that the new member has synchronized and is no longer in the `Recovering` state.

Step 9: Remove the old member from the replica set.

Select the *Deployment* page's *edit mode*, and then click the arrow to the right of the replica set member. Then click the gear icon and select *Remove Member*. Then click *Review & Deploy*. Then click *Confirm & Deploy*.

Step 10: Shut down the old member.

Select the arrow to the right of the removed replica set member, and then click the gear icon and select *Shut Down*. Then click *Review & Deploy*. Then click *Confirm & Deploy*.

Step 11: Optionally, unmanage the old member.

Select the arrow to the right of the removed replica set member, and then click the gear icon and select *Unmanage*. Then click *Review & Deploy*. Then click *Confirm & Deploy*.

5.3 Move or Add a Monitoring or Backup Agent

On this page

Overview

When you deploy MongoDB as a replica set or sharded cluster to a group of servers, Ops Manager selects one server to run the Monitoring Agent. If you enable Ops Manager Backup, Ops Manager also selects a server to run the Backup Agent.

You can move the Monitoring and Backup Agents to different servers in the deployment. You might choose to do this, for example, if you are terminating a server.

You also can add additional instances of each agent as hot standbys for high availability. However, this is not standard practice. A single Monitoring Agent and single Backup Agent are sufficient and strongly recommended. If you run multiple agents, only one Monitoring Agent and one Backup Agent per group or environment are primary. Only the primary agent reports cluster status and performs backups. If you run multiple agents, see *Confirm Only One Agent is Actively Monitoring*.

Procedures

Move a Monitoring or Backup Agent to a Different Server

To move an agent to a new server, you install a new instance of the agent on the target server, and then remove the agent from its original server.

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then select the *Server View*.

The *Server View* displays each provisioned server that is currently running one or more agents.

Step 3: On the server to which to move the agent, click the gear icon and select to install that type of agent.

Step 4: On the server from which to remove the agent, click the gear icon and remove the agent.

Step 5: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 6: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

Install Additional Agent as Hot Standby for High Availability

In general, using only one Monitoring Agent and one Backup Agent is sufficient and strongly recommended. If you run multiple agents, see *Confirm Only One Agent is Actively Monitoring* to ensure no conflicts.

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then select the *Server View*.

The *Server View* displays each provisioned server that is currently running one or more agents.

Step 3: On the server to which to add an additional agent, click the gear icon and select the agent to add.

Step 4: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 5: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

5.4 Change the Version of MongoDB

On this page

Overview

For Ops Manager managed MongoDB, Ops Manager supports safe automatic upgrade and downgrade operations between releases of MongoDB while maximizing the availability of your deployment. Ops Manager supports upgrade and downgrade operations for sharded clusters, replica sets,and standalone MongoDB instances.

Considerations

Before changing a deployment's MongoDB version:

- Consult the following documents for any special considerations or application compatibility issues:
  - The MongoDB Release Notes
  - The documentation for your driver.
- Plan the version change during a predefined maintenance window.
- Before changing version on a production environment, change versions on a *staging* environment that reproduces your production environment to ensure your configuration is compatible with all changes.
- To monitor or back up MongoDB 3.0 deployments, you must install Ops Manager 1.6 or higher. To monitor a MongoDB 3.0 deployment, you must also run Monitoring Agent version 2.7.0 or higher.

Procedure

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click the arrow to the right of the deployment.

Step 3: In the configuration screen, click the gear icon and select *Edit*.

Step 4: In the *Version* field select the version. Then click *Apply*.

If the drop-down menu does not include the desired MongoDB version, you must first *enable it in the Version Manager*.

Step 5: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 6: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

5.5 Restart a MongoDB Process

On this page

Overview

If an Ops Manager-managed MongoDB process is not currently running, you can restart it directly from the Ops Manager console.

Considerations

If the Monitoring Agent cannot collect information from a MongoDB process, Ops Manager stops monitoring the process. By default, Ops Manager stops monitoring a `mongos` that is unreachable for 24 hours and a `mongod` that is unreachable for 7 days. Your group might have different default behavior. Ask your system administrator.

Procedure

To restart a process:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click the arrow to the right of the deployment.

Step 3: In the box displaying the deployment configuration, click the gear icon and select *Start Up*.

Step 4: Click *Startup* to confirm.

Step 5: Click *Review & Deploy* to review the configuration.

Ops Manager displays the full configuration for you to review.

Step 6: Click *Confirm & Deploy*.

To view deployment progress, click *View Agent Logs* and select an agent at the top of the *Agent Logs* page. To check for updated entries, refresh the page.

If you diagnose an error and need to correct the deployment configuration, click *Edit Configuration* and then click *Edit Configuration* again. Reconfigure the deployment through the deployment arrow button or through the *Add* button. If you cannot find a solution, *shut down* the deployment. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

5.6 Shut Down MongoDB Processes

On this page

Overview

You can shut down selected `mongod` and `mongos` processes, or you can shut down all the processes at once for a sharded cluster or replica set. Ops Manager continues to monitor the processes, and you can later *restart* them. If you no longer want Ops Manager to monitor the processes, you can *remove them from monitoring*.

Procedure

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click the arrow to the right of the deployment.

Step 3: Click the gear icon drop-down list and select *Shut Down*.

Step 4: Click *Shutdown* to confirm.

Step 5: Click *Review & Deploy*.

Step 6: Click *Confirm & Deploy*.

Additional Information

To restart processes after shutting them down, see *Restart a MongoDB Process*.

To remove processes from Ops Manager monitoring, see *Remove Processes from Monitoring*.

5.7 Remove Processes from Monitoring

On this page

Overview

You can remove `mongod` or `mongos` processes from Ops Manager monitoring by "unmanaging" them. Unmanaging a process removes it from the *Deployment* page and from management by Ops Manager but does not shut it down. When you unmanage a `mongod` or `mongos` process, it continues to run. It simply isn't managed by Ops Manager anymore.

Considerations

If you intend to shut down a process and intend to do so through Ops Manager, do that first, before unmanaging the process. See *Shut Down MongoDB Processes*.

Instead of removing processes from monitoring, you can optionally disable their alerts, which allows you to continue to view the processes in the *Deployment* page. See *Manage Host Alerts*.

Procedure

To unmanage a process:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *edit mode*, and then click the arrow to the right of the deployment.

Step 3: In the box displaying the deployment configuration, click the gear icon and select *Unmanage*.

Step 4: Click *Review & Deploy*.

Step 5: Click *Confirm & Deploy*.

5.8 Alerts

Manage Host Alerts Procedure to enable/disable alerts for hosts.

Create an Alert Configuration Procedures to create alert configurations.

Manage Alert Configuration Procedures for managing alert configurations.

Manage Alerts Procedures for managing alerts.

Alert Conditions Identifies all available alert triggers and conditions.

Manage Host Alerts

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: On the line listing the process, click the gear icon and select *Edit Host*.

To narrow or expand the list of processes, click the drop-down box above the list and select the type of processes to display.

Step 3: Select *Alert Status* and then modify the alert settings.

Create an Alert Configuration

On this page

Overview

An alert configuration defines the conditions that trigger an alert and defines the notifications to be sent.

You can *create an alert configuration from scratch* or *clone it from an existing alert*.

Considerations

Costs

Costs to send alerts depend on your telephone service contract. Many factors may affect alert delivery, including do not call lists, caps for messages sent or delivered, delivery time of day, and message caching.

Alert Intervals

To implement alert escalation, you can create multiple alert configurations with different minimum frequencies. Ops Manager processes alerts on a 5-minute interval. Therefore, the minimum frequency for an alert is 5 minutes. The time between re-notifications increases by the frequency amount every alert cycle (e.g. 5 minutes, 10 minutes, 15 minutes, 20 minutes, etc.) up to a maximum of 24 hours. The default frequency for a new alert configuration is 60 minutes.

When an alert state triggers, you can set a time to elapse before Ops Manager will send alert messages at the specified interval. This helps eliminate false positives. Type in the *after waiting* field the number of minutes to wait before sending the alert at the specified interval for each recipient.

Procedures

You can create a new alert configuration or clone an existing one. This section provides both procedures.

Create an Alert Configuration

Step 1: Select the *Activity* tab and then select *Alert Settings*.

Step 2: Click the *Add Alert* button.

Step 3: Select the component to monitor and the condition that triggers the alert.

In *Alert if*, select the target component. If you select `Host`, you must also select the type of host.

Next, select the condition and, if applicable, specify the threshold for the metric. For explanations of alert conditions and metrics, see *Alert Conditions*.

If the options in the *For* section are available, you can optionally filter the alert to apply only to a subset of the monitored targets. *is*, *is not*, *contains*, *does not contain*, *starts with*, and *ends with* use direct string comparison, while *matches* uses regular expressions. These options are available only if the targets are hosts or replica sets.

Step 4: Select the alert recipients and choose how they receive the alerts.

In *Send to*, specify the alert interval and distribution method for each alert recipient. Click *Add* to add more recipients.

To receive an **SMS** alert, a user must have correctly entered their telephone number in their *Account* page on the *Administration* tab. Ops Manager removes all punctuation and letters and only uses the digits for the telephone number.

If you are outside of the United States or Canada, you will need to include '011' and your country code. For instance, for New Zealand (country code 64), you would need to enter '01164', followed by your phone number. Alternately, you can sign up for a Google Voice number, and use that number for your authentication.

For **HipChat** alerts, enter the HipChat room name and API token. Alerts will appear in the HipChat room message stream. See the *Group Settings page* to define default group settings for HipChat.

For **PagerDuty** alerts, enter only the service key. Define escalation rules and alert assignments in PagerDuty. See the *Group Settings page* to define default group settings for PagerDuty.

For **SNMP** alerts, specify the hostname that will receive the v2c trap on standard port `162`. The MIB file for SNMP is available for download here.

Step 5: Click *Save*.

Clone an Alert Configuration

You can create new alert configurations by cloning an existing one then editing it.

Step 1: Select the *Activity* tab and then select *Alert Settings*.

Step 2: Click the *gear icon* to the right of an alert and then select *Clone*.

Step 3: Select the component to monitor and the condition that triggers the alert.

In *Alert if*, select the target component. If you select `Host`, you must also select the type of host.

Next, select the condition and, if applicable, specify the threshold for the metric. For explanations of alert conditions and metrics, see *Alert Conditions*.

If the options in the *For* section are available, you can optionally filter the alert to apply only to a subset of the monitored targets. *is*, *is not*, *contains*, *does not contain*, *starts with*, and *ends with* use direct string comparison, while *matches* uses regular expressions. These options are available only if the targets are hosts or replica sets.

Step 4: Select the alert recipients and choose how they receive the alerts.

In *Send to*, specify the alert interval and distribution method for each alert recipient. Click *Add* to add more recipients.

To receive an **SMS** alert, a user must have correctly entered their telephone number in their *Account* page on the *Administration* tab. Ops Manager removes all punctuation and letters and only uses the digits for the telephone number.

If you are outside of the United States or Canada, you will need to include '011' and your country code. For instance, for New Zealand (country code 64), you would need to enter '01164', followed by your phone number. Alternately, you can sign up for a Google Voice number, and use that number for your authentication.

For **HipChat** alerts, enter the HipChat room name and API token. Alerts will appear in the HipChat room message stream. See the *Group Settings page* to define default group settings for HipChat.

For **PagerDuty** alerts, enter only the service key. Define escalation rules and alert assignments in PagerDuty. See the *Group Settings page* to define default group settings for PagerDuty.

For **SNMP** alerts, specify the hostname that will receive the v2c trap on standard port `162`. The MIB file for SNMP is available for download here.

Step 5: Click *Save*.

Manage Alert Configuration

On this page

Overview

You can manage alert configurations from the *Activity* tab. An alert configuration defines the conditions that trigger an alert and defines the notifications to be sent.

Manage Alert Configurations

View Alert Configurations

To view alert configurations, click the *Activity* tab and then select the *Alert Settings* page.

Alert configurations define the conditions that trigger alerts and the notifications sent when alerts are triggered.

Ops Manager creates the following alert configurations automatically when you create a new group:

- Users awaiting approval to join group
- Host is exposed to the public internet
- User added to group
- Monitoring Agent is down

If you enable backup, Ops Manager creates the following alert configurations for the group if they do not already exist:

- Oplog Behind
- Resync Required
- Cluster Mongos Is Missing

Create or Clone an Alert Configuration

To create or clone an alert configuration, see *Create an Alert Configuration*.

Modify an Alert Configuration

Each alert configuration has a distribution list, a frequency for sending the alert, and a waiting period after an alert state triggers before sending the first alert.

By default, an alert configuration sends alerts at 60-minute intervals. You can modify the interval. The minimum interval is 5 minutes.

Step 1: Select the *Activity* tab and then select *Alert Settings*.

Step 2: Click the *gear icon* to the right of an alert and then select *Edit*.

Step 3: Select the component to monitor and the condition that triggers the alert.

In *Alert if*, select the target component. If you select `Host`, you must also select the type of host.

Next, select the condition and, if applicable, specify the threshold for the metric. For explanations of alert conditions and metrics, see *Alert Conditions*.

If the options in the *For* section are available, you can optionally filter the alert to apply only to a subset of the monitored targets. *is*, *is not*, *contains*, *does not contain*, *starts with*, and *ends with* use direct string comparison, while *matches* uses regular expressions. These options are available only if the targets are hosts or replica sets.

Step 4: Select the alert recipients and choose how they receive the alerts.

In *Send to*, specify the alert interval and distribution method for each alert recipient. Click *Add* to add more recipients.

To receive an **SMS** alert, a user must have correctly entered their telephone number in their *Account* page on the *Administration* tab. Ops Manager removes all punctuation and letters and only uses the digits for the telephone number.

If you are outside of the United States or Canada, you will need to include '011' and your country code. For instance, for New Zealand (country code 64), you would need to enter '01164', followed by your phone number. Alternately, you can sign up for a Google Voice number, and use that number for your authentication.

For **HipChat** alerts, enter the HipChat room name and API token. Alerts will appear in the HipChat room message stream. See the *Group Settings page* to define default group settings for HipChat.

For **PagerDuty** alerts, enter only the service key. Define escalation rules and alert assignments in PagerDuty. See the *Group Settings page* to define default group settings for PagerDuty.

For **SNMP** alerts, specify the hostname that will receive the v2c trap on standard port `162`. The MIB file for SNMP is available for download here.

Step 5: Click *Save*.

Delete an Alert Configuration

Step 1: Select the *Activity* tab and then select *Alert Settings*.

Step 2: Click the *gear icon* to the right of an alert and then select *Delete*.

Step 3: Click *Confirm*.

When you delete an alert configuration that has open alerts associated to it, Ops Manager cancels the open alerts and sends no further notifications. This is true whether users have acknowledged the alerts or not.

Disable or Enable an Alert Configuration

Step 1: Select the *Activity* tab and then select *Alert Settings*.

Step 2: Click the *gear icon* to the right of an alert and then select either *Disable* or *Enable*.

When you disable an alert configuration it remains visible in a *grayed out* state. Ops Manager automatically cancels active alerts related to a disabled alert configuration. You can reactivate disabled alerts.

For example, if you have an alert configured for *Host Down* and you currently have an active alert telling you a host is down, Ops Manager automatically cancels active *Host Down* alerts if you disable the default *Host Down* configuration. Ops Manager will send no further alerts of this type unless the disabled alert is re-enabled.

Manage Alerts

On this page

Overview

You can manage alerts from the *Activity* tab.

When a condition triggers an alert, users receive the alert at regular intervals until the alert is resolved or canceled. Users can mark the alert as acknowledged for a period of time but will again receive notifications when the acknowledgment period ends if the alert condition still exists.

Alerts end when the alert is resolved or canceled. An alert is resolved, also called "closed," when the condition that triggered the alert has been corrected. Ops Manager sends users a notification at the time the alert is resolved.

An alert is canceled if the alert configuration that triggered the alert is deleted or disabled, or if the target of the alert is removed from the system. For example, if you have an open alert for "Host Down" and you delete that host from Ops Manager, then the alert is canceled. When an alert is canceled, Ops Manager does not send a notification and does not record an entry in the activity feed.

Manage Alerts

View Open Alerts

To view open alerts, click the *Activity* tab and then select *All Activity*. The *All Activity* page displays a feed of all events tracked by Ops Manager. If you have open alerts, the page displays them above the feed.

Filter Activity Feed

You can filter the event feed by date.

Step 1: Select the *Activity* tab and then select *All Activity*.

Step 2: Click the gear icon and specify a date range.

Download Activity Feed

You can download the event feed as a CSV file (comma-separated values).

Step 1: Select the *Activity* tab and then select *All Activity*.

Step 2: Click the gear icon and select *Download as CSV File*.

You can download all events or choose to filter the feed before downloading. Ops Manager limits the number of events returned to 10,000.

Acknowledge an Open Alert

Step 1: Select the *Activity* tab.

The *All Activity* page appears.

Step 2: On the line item for the alert, click *Acknowledge*.

Step 3: Select the time period for which to acknowledge the alert.

Ops Manager will send no further alert messages for the period of time you select.

Step 4: Click *Acknowledge*.

After you acknowledge the alert, Ops Manager sends no further notifications to the alert's distribution list until the acknowledgement period has passed or until the alert is resolved. The distribution list receives *no* notification of the acknowledgment.

If the alert condition ends during the acknowledgment period, Ops Manager sends a notification of the resolution. For example, if you acknowledge a host-down alert and the host comes back up during the acknowledgement period, Ops Manager sends you a notification that the host is up.

If you configure an alert with PagerDuty, a third-party incident management service, you can only acknowledge the alert on your PagerDuty dashboard.

Unacknowledge an Acknowledged Alert

Step 1: Select the *Activity* tab.

The *All Activity* page appears.

Step 2: On the line item for the alert, click *Unacknowledge*.

Step 3: Click *Confirm*.

If the alert condition continues to exist, Ops Manager will resend alerts.

View Closed Alerts

To view closed alerts, click the *Activity* tab and then select *Closed Alerts*. The *Closed Alerts* page displays alerts that users have closed explicitly or where the metric has dropped below the threshold of the alert.

Alert Conditions

On this page

Overview

Ops Manager provides configurable alert conditions that you can apply to Ops Manager components, such as hosts, clusters, or agents. This document groups the conditions according to the target components to which they apply.

Select alert conditions when configuring alerts, for more information on configuring alerts, see the *Create an Alert Configuration* and *Manage Alerts* documents.

Host Alerts

The Host Alerts are applicable to MongoDB hosts (i.e. `mongos` and `mongod` instances). and are grouped here according to the category monitored.

Host Status

`is down`
Sends an alert when Ops Manager does not receive a ping from a host for more than 9 minutes. Under normal operation the Monitoring Agent connects to each monitored host about once per minute. Ops Manager will not alert immediately, however, but waits nine minutes in order to minimize false positives, as would occur, for example, during a host restart.

`is recovering`
Sends an alert when a secondary member of a replica set enters the `RECOVERING` state. For information on the `RECOVERING` state, see Replica Set Member States.

`does not have latest version`
This does not apply to Ops Manager.

Sends an alert when the version of MongoDB running on a host is more than two releases behind. For example if the current production version of MongoDB is 2.6.0 and the previous release is 2.4.9 then a host running version 2.4.8 will trigger this alert but a host running 2.4.9 (previous) 2.6.0 (current) or 2.6.1-rc2 (nightly) will not.

Asserts

These alert conditions refer to the metrics found on the host's `asserts` chart. To view the chart, see *Accessing a Host's Statistics*.

`Asserts: Regular is`
Sends an alert if the rate of regular asserts meets the specified threshold.

`Asserts: Warning is`
Sends an alert if the rate of warnings meets the specified threshold.

`Asserts: Msg is`
Sends an alert if the rate of message asserts meets the specified threshold. Message asserts are internal server errors. Stack traces are logged for these.

`Asserts: User is`
Sends an alert if the rate of errors generated by users meets the specified threshold.

Opcounter

These alert conditions refer to the metrics found on the host's `opcounters` chart. To view the chart, see *Accessing a Host's Statistics*.

`Opcounter: Cmd is`
Sends an alert if the rate of commands performed meets the specified threshold.

`Opcounter: Query is`
Sends an alert if the rate of queries meets the specified threshold.

`Opcounter: Update is`
Sends an alert if the rate of updates meets the specified threshold.

Opcounter: Delete is
Sends an alert if the rate of deletes meets the specified threshold.

Opcounter: Insert is
Sends an alert if the rate of inserts meets the specified threshold.

Opcounter: Getmores is
Sends an alert if the rate of getmore (i.e. cursor batch) operations meets the specified threshold. For more information on getmore operations, see the Cursors page in the MongoDB manual.

Opcounter - Repl

These alert conditions apply to hosts that are secondary members of replica sets. The alerts use the metrics found on the host's `opcounters - repl` chart. To view the chart, see *Accessing a Host's Statistics*.

Opcounter: Repl Cmd is
Sends an alert if the rate of replicated commands meets the specified threshold.

Opcounter: Repl Update is
Sends an alert if the rate of replicated updates meets the specified threshold.

Opcounter: Repl Delete is
Sends an alert if the rate of replicated deletes meets the specified threshold.

Opcounter: Repl Insert is
Sends an alert if the rate of replicated inserts meets the specified threshold.

Memory

These alert conditions refer to the metrics found on the host's `memory` and `non-mapped virtual memory` charts. To view the charts, see *Accessing a Host's Statistics*. For additional information about these metrics, click the *i* icon for each chart.

Memory: Resident is
Sends an alert if the size of the resident memory meets the specified threshold. It is typical over time, on a dedicated database server, for the size of the resident memory to approach the amount of physical RAM on the box.

Memory: Virtual is
Sends an alert if the size of virtual memory for the mongod process meets the specified threshold. You can use this alert to flag excessive memory outside of memory mapping. For more information, click the `memory` chart's *i* icon.

Memory: Mapped is
Sends an alert if the size of mapped memory, which maps the data files, meets the specified threshold. As MongoDB memory-maps all the data files, the size of mapped memory is likely to approach total database size.

Memory: Computed is
Sends an alert if the size of virtual memory that is not accounted for by memory-mapping meets the specified threshold. If this number is very high (multiple gigabytes), it indicates that excessive memory is being used outside of memory mapping. For more information on how to use this metric, view the `non-mapped virtual memory` chart and click the chart's *i* icon.

B-tree

These alert conditions refer to the metrics found on the host's `btree` chart. To view the chart, see *Accessing a Host's Statistics*.

`B-tree: accesses is`
Sends an alert if the number of accesses to B-tree indexes meets the specified average.

`B-tree: hits is`
Sends an alert if the number of times a B-tree page was in memory meets the specified average.

`B-tree: misses is`
Sends an alert if the number of times a B-tree page was *not* in memory meets the specified average.

`B-tree: miss ratio is`
Sends an alert if the ratio of misses to hits meets the specified threshold.

Lock %

This alert condition refers to metric found on the host's `lock %` chart. To view the chart, see *Accessing a Host's Statistics*.

`Effective Lock % is`
Sends an alert if the amount of time the host is write locked meets the specified threshold. For details on this metric, view the `lock %` chart and click the chart's *i* icon.

Background

This alert condition refers to metric found on the host's `background flush avg` chart. To view the chart, see *Accessing a Host's Statistics*.

`Background Flush Average is`
Sends an alert if the average time for background flushes meets the specified threshold. For details on this metric, view the `background flush avg` chart and click the chart's *i* icon.

Connections

The following alert condition refers to a metric found on the host's `connections` chart. To view the chart, see *Accessing a Host's Statistics*.

`Connections is`
Sends an alert if the number of active connections to the host meets the specified average.

Queues

These alert conditions refer to the metrics found on the host's `queues` chart. To view the chart, see *Accessing a Host's Statistics*.

`Queues: Total is`
Sends an alert if the number of operations waiting on a lock of any type meets the specified average.

`Queues: Readers is`
Sends an alert if the number of operations waiting on a read lock meets the specified average.

Queues: Writers is
Sends an alert if the number of operations waiting on a write lock meets the specified average.

Page Faults

These alert conditions refer to metrics found on the host's `Record Stats` and `Page Faults` charts. To view the charts, see *Accessing a Host's Statistics*.

Accesses Not In Memory: Total is
Sends an alert if the rate of disk accesses meets the specified threshold. MongoDB must access data on disk if your working set does not fit in memory. This metric is found on the host's `Record Stats` chart.

Page Fault Exceptions Thrown: Total is
Sends an alert if the rate of page fault exceptions thrown meets the specified threshold. This metric is found on the host's `Record Stats` chart.

Page Faults is
Sends an alert if the rate of page faults (whether or not an exception is thrown) meets the specified threshold. This metric is found on the host's `Page Faults` chart.

Cursors

These alert conditions refer to the metrics found on the host's `cursors` chart. To view the chart, see *Accessing a Host's Statistics*.

Cursors: Open is
Sends an alert if the number of cursors the server is maintaining for clients meets the specified average.

Cursors: Timed Out is
Sends an alert if the number of timed-out cursors the server is maintaining for clients meets the specified average.

Cursors: Client Cursors Size is
Sends an alert if the cumulative size of the cursors the server is maintaining for clients meets the specified average.

Network

These alert conditions refer to the metrics found on the host's `network` chart. To view the chart, see *Accessing a Host's Statistics*.

Network: Bytes In is
Sends an alert if the number of bytes sent *to* the database server meets the specified threshold.

Network: Bytes Out is
Sends an alert if the number of bytes sent *from* the database server meets the specified threshold.

Network: Num Requests is
Sends an alert if the number of requests sent to the database server meets the specified average.

Replication

These alert conditions refer to the metrics found on a primary's `replication oplog window` chart or a secondary's `replication lag` chart. To view the charts, see *Accessing a Host's Statistics*.

Replication Oplog Window is
Sends an alert if the approximate amount of time available in the primary's replication oplog meets the specified threshold.

Replication Lag is
Sends an alert if the approximate amount of time that the secondary is behind the primary meets the specified threshold.

Replication Headroom is
Sends an alert when the difference between the primary oplog window and the replication lag time on a secondary meets the specified threshold.

Oplog Data per Hour is
Sends an alert when the amount of data per hour being written to a primary's oplog meets the specified threshold.

DB Storage

This alert condition refers to the metric displayed on the host's `db storage` chart. To view the chart, see *Accessing a Host's Statistics*.

DB Storage is
Sends an alert if the amount of on-disk storage space used by extents meets the specified threshold. Extents are contiguously allocated chunks of datafile space.

DB storage size is larger than DB data size because storage size measures the entirety of each extent, including space not used by documents. For more information on extents, see the collStats command.

DB Data Size is
Sends an alert if approximate size of all documents (and their paddings) meets the specified threshold.

Journaling

These alert conditions refer to the metrics found on the host's `journal - commits in write lock` chart and `journal stats` chart. To view the charts, see *Accessing a Host's Statistics*.

Journaling Commits in Write Lock is
Sends an alert if the rate of commits that occurred while the database was in write lock meets the specified average.

Journaling MB is
Sends an alert if the average amount of data written to the recovery log meets the specified threshold.

Journaling Write Data Files MB is
Sends an alert if the average amount of data written to the data files meets the specified threshold.

Replica Set Alerts

These alert conditions are applicable to replica sets.

Primary Elected
Sends an alert when a set elects a new primary. Each time Ops Manager receives a ping, it inspects the output of the replica set's rs.status() method for the status of each replica set member. From this output, Ops Manager determines which replica set member is the primary. If the primary found in the ping data is different than the current primary known to Ops Manager, this alert triggers.

`Primary Elected` does not always mean that the set elected a *new* primary. `Primary Elected` may also trigger when the same primary is re-elected. This can happen when Ops Manager processes a ping in the midst of an election.

No Primary
Sends an alert when a replica set does not have a primary. Specifically, when none of the members of a replica set have a status of `PRIMARY`, the alert triggers. For example, this condition may arise when a set has an even number of voting members resulting in a tie.

If the Monitoring Agent collects data during an election for primary, this alert might send a false positive. To prevent such false positives, set the alert configuration's *after waiting* interval (in the configuration's *Send to* section).

Number of Healthy Members is below
Sends an alert when a replica set has fewer than the specified number of healthy members. If the replica set has the specified number of healthy members or more, Ops Manager triggers no alert.

A replica set member is healthy if its state, as reported in the rs.status() output, is either `PRIMARY` or `SECONDARY`. Hidden secondaries and arbiters are not counted.

As an example, if you have a replica set with one member in the `PRIMARY` state, two members in the `SECONDARY` state, one hidden member in the `SECONDARY`, one `ARBITER`, and one member in the `RECOVERING` state, then the healthy count is 3.

Number of Unhealthy Members is above
Sends an alert when a replica set has more than the specified number of unhealthy members. If the replica set has the specified number or fewer, Ops Manager sends no alert.

Replica set members are unhealthy when the agent cannot connect to them, or the member is in a rollback or recovering state.

Hidden secondaries are not counted.

Agent Alerts

These alert conditions are applicable to Monitoring Agents and Backup Agents.

Monitoring Agent is down
Sends an alert if the Monitoring Agent has been down for at least 7 minutes. Under normal operation, the Monitoring Agent sends a ping to Ops Manager roughly once per minute. If Ops Manager does not receive a ping for at least 7 minutes, this alert triggers. However, this alert will never trigger for a group that has no hosts configured.

Important: When the Monitoring Agent is down, Ops Manager will trigger no other alerts. For example, if a host is down there is no Monitoring Agent to send data to Ops Manager that could trigger new alerts.

Backup Agent is down
Sends an alert if the Backup Agent has been down for at least 15 minutes. Under normal operation, the Backup Agent periodically sends data to Ops Manager. This alert is never triggered for a group that has no running backups.

Monitoring Agent is out of date
Sends an alert when the Monitoring Agent is not running the latest version of the software.

Backup Agent is out of date
Sends an alert when the Backup Agent is not running the latest version of the software.

Backup Alerts

These alert conditions are applicable to the Ops Manager Backup service.

`Oplog Behind`
Sends an alert if the most recent oplog data received by Ops Manager is more than 75 minutes old.

`Resync Required`
Sends an alert if the replication process for a backup falls too far behind the oplog to catch up. This occurs when the host overwrites oplog entries that backup has not yet replicated. When this happens, backup must be fully resynced.

`Cluster Mongos Is Missing`
Sends an alert if Ops Manager cannot reach a `mongos` for the cluster.

User Alerts

These alert conditions are applicable to the Ops Manager Users.

`Added to Group`
Sends an alert when a new user joins the group.

`Removed from Group`
Sends an alert when a user leaves the group.

`Changed Roles`
Sends an alert when a user's roles have been changed.

Group Alert

This alert condition applies to Ops Manager groups.

`Users awaiting approval to join group`
Sends an alert if there are users who have asked to join the group. A user can ask to join a group when first registering for Ops Manager.

5.9 Monitoring Metrics

Deployment Description of the *Deployment* tab, which lists all hosts that are currently being monitored.

Host Statistics In-depth guide to host statistics and the options that you can specify to customize your view.

Aggregated Cluster Statistics Compare hosts dynamically across the cluster.

Replica Set Statistics Compare hosts dynamically across a replica set.

Profile Databases Collect profile data for the host.

Deployment

On this page

- *Host Mapping Page*

Deployment provides access to all your monitored objects. The *Deployment* tab includes the pages described here.

Deployment Page

The *Deployment* page provides access to all monitored `mongod` and `mongos` instances. The page includes the following information:

Field	Description
Last Ping	The last time this agent sent a ping to the Ops Manager servers. Click a ping to view a detailed status from the ping.
Host	The hostname and port of the instance. Click the hostname to view *host statistics*.
Orange triangle icon under a host name.	The startup warning indicator for the host. Only displayed when warnings exist. Click the host's last ping for warning details. Ops Manager startup warnings can include the following: • Ops Manager suspects the host has a low `ulimit` setting of less than `1024`. Ops Manager infers the host's `ulimit` setting using the total number of available and current connections. See the UNIX ulimit Settings reference page. • Ops Manager flags a deactivated host. **Important:** If you have deactivated hosts, review all deactivated hosts to ensure that they are still in use, and remove all hosts that are not active. Then click on the warning icon and select *Reactive ALL hosts*.
Type	The type of host. Possible types include the following: • `PRIMARY` • `SECONDARY` • `STANDALONE` • `ARBITER` When the host recovers, the rectangle flag turns yellow and displays `RECOVERING`. When the host returns a fatal error, the flag displays `FATAL`. The flag also can display `NO DATA`.
Cluster	The name of the cluster to which this instance belongs. Only cluster members display this value. Click the cluster name to display aggregated information on the cluster's replica sets. See *Aggregated Cluster Statistics* for details.
Shard	The name of the shard.
Repl Set	The name of the shard's replica set. Click the replica set name to display replica set statistics. See *Replica Set Statistics* for details.
Up Since	The date the host first pinged Ops Manager.
Version	The version of the MongoDB running on this instance.

Host Mapping Page

The *Host Mapping* page shows the mapping between system hostnames and the names provided by the monitored `mongod` and `mongos` processes.

Host Statistics

On this page

For each host, Ops Manager provides an extensive set of charts for analyzing the statistics collected by the Monitoring Agent.

Accessing a Host's Statistics

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the name of the cluster, replica set or process for which to view statistics.

Step 4: Hover the mouse pointer over a chart to display chart controls.

To use the controls, see *Accessing Information on a Host's Chart*.

Accessing Information on a Host's Chart

Hover the mouse cursor over the chart to display the chart controls.

- Click the *i* icon for a description of the chart.
- Click-and-drag to select a portion of the chart to zoom into. All charts on the page zoom to the same level.
- Double-click to revert the charts back to the default zoom setting.
- Hover the mouse pointer over a point on the chart to display statistics for that point in time.
- Click the two-way arrow to open an expanded version of the chart.
- Click the curved arrow for a list of additional actions:
  - *Chart Permalink* opens a page that displays only this chart.
  - *Email Chart* opens a dialogue box where you can input an email address and short message to send the chart by email.
- Click and hold the upper-left triangular grabber to move the chart to a different place on the page.

Chart Annotations

Annotations may appear as colored vertical lines on your charts to indicate server events. The following color/event combinations are:

- A *red bar* indicates a server restart.
- A *purple bar* indicates the server is now a primary.
- A *yellow bar* indicates the server is now a secondary.

If you do not wish to see the chart annotations, you can disable them on the *Administration* tab's *Personalization* page.

Aggregated Cluster Statistics

On this page

Overview

Cluster statistics provide an interface to view data for an entire cluster at once. You can compare components dynamically across the cluster and view host-specific and aggregated data, as well as pinpoint moments in time and isolate data to specific components.

Procedure

To view cluster statistics:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the name of the sharded cluster.

Ops Manager displays a chart and table with an initial set of cluster statistics. At the top of the chart, the *DATA SIZE* field measures the cluster's data size on disk. For more information, see the explanation of dataSize on the dbStats page.

If *Backup* is enabled, hover the mouse pointer over the "clock" icon to view the time of the last snapshot and time of the next scheduled snapshot. Click the icon to view snapshots.

Step 4: Select the components to display.

In the buttons above the chart, select whether to display the cluster's shards, mongos instances, or config servers.

If you select shards, select whether to display *Primaries*, *Secondaries*, or *Both* using the buttons at the chart's lower right.

The chart displays a different colored line for each component. The table below displays additional data for each component, using the same colors.

Step 5: Select the data to display.

Select the type of data in the *CHART* drop-down list. Ops Manager graphs the data for each component individually. You can instead graph the data as an average or sum by clicking the *Averaged* or *Sum* button at the chart's lower right.

Step 6: Change the granularity and zoom.

To the right of the chart, select a *GRANULARITY* for the data. The option you select determines the available *ZOOM* options. Whenever you change the granularity, the selected zoom level changes to the closest zoom level available for that granularity.

To zoom further and isolate a specific region of data, click-and-drag on that region of the chart. To reset the zoom level, double-click anywhere on the chart.

Step 7: View metrics for a specific date and time.

Move the mouse pointer over the chart to view data for a point in time. The data in the table below the chart changes as you move the pointer.

Step 8: Isolate certain components for display.

To remove a component from the chart, click its checkmark in the table below the chart. To again display it, click the checkmark again.

To quickly isolate just a few components from a large number displayed, select the *None* button below the chart and then select the checkmark for the individual components to display. Alternatively, select the *All* button and then deselect the checkmark for individual components not to display.

Step 9: View statistics for a specific component.

In the table below the chart, click a component's name to display its statistics page.

If your are viewing shards, you can click the replica set name in the *SHARDS* column to display *replica set statistics*, or you can click the *P* or *S* icon in the *MEMBERS* column to display *host statistics* for a primary or secondary. Hover over an icon for tooltip information.

Step 10: Change the name of the cluster.

If you want to change the name of the cluster, hover the mouse pointer over the cluster name. A pencil icon appears. Click the icon and enter the new name.

Replica Set Statistics

On this page

Overview

The Replica set statistics interface makes is possible to view data from all replica set members at once.

Procedure

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the name of the replica set.

Ops Manager displays a separate chart for each replica set member.

Step 4: Select the members to display.

In the *TOGGLE MEMBERS* section at the top of the page, click the *P* and *S* icons to choose which members to display. Hover the mouse pointer over an icon to display member information.

Step 5: Select the granularity and the zoom.

Select the *GRANULARITY* of the data displayed. The selected granularity option determines the available *ZOOM* options.

To isolate a specific region of data, click-and-drag on that region of the chart. All other charts automatically zoom to the same region.

To reset the zoom level, double-click anywhere on the chart.

Step 6: Add, remove, and reorder charts.

Add and remove charts using either the *Add Chart* drop-down list or the buttons at bottom of the page.

Move a chart within the display by hovering the mouse over the chart, clicking the grabber in the upper left corner, and dragging the chart to the new position.

Step 7: View an explanation of a chart's data.

Hover the mouse pointer over the and click the *i* icon.

Step 8: View metrics for a specific date and time.

Move the mouse pointer over the chart to view data for a point in time.

Profile Databases

On this page

- *Procedures*

Overview

Monitoring can collect data from MongoDB's profiler to provide statistics about performance and database operations.

Considerations

Before enabling profiling, be aware of these issues:

- Profile data can include sensitive information, including the content of database queries. Ensure that exposing this data to Monitoring is consistent with your information security practices.
- The profiler can consume resources which may adversely affect MongoDB performance. Consider the implications before enabling profiling.

Procedures

Enable Profiling

To allow Monitoring to collect profile data for a specific process:

Note: The Monitoring Agent attempts to minimize its effect on the monitored systems. If resource intensive operations, like polling profile data, begins to impact the performance of the database, Monitoring will throttle the frequency that it collects data. See *How does Ops Manager gather database statistics?* for more information about the agent's throttling process.

When enabled, Monitoring samples profiling data from monitored processes. The agent sends only the most recent 20 entries from last minute.

With profiling enabled, configuration changes made in Ops Manager can take up to 2 minutes to propagate to the agent and another minute before profiling data appears in the Ops Manager interface.

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*, and then click the process's gear icon and select *Edit Host*.

Step 3: Click the *Profiling* tab.

Step 4: Turn on profiling.

Click the button to toggle between *Off* and *On*. When the button is *On*, Ops Manager receives database profile statistics.

Step 5: Start database profiling by using the mongo shell to modify the `setProfilingLevel` command.

See the database profiler documentation for instructions for using the profiler.

Display Profiling Levels

When profiling is on, the *Profile Data* tab displays profiled data. For more information on profiling, see the database profiler documentation in the MongoDB manual.

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Select the *Profile Data* tab.

Delete Profile Data

Deleting profile data deletes the Web UI cache of the current profiling data. You must then disable profiling or drop or clear the source collection, or Ops Manager will repopulate the profiling data.

If Monitoring is storing a large amount of profile data for your instance, the removal process will not be instantaneous.

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*, and then click the process.

Step 3: Select the *Profile Data* tab.

Step 4: Click the *Delete Profile Data* button at the bottom of the page.

Step 5: Confirm the deletion.

Ops Manager begins removing stored profile data from the server's record. Ops Manager removes **only** the Web UI cache of the current profiling data. The cache quickly re-populates with the same data if you do not disable profiling or drop or clear the profiled collection.

5.10 View Logs

On this page

Overview

Ops Manager collects log information for both MongoDB and the Ops Manager agents. For MongoDB deployments, Ops Manager provides access to both real-time logs and on-disk logs.

The MongoDB logs provide the diagnostic logging information for your mongod and mongos instances. The Agent logs provide insight into the behavior of your Ops Manager agents.

MongoDB Real-Time Logs

The Monitoring Agent collects real-time log information from each MongoDB deployment by issuing the getLog command with every monitoring ping. The getLog command collects log entries from the MongoDB RAM cache.

Ops Manager enables real-time log collection by default. You can disable log collection for either the whole Ops Manager group or for individual MongoDB instances. If you disable log collection, Ops Manager continues to display previously collected log entries.

View MongoDB Real-Time Logs

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the process.

Step 4: Click the *Logs* tab.

The tab displays log information. If the tab instead diplays the *Collect Logs For Host* option, toggle the option to *On* and refresh the page.

Step 5: Refresh the browser window to view updated entries.

Enable or Disable Log Collection for a Deployment

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Click *view mode*.

Step 3: Click the deployment's gear icon and select *Edit Host*.

Step 4: Click the *Logs* tab and toggle the *Off*/*On* button as desired.

Step 5: Click x to close the *Edit Host* box.

The deployment's previously existing log entries will continue to appear in the *Logs* tab, but Ops Manager will not collect new entries.

Enable or Disable Log Collection for the Group

Step 1: Select the *Administration* tab, then the *Group Settings* page.

Step 2: Set the *Collect Logs For All Hosts* option to *On* or *Off*, as desired.

MongoDB On-Disk Logs

Ops Manager can collect on-disk logs even if the MongoDB instance is not running. The Automation Agent collects the logs from the location specified by the MongoDB `systemLog.path` configuration option. The MongoDB on-disk logs are a subset of the real-time logs and therefore less verbose.

You can configure log rotation for the on-disk logs. Ops Manager enables log rotation by default.

View MongoDB On-Disk Logs

Step 1: Select the *Deployment* tab and then *Mongo Logs* page.

Alternatively, you can select the *Deployment* page's *edit mode*, then the arrow to the right of a deployment, then the gear icon drop-down list, and then *Request Logs*.

Step 2: Request the latest logs.

To request the latest logs:

1. Click the *Manage* drop-down button and select *Request Server Logs*.
2. Select the checkboxes for the logs you want to request, then click *Request Logs*.

Step 3: To view a log, select the *Show Log* link for the desired date and hostname.

Configure Log Rotation

Step 1: Select the *Deployment* tab and then *Mongo Logs* page.

Step 2: Click the *Manage* drop-down button and select *MongoDB Log Settings*.

Step 3: Configure the log rotation settings and click *Save*.

Step 4: Click *Review & Deploy*.

Step 5: Click *Confirm & Deploy*.

Agent Logs

Ops Manager collects logs for all your Automation Agents, Monitoring Agents, and Backup Agents.

View Agent Logs

Step 1: From any page, click an agent icon at the top of the page and select *Logs*.

Ops Manager opens the *Agent Logs* page and displays the log entries for agents of the same type.

You can also open the *Agent Logs* page by selecting the *Administration* tab, then *Agents* page, and then *view logs* link for a particular agent. The page displays the agent's log entries.

Step 2: Filter the log entries.

Use the drop-down list at the top of the page to display different types of agents.

Use the gear icon to the right of the page to clear filters and to export logs.

Configure Agent Log Rotation

Step 1: Select the *Administration* tab and then *Agents* page.

Step 2: Edit the log settings.

Under the *Agent Log Settings* header, click the *pencil* icon to edit the log settings for the appropriate agent.

You can modify the following fields:

Name	Type	Description
Linux Log Path	string	The path to which the agent writes its logs.
Rotate Logs	boolean	Specifies whether Ops Manager should rotate the logs for the agent.
Size Threshold (MB)	number	Max size in MB for an individual log file before rotation.
Time Threshold (Hours)	integer	Max time in hours for an individual log file before rotation.
Max Uncompressed Files	integer	*Optional* Max number of total log files to leave uncompressed, including the current log file.
Max Percent of Disk	number	*Optional* Max percent of the total disk space all log files can occupy before deletion.

When you are done modifying the agent log settings, click *Confirm*.

Step 3: Return to the *Deployment* page

Step 4: Click *Review & Deploy*.

Step 5: Click *Confirm & Deploy*.

6 Back Up MongoDB Deployments

Backup Flows Describes how Ops Manager backs up MongoDB deployments.

Backup Preparations Before backing up your cluster or replica set, decide how to back up the data and what data to back up.

Activate Backup Activate Backup for a cluster or replica set.

Edit a Backup's Settings Modify a backup's schedule, storage engines, and excluded namespaces.

Restore MongoDB Deployments Procedures to restore complete MongoDB deployments using Backup data.

Restore MongoDB Data Procedures to restore data from Backup to MongoDB instances.

Backup Maintenance Procedures to manage backup operations for maintenance.

6.1 Backup Flows

On this page

Introduction

The Backup service's process for keeping a backup in sync with your deployment is analogous to the process used by a secondary to replicate data in a replica set. Backup first performs an initial sync to catch up with your deployment and then tails the oplog to stay caught up. Backup takes scheduled snapshots to keep a history of the data.

Initial Sync

Transfer of Data and Oplog Entries

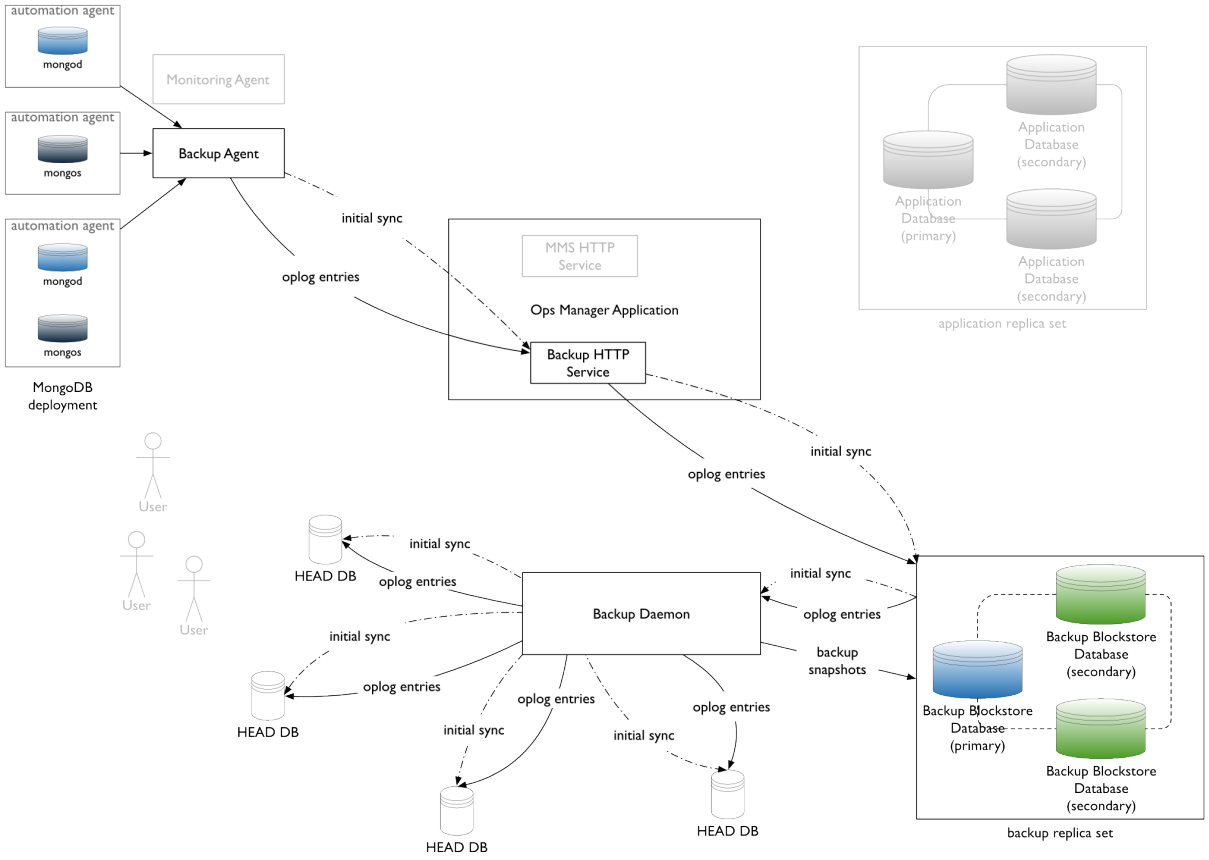
When you start a backup, the Backup Agent streams your deployment's existing data to the Backup HTTP Service in batches of documents totaling roughly 10MB. The batches are called "slices." The Backup HTTP Service stores the slices in a sync store for later processing. The sync store contains only the data as it existed when you started the backup.

While transferring the data, the Backup Agent also tails the oplog and also streams the oplog updates to the Backup HTTP Service. The service places the entries in the oplog store for later processing offline.

By default, both the sync store and oplog store reside on the backing MongoDB replica set that hosts the Backup Blockstore database.

Building the Backup

When the Backup HTTP Service has received all of the slices, a Backup Daemon creates a local database on its server and inserts the documents that were captured as slices during the initial sync. The daemon then applies the oplog entries from the oplog store.



The Backup Daemon then validates the data. If there are missing documents, Ops Manager queries the deployment for the documents and the Backup Daemon inserts them. A missing document could occur because of an update that caused a document to move during the initial sync.

Once the Backup Daemon validates the accuracy of the data directory, it removes the data slices from the sync store. At this point, Backup has completed the initial sync process and proceeds to routine operation.

Routine Operation

The Backup Agent tails the deployment's oplog and routinely batches and transfers new oplog entries to the Backup HTTP Service, which stores them in the oplog store. The Backup Daemon applies all newly received oplog entries in batches to its local replica of the backed-up deployment.

Snapshots

During a preset interval, the Backup Daemon takes a snapshot of the data directory for the backed-up deployment, breaks it into blocks, and transfers the blocks to the Backup Blockstore database. For a sharded cluster, the daemon takes a snapshot of each shard and of the config servers. The daemon uses *checkpoints* to synchronize the shards and config servers for the snapshots.

When a user requests a snapshot, a Backup Daemon retrieves the data from the Backup Blockstore database and delivers it to the requested destination. See: *Restore Flows* for an overview of the restore process.

Grooms

Groom jobs perform periodic "garbage collection" on the Backup Blockstore database to remove unused blocks and reclaim space. Unused blocks are those that are no longer referenced by a live snapshot. A scheduling process determines when grooms are necessary.

6.2 Backup Preparations

On this page

Overview

Before backing up your cluster or replica set, decide how to back up the data and what data to back up. This page describes items you must consider before starting a backup.

For an overview of how Backup works, see *Backup*.

Snapshot Frequency and Retention Policy

By default, Ops Manager takes a base snapshot of your data every 6 hours. If desired, administrators can change the frequency of base snapshots to 8, 12, or 24 hours. Ops Manager creates snapshots automatically on a schedule. You cannot take snapshots on demand.

Ops Manager retains snapshots for the time periods listed in the following table. If you terminate a backup, Ops Manager immediately deletes the backup's snapshots.

Snapshot	Default Retention Policy	Maximum Retention Setting
Base snapshot.	2 days	5 days.
Daily snapshot	1 week	1 year
Weekly snapshot	1 month	1 year
Monthly snapshot	1 year	3 years.

You can change a backed-up deployment's schedule through its *Edit Snapshot Schedule* menu option, available through the *Backup* tab. Administrators can change snapshot frequency and retention through the *snapshotSchedule resource in the API*. If you change the schedule to save fewer snapshots, Ops Manager does **not** delete existing snapshots to conform to the new schedule. To delete unneeded snapshots, see *Delete Snapshots for Replica Sets and Sharded Clusters*.

Excluded Namespaces

Excluded namespaces are databases or collections that Ops Manager will not back up. Exclude namespaces to prevent backing up collections that contain logging data, caches, or other ephemeral data. Excluding these kinds of databases and collections will allow you to reduce backup time and costs.

Storage Engine

When you enable backups for a cluster or replica set that runs on MongoDB 3.0 or higher, you can choose the storage engine for the backups. Your choices are the MMAPv1 engine or WiredTiger engine. If you do not specify a storage engine, Ops Manager uses MMAPv1 by default. For more information on storage engines, see Storage in the MongoDB manual.

You can choose a different storage engine for a backup than you do for the original data. There is no requirement that the storage engine for a backup match that of the data it replicates. If your original data uses MMAPv1, you can choose WiredTiger for backing up, and vice versa.

You can change the storage engine for a cluster or replica set's backups at any time, but doing so requires an initial sync of the backup on the new engine.

If you choose the WiredTiger engine to back up a collection that already uses WiredTiger, the initial sync replicates all the collection's WiredTiger options. For information on these options, see the `storage.wiredTiger.collectionConfig` section of the Configuration File Options page in the MongoDB manual.

For collections created after initial sync, the Backup Daemon uses its own defaults for storing data. The Daemon will not replicate any WiredTiger options for a collection created after iniitial sync.

Important: The storage engine chosen for a backup is *independent* from the storage engine used by the Backup Database. If the *Backup Database* uses the MMAPv1 storage engine, it can store backup snapshots for WiredTiger backup jobs in its blockstore.

Index collection options are never replicated.

Resyncing Production Deployments

For production deployments, it is recommended that as a best practice you periodically (annually) *resync* all backed-up replica sets. When you resync, data is read from a secondary in each replica set. During resync, no new snapshots are generated.

Checkpoints

For sharded clusters, checkpoints provide additional restore points between snapshots. With checkpoints enabled, Ops Manager Backup creates restoration points at configurable intervals of every 15, 30 or 60 minutes between snapshots.

To create a checkpoint, Ops Manager Backup stops the balancer and inserts a token into the oplog of each shard and config server in the cluster. These checkpoint tokens are lightweight and do not have a consequential impact on performance or disk use.

Backup does not require checkpoints, and they are disabled by default.

Restoring from a checkpoint requires Ops Manager Backup to apply the oplog of each shard and config server to the last snapshot captured before the checkpoint. Restoration from a checkpoint takes longer than restoration from a snapshot.

Snapshots when Agent Cannot Stop Balancer

For sharded clusters, Ops Manager disables the balancer before taking a cluster snapshot. In certain situations, such as a long migration or no running `mongos`, Ops Manager tries to disable the balancer but cannot. In such cases, Ops Manager will continue to take cluster snapshots but will flag the snapshots with a warning that data may be incomplete and/or inconsistent. Cluster snapshots taken during an active balancing operation run the risk of data loss or orphaned data.

Snapshots when Agent Cannot Contact a `mongod`

For sharded clusters, if the Backup Agent cannot reach a `mongod` instance, whether a shard or config server, then the agent cannot insert a synchronization oplog token. If this happens, Ops Manager will not create the snapshot and will display a warning message.

6.3 Activate Backup

On this page

Overview

You can back up a sharded cluster or replica set. To back up a standalone `mongod` process, you must first convert it to a single-member replica set. You can choose to back up all databases and collections on the deployment or specific ones.

Prerequisites

- Ops Manager must be monitoring the deployment. For a sharded cluster, Ops Manager must also be monitoring at least one `mongos` in the cluster.
- A replica set must be MongoDB version 2.2.0 or later.
- A sharded-cluster must be MongoDB version 2.4.3 or later.
- Each replica set must have an active primary.
- For a sharded cluster, all config servers must be running and the balancing round must have completed within the last hour.
- If you explicitly select a sync target, ensure that the sync target is accessible on the network and keeping up with replication.

Procedure

Before using this procedure, see the *Backup Preparations* to decide how to back up the data and what data to back up.

Step 1: Select the *Backup* tab.

Step 2: Select the replica set or cluster to back up.

If you have not yet enabled Backup, select *Begin Setup* and follow the prompts. Skip the rest of the this procedure.

If you have already enabled Backup, navigate to either the *Sharded Cluster Status* or *Replica Set Status* page. Then click the *Start* button for the replica set or cluster to back up.

Step 3: Select which process to use for the initial *Sync Source*.

To minimize the impact on the primary, sync off a secondary.

Step 4: If using access control, specify mechanism and credentials, as needed.

Auth Mechanism	The authentication mechanism used by the host. Can specify *MONGODB-CR*, *LDAP (PLAIN)*, or *Kerberos(GSSAPI)*.
Current DB Username	If the authentication mechanism is MONGODB-CR or LDAP, the username used to authenticate the Monitoring Agent to the MongoDB deployment. See *Configure Backup Agent for MONGODB-CR*, *Configure Backup Agent for LDAP Authentication*, or *Configure the Backup Agent for Kerberos* for setting up user credentials.
Current DB Password	If the authentication mechanism is MONGODB-CR or LDAP, the password used to authenticate the Monitoring Agent to the MongoDB deployment. See *Configure Backup Agent for MONGODB-CR*, *Configure Backup Agent for LDAP Authentication*, or *Configure the Backup Agent for Kerberos* for setting up user credentials.
My deployment supports SSL for MongoDB connections	If checked, the Monitoring Agent must have a trusted CA certificate in order to connect to the MongoDB instances. See *Configure Monitoring Agent for SSL*.

You can optionally configure authentication credentials later through the deployment's gear icon.

Step 5: To optionally select a storage engine or exclude namespaces, click *Show Advanced Options*.

Select the following as desired:

Storage Engine: Select *MongoDB Memory Mapped* for the MongoDB default MMAPv1 engine or *WiredTiger* for the 64-bit WiredTiger engine available beginning with MongoDB 3.0. Before selecting a storage engine, see the considerations in *Storage Engines*.

Manage Excluded Namespaces: Click *Manage Excluded Namespaces* and enter the databases and collections to exclude. For colletions, enter the full namespace: `<database>.<collection>`. Click *Save*. You can later add or remove namespaces from the backup, as needed. For more information, see *Excluded Namespaces*.

Step 6: Click *Start Backup*.

6.4 Edit a Backup's Settings

On this page

Overview

You can modify a backup's *schedule*, *excluded namespaces*, and *storage engine*.

Procedure

To edit Backup's settings, select the *Backup* tab and then the *Overview* page. The *Overview* page lists all available backups. You can then access the settings for each backup using the ellipsis icon.

Enable Cluster Checkpoints

Step 1: Select *Edit Snapshot Schedule*.

On the line listing the process, click the ellipses icon and click *Edit Snapshot Schedule*.

Step 2: Enable cluster checkpoints.

Select *Create cluster checkpoint every* and set the interval. Then click *Submit*.

Change Snapshot Settings

Step 1: Select *Edit Snapshot Schedule*

On the line listing the process, click the ellipses icon and click *Edit Snapshot Schedule*.

Step 2: Configure the Snapshot

Enter the following information as needed and click *Submit*.

Take snapshots every . . . and save for	Sets how often Ops Manager takes a base snapshot of the deployment and how long Ops Manager retains base snapshots. For information on how these settings affect Ops Manager, see *Snapshot Frequency and Retention Policy*.
Create cluster checkpoint every (Sharded Clusters only)	Sets how often Ops Manager creates a *checkpoint* in between snapshots of a sharded cluster. Checkpoints provide restore points that you can use to create custom "point in time" snapshots. For more information, see *Checkpoints*.
Store daily snapshots for	Sets the time period that Ops Manager retains daily snapshots. For defaults, see *Snapshot Frequency and Retention Policy*.
Store weekly snapshots for	Sets the time period that Ops Manager retains weekly snapshots. For defaults, see *Snapshot Frequency and Retention Policy*.
Store monthly snapshots for	Sets the time period that Ops Manager retains monthly snapshots. For defaults, see *Snapshot Frequency and Retention Policy*.

Configure Excluded Namespaces

Step 1: Select *Edit Excluded Namespaces*

On the line listing the process, click the ellipses icon and click *Edit Excluded Namespaces*. *Excluded namespaces* are databases and collections that Ops Manager will not back up.

Step 2: Modify the excluded namespaces.

Add or remove excluded namespaces as desired and click *Submit*.

Modify the Storage Engine Used for Backups

Step 1: Select *Edit Storage Engine*

On the line listing the process, click the ellipses icon and click *Edit Storage Engine*.

Step 2: Select the storage engine.

Select the storage engine. See: *Storage Engine* for more about choosing an appropriate storage engine for your backup.

Step 3: Select the sync source.

Select the *Sync source* from which to create the new backup. In order to use the new storage engine, Ops Manager must resync the backup on the new storage engine.

Step 4: Click *Submit*.

6.5 Restore MongoDB Deployments

Use these procedures to restore an entire MongoDB deployment using Backup artifacts. For more specific tutorials for restoration, please see the *Restore MongoDB Instances with Backup* procedures.

Restore Flows Overview of the different restore types, and how they operate internally.

Restore Sharded Cluster Restore a sharded cluster from a stored snapshot.

Restore Replica Set Restore a replica set from a stored snapshot or custom point-in-time snapshot.

Restore Flows

On this page

Overview

Ops Manager Backup enables you to restore your `mongod` instance, replica set, or sharded cluster using stored snapshots, or from a point in time as far back as the retention period of your longest snapshot.

The general flows and options are the same whether you are restoring a `mongod`, replica set, or sharded cluster; the only major difference is that sharded cluster restores result in the production of multiple restore files that must be copied to the correct destination.

This page describes the different types of restores and different delivery options, and then provides some insight into the actual process that occurs when you request a restore through Ops Manager.

For a step-by-step guide to restoring a replica set or sharded cluster using Backup, see: *Restore MongoDB Deployments*.

Restore Types

With Backup, you can restore from a stored snapshot or build a custom snapshot reflecting a specific point in time. For all backups, restoring from a stored snapshot is faster than restoring from a specific point in time.

Snapshots provide a complete backup of the state of your MongoDB deployment at a given point in time. You can take snapshots every 6, 8, 12, or 24 hours and set a *retention policy* to determine for how long the snapshots are stored.

Point-in-time restores let you restore your `mongod` instance or replica set to a specific point in time in the past. You can restore to any point back to your oldest retained snapshot. For sharded clusters, point-in-time restores let you restore to a checkpoint. See *Checkpoints* for more information.

Point-in-time restores take longer to perform than snapshot restores, but allow you to restore more granularly. When you perform a point-in-time restore, Ops Manager takes the most recent snapshot that occurred prior to that point and then applies the oplog to bring the database up to the state it was in at that point in time. This way, Ops Manager creates a custom snapshot, which you can then use in your restore.

Delivery Methods and File Formats

Ops Manager provides two delivery methods: HTTP delivery, and SCP.

With HTTP delivery, Ops Manager creates a link that you can use to download the snapshot file or files. See: *Advanced Backup Restore Settings* for informaton about configuring the restore behaviors.

With the `SCP` delivery option, the Backup Daemon securely copies the restore file or files directly to your system. The *Backup File Format and Method* tutorial describes how to select a restore's delivery method and file format.

For SCP delivery, you can choose your file format to better suit your restore needs. With the *Individual DB Files* format, Ops Manager transmits the MongoDB data files directly to the target directory. The individual files format only requires sufficient space on the destination server for the data files.

In contrast, the *Archive (tar.gz)* option bundles the database files into a single `tar.gz` archive file, which you must extract before reconstruction your databases. This is generally faster than the individual files option, but requires temporary space on the server hosting the Backup Daemon and sufficient space on the destination server to hold the archive file and extract it.

Windows does not come with `SCP` and require additional setup outside the scope of this manual.

Restore Flows

Regardless of the delivery method and restore type, Ops Manager's restore flow follows a consistent pattern: when you request a restore, the MMS HTTP service calls out to the Backup Daemon, which prepares the snapshot you will receive, then either the user downloads the files from the MMS HTTP service, or the Backup Daemon securely copies the files to the destination server.

The following sections describe the restore flows for both snapshot restores and point-in-time restores, for each delivery and file format option.

HTTP Restore

Snapshot

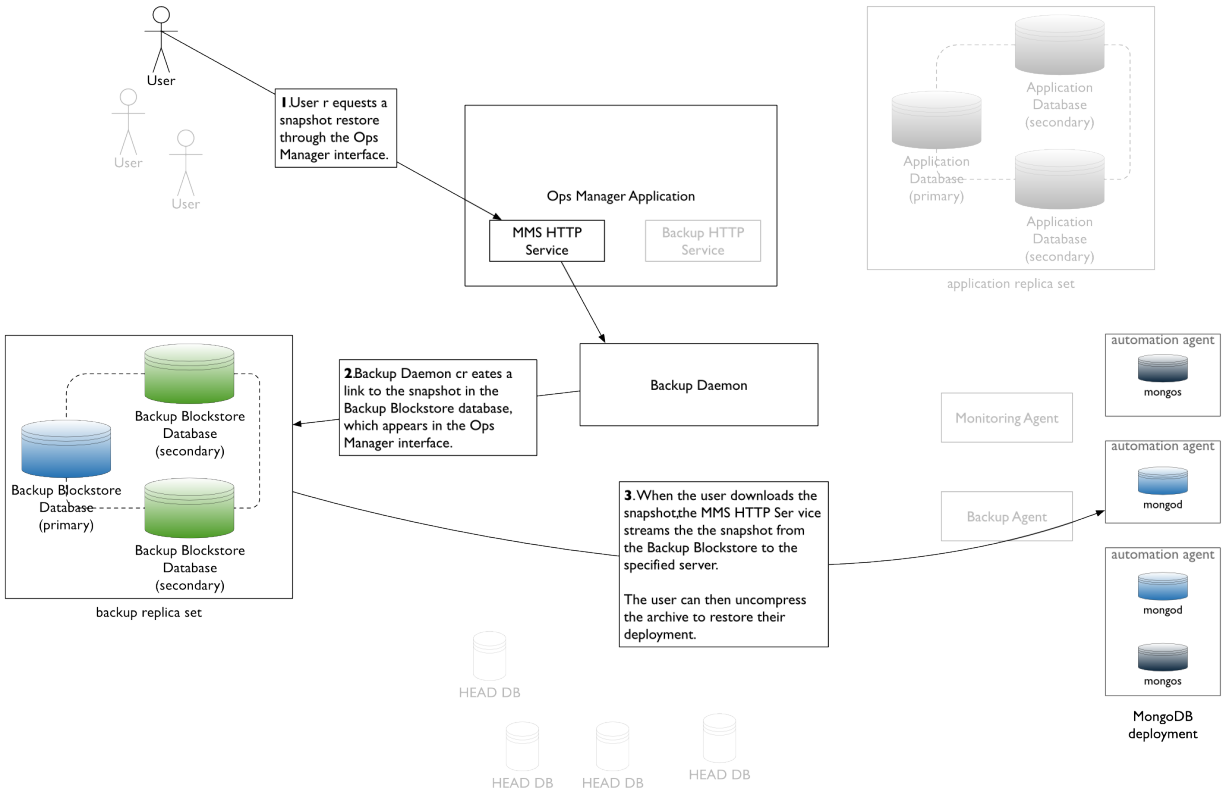
With the HTTP PULL snapshot restore, the Backup Daemon simply creates a link to the appropriate snapshot in the Backup Blockstore database. When the user clicks the download link, they download the snapshot from the MMS HTTP Service, which streams the file out of the Backup Blockstore.

This restore method has the advantage of taking up **no** space on the server hosting the Backup Daemon: the file passes directly from the Backup Blockstore to the destination server.

Point-In-Time

The HTTP PULL point-in-time restore follows the same pattern as the HTTP PULL snapshot restore, with added steps for applying the oplog. When the user requests the restore, the Backup Daemon retrieves the snapshot that immediately precedes the point in time and writes that snapshot to disk. The Backup Daemon then retrieves oplog entries from the Backup Blockstore and applies them, creating a custom snapshot from that point in time. The Daemon then writes the snapshot back to the Backup Blockstore. Finally, when the user clicks the download link, the user downloads the snapshot from the MMS HTTP Service, which streams the file out of the Backup Blockstore.

This restore method requires that you have adequate space on the server hosting the Backup Daemon for the snapshot files and oplog.



Archive SCP Restore

Snapshot

For a snapshot restore, with SCP archive delivery, the Backup Daemon simply retrieves the snapshot from the Backup Blockstore and writes it to its disk. The Backup Daemon then combines and compresses the snapshot into a `.tar.gz` archive and securely copies the archive to the destination server.

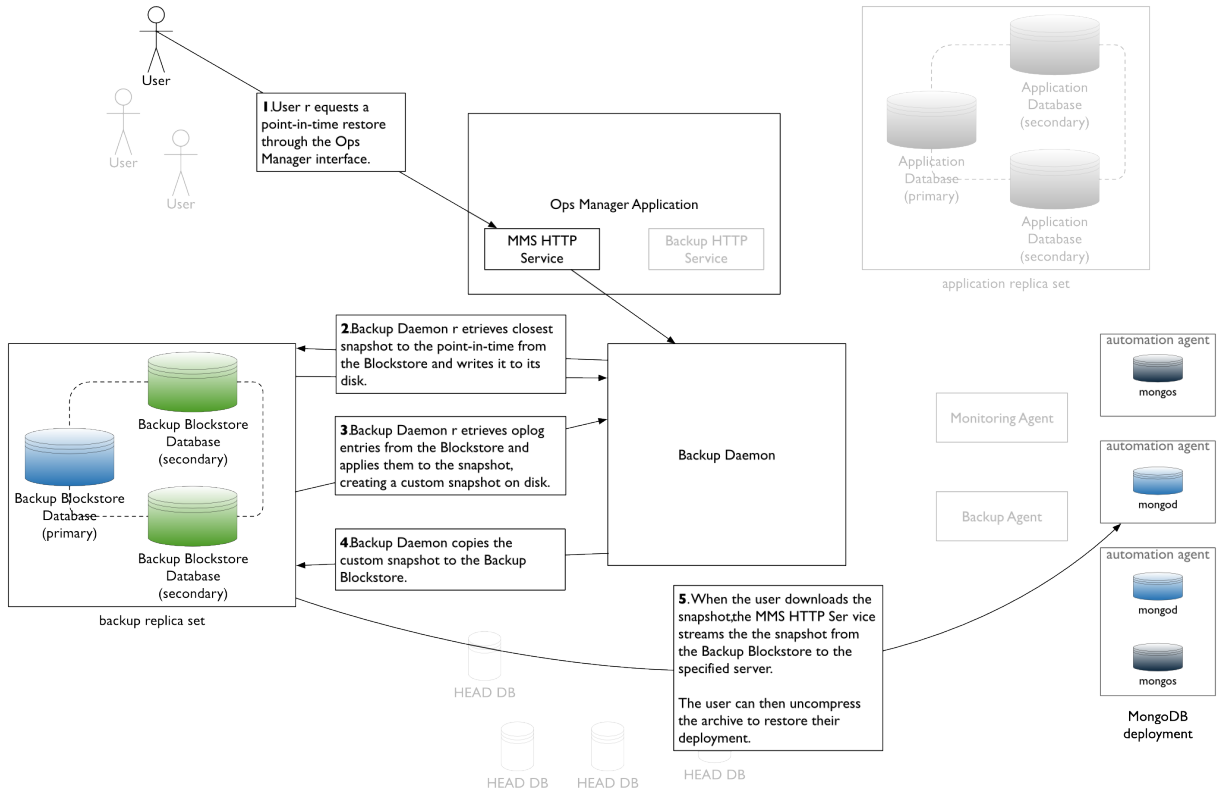
This restore method requries that you have adequate space on the server hosting the Backup Daemon for the snapshot files and archive.

Point-In-Time

The point-in-time restore with SCP archive delivery follows the same pattern as the snapshot restore, but with added steps for applying the oplog.

When the user requests the restore, the Backup Daemon retrieves the snapshot that immediately precedes the point in time and writes that snapshot to disk. The Backup Daemon then retrieves oplog entries from the Backup Blockstore and applies them, creating a custom snapshot for that point in time. The Backup Daemon then combines and compresses the snapshot into a `tar.gz` archive and securely copies the archive to the destination server.

This restore method requries that you have adequate space on the server hosting the Backup Daemon for the snapshot files, oplog, and archive.



Individual Files SCP Restore

Snapshot

For a snapshot restore, with SCP individual files delivery, the Backup Daemon simply retrieves the snapshot from the Backup Blockstore and securely copies the data files to the target directory on the desintation server.

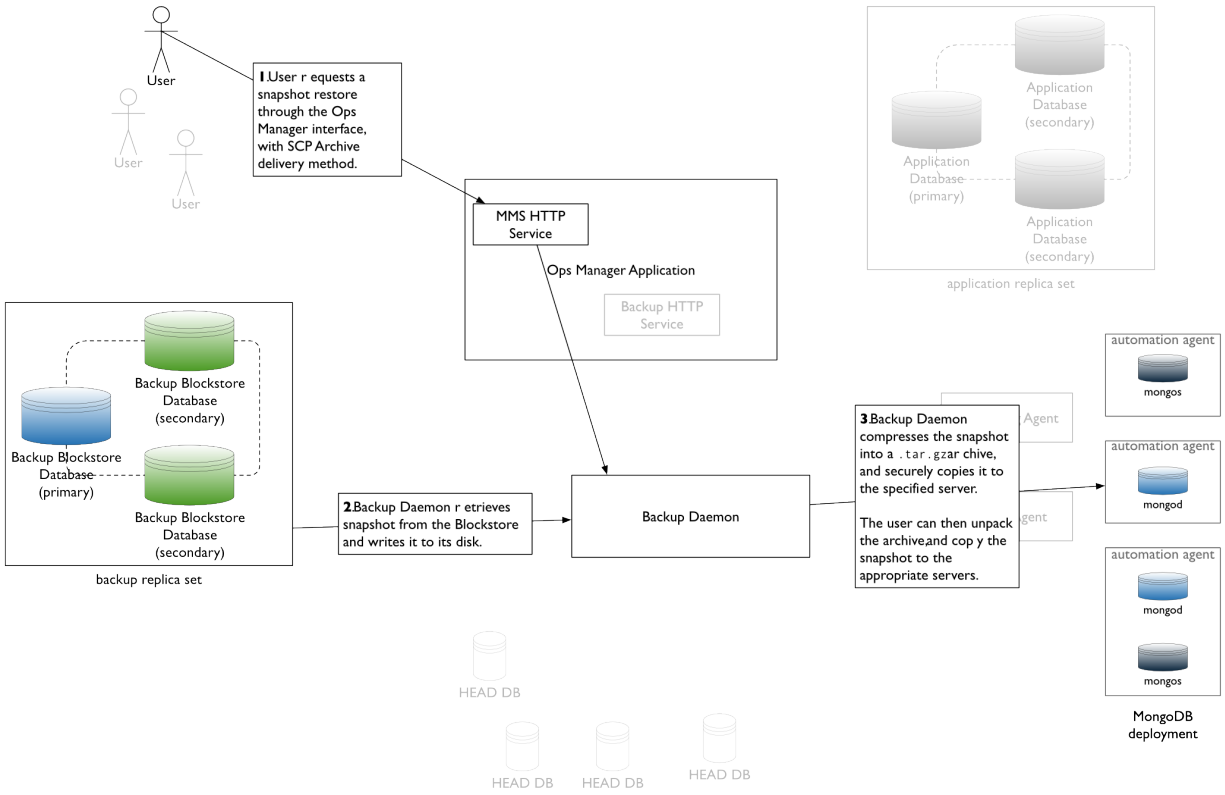
This restore method also has the advantage of taking up no space on the server hosting the Backup Daemon: the file passes directly from the Backup Blockstore to the destination server. The destination server requires only sufficient space for the uncompressed data files. The data *is* compressed during transmission.

Point-In-Time

The point-in-time restore with SCP individual files delivery follows the same pattern as the snapshot restore, but with added steps for applying the oplog.

When the user requests the restore, the Backup Daemon retrieves the snapshot that immediately precedes the point in time and writes that snapshot to disk. The Backup Daemon then retrieves oplog entries from the Backup Blockstore and applies them, creating a custom snapshot for that point in time. The Backup Daemon then securely copies the data files to the target directory on the desintation server

This restore method also requries that you have adequate space on the server hosting the Backup Daemon for the snapshot files and oplog. The destination server requires only sufficient space for the uncompressed data files. The data *is* compressed during transmission.



Restore a Sharded Cluster from a Backup

On this page

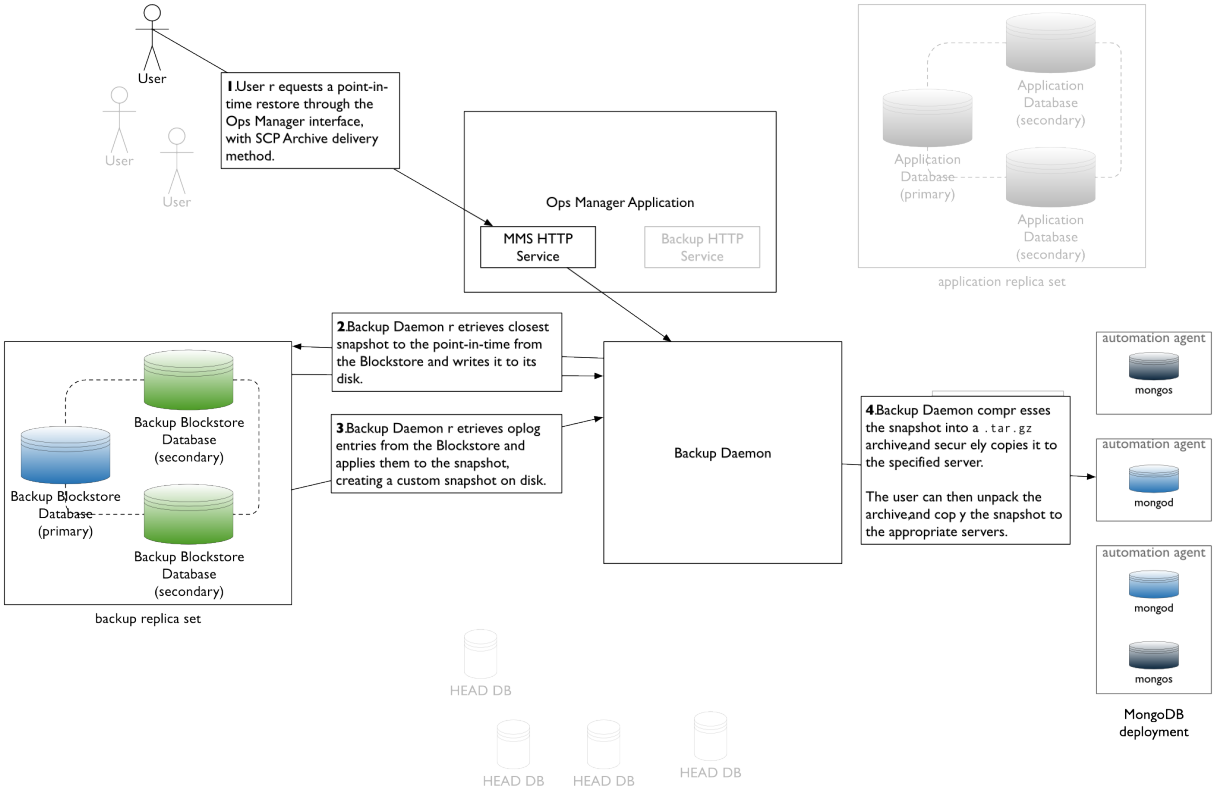
Overview

You can restore a sharded cluster onto new hardware from the artifacts captured by Backup.

You can restore from a snapshot or *checkpoint*. You must *enable checkpoints* to use them. When you restore from a checkpoint, Ops Manager takes the snapshot previous to the checkpoint and applies the oplog to create a custom snapshot. Checkpoint recovery takes longer than recovery from a stored snapshot.

Ops Manager provides restore files as downloadable archive; Ops Manager can also `scp` files directly to your system. The `scp` delivery method requires additional configuration but provides faster delivery.

Ops Manager provides a separate backup artifacts for each shard and one file for the config servers.



Sequence

The sequence to restore a snapshot is to:

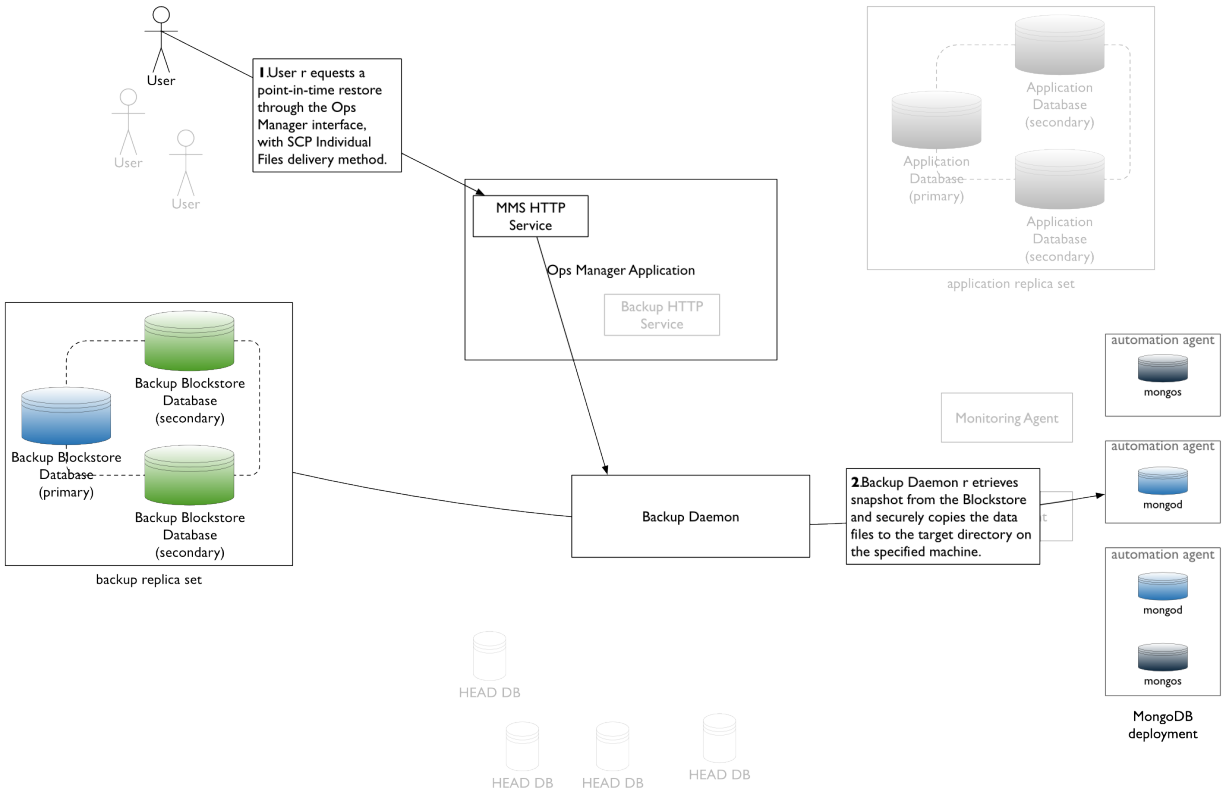
- select and download the restore files,
- distribute the restore files to their new locations,
- start the `mongod` instances,
- configure each shard's replica set, and
- configure and start the cluster.

Considerations

Client Requests During Restoration

You must ensure that the MongoDB deployment does not receive client requests during restoration. You must either:

- restore to new systems with new hostnames and reconfigure your application code once the new deployment is running, *or*
- ensure that the MongoDB deployment will *not* receive client requests while you restore data.



Snapshots when Agent Cannot Stop Balancer

Ops Manager displays a warning next to cluster snapshots taken while the balancer is enabled. If you restore from such a snapshot, you run the risk of lost or orphaned data. For more information, see *Snapshots when Agent Cannot Stop Balancer*.

Procedures

Select and Download the Snapshot Files

Step 1: Select the *Backup* tab and then select *Sharded Cluster Status*.

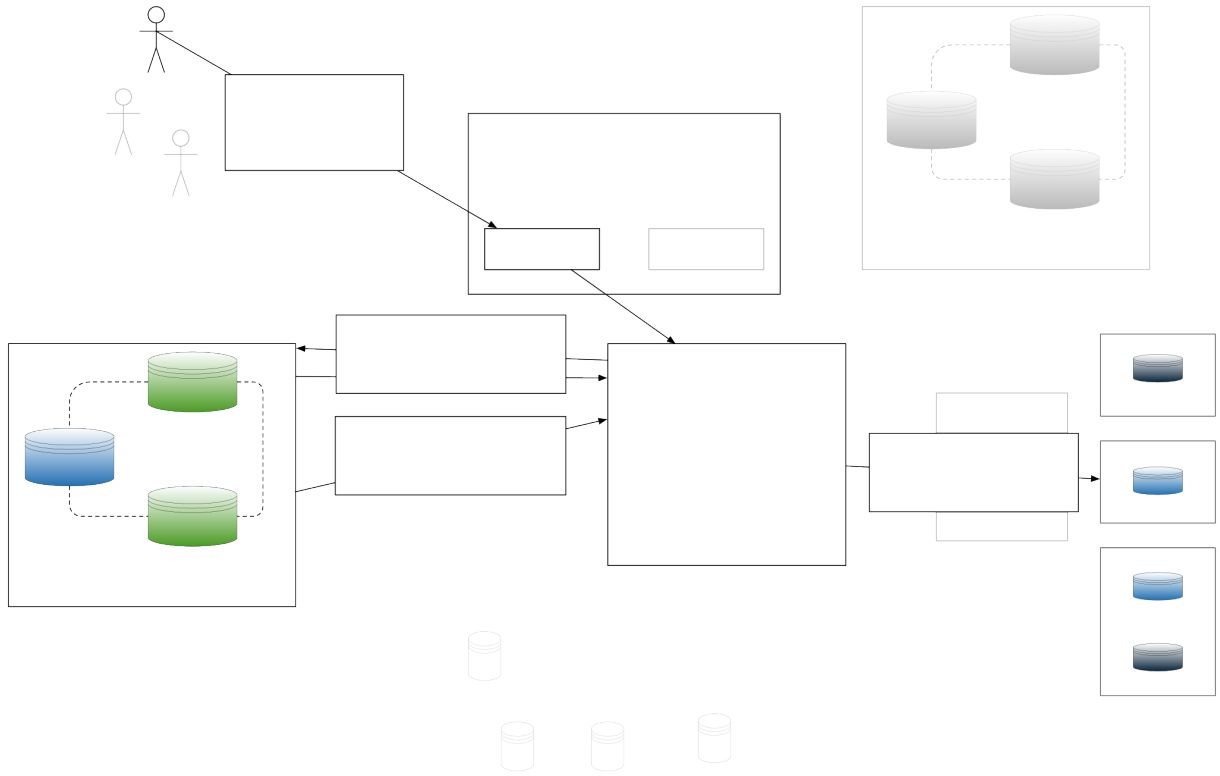
Step 2: Click the name of the sharded cluster to restore.

Ops Manager displays your selection's stored snapshots.

Step 3: Click the *Restore* button and select the snapshot from which to restore.

Click the *Restore* button at the top of the page. On the resulting page, choose from restoring from a stored snapshot or creating a custom snapshot.

To select a **stored snapshot**, click the *Restore this snapshot* link next to the snapshot.



To create a **custom snapshot** from a checkpoint, select the *Use Custom Point In Time* checkbox and enter the point in time in the *Date* and *Time* fields, and click *Next*.

Step 4: Select the checkpoint.

Ops Manager lists the checkpoints that are the closest match to the point in time that you selected. Select a checkpoint from which to create the snapshot, and click *Next*.

Step 5: Select HTTP as the delivery method for the snapshot.

In the *Delivery Method* field, select *Pull via Secure HTTP (HTTPS)*.

Optionally, you can instead choose SCP as the delivery method. See: *Retrieve a Snapshot with SCP Delivery* for the SCP delivery option's configuration. If you choose SCP, you must provide the hostname and port of the server to receive the files and provide access to the server through a username and password or though an SSH key. Follow the instructions on the Ops Manager screen.

Step 6: Select `tar.gz` as the download format.

In the *Format* drop-down list, select *Archive (tar.gz)*.

Step 7: Finalize the request.

Click *Finalize Request* and confirm your identify via two-factor verification. Then click *Finalize Request* again.

Step 8: Retrieve the snapshot.

Ops Manager creates one-time links to tar files for the snapshot. The links are available for one download each, and each expires after an hour.

To download the tar files, select the Ops Manager *Backup* tab and then *Restore Jobs*. When the restore job completes, the *download* link appears for every config server and shard in the cluster. Click each link to download the tar files and copy each tar file to its server. For a shard, copy the file to every member of the shard's replica set.

If you **optionally** chose SCP as the delivery method, the files are copied to the server directory you specfied. To verify that the files are complete, *see the section on how to validate an SCP restore*.

Restore Each Shard's Primary

For *all* shards, restore the primary. You must have a copy of the snapshot on the server that provides the primary:

Step 1: Shut down the entire replica set.

Shut down the replica set's `mongod` processes using one of the following methods, depending on your configuration:

- **Automated Deployment**:

  If you use Ops Manager Automation to manage the replica set, you must shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

- **Non-Automated Deployment on MongoDB 2.6 or Later**:

  Connect to each member of the set and issue the following:

  ```
  use admin
  db.shutdownServer()
  ```

- **Non-Automated Deployment on MongoDB 2.4 or earlier**:

  Connect to each member of the set and issue the following:

  ```
  use admin
  db.shutdownServer( { force: true } )
  ```

Step 2: Restore the snapshot data files to the primary.

Extract the data files to the location where the `mongod` instance will access them through the `dbpath` setting. If you are restoring to existing hardware, use a different data directory than used previously. The following are example commands:

```
tar -xvf <backup-restore-name>.tar.gz
mv <backup-restore-name> /data
```

Step 3: Start the primary with the new `dbpath`.

For example:

```
mongod --dbpath /<path-to-data> --replSet <replica-set-name> --logpath /<path-to-data>
↪/mongodb.log --fork
```

Step 4: Connect to the primary and initiate the replica set.

For example, first issue the following to connect:

```
mongo
```

And then issue `rs.initiate()`:

```
rs.initiate()
```

Step 5: Restart the primary as a standalone, *without* the `--replSet` option.

Use the following sequence:

1. Shut down the process using one of the following methods:

   - **Automated Deployment**:

     Shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

   - **Non-Automated Deployment on MongoDB 2.6 or Later**:

     ```
     use admin
     db.shutdownServer()
     ```

   - **Non-Automated Deployment on MongoDB 2.4 or earlier**:

     ```
     use admin
     db.shutdownServer( { force: true } )
     ```

2. Restart the process as a standalone:

   ```
   mongod --dbpath /<path-to-data> --logpath /<path-to-data>/mongodb.log --fork
   ```

Step 6: Connect to the primary and drop the oplog.

For example, first issue the following to connect:

```
mongo
```

And then issue `rs.drop()` to drop the oplog.

```
use local
db.oplog.rs.drop()
```

Step 7: Run the `seedSecondary.sh` script on the primary.

The seedSecondary.sh script re-creates the oplog collection and seeds it with the timestamp of the snapshot's creation. This will allow the secondary to come back up to time without requiring a full initial sync. This script is customized by Ops Manager for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command at the system prompt, where `<mongod-port>` is the port of the `mongod` instance and `<oplog-size-in-gigabytes>` is the size of the replica set's oplog:

```
./seedSecondary.sh <mongod-port> <oplog-size-in-gigabytes>
```

Step 8: Restart the primary as part of a replica set.

Use the following sequence:

1. Shut down the process using one of the following methods:

   - **Automated Deployment**:

     Shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

   - **Non-Automated Deployment on MongoDB 2.6 or Later**:

     ```
     use admin
     db.shutdownServer()
     ```

   - **Non-Automated Deployment on MongoDB 2.4 or earlier**:

     ```
     use admin
     db.shutdownServer( { force: true } )
     ```

2. Restart the process as part of a replica set:

   ```
   mongod --dbpath /<path-to-data> --replSet <replica-set-name>
   ```

Restore All Secondaries

After you have restored the primary for a shard you can restore all secondaries. You must have a copy of the snapshot on all servers that provide the secondaries:

Step 1: Connect to the server where you will create the new secondary.

Step 2: Restore the snapshot data files to the secondary.

Extract the data files to the location where the `mongod` instance will access them through the `dbpath` setting. If you are restoring to existing hardware, use a different data directory than used previously. The following are example commands:

```
tar -xvf <backup-restore-name>.tar.gz
mv <backup-restore-name> /data
```

Step 3: Start the secondary as a standalone, *without* the `--replSet` option.

Use the following sequence:

1. Shut down the process using one of the following methods:

   - **Automated Deployment**:

     Shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

   - **Non-Automated Deployment on MongoDB 2.6 or Later**:

     ```
     use admin
     db.shutdownServer()
     ```

   - **Non-Automated Deployment on MongoDB 2.4 or earlier**:

     ```
     use admin
     db.shutdownServer( { force: true } )
     ```

2. Restart the process as a standalone:

   ```
   mongod --dbpath /<path-to-data> --logpath /<path-to-data>/mongodb.log --fork
   ```

Step 4: Run the `seedSecondary.sh` script on the secondary.

The seedSecondary.sh script re-creates the oplog collection and seeds it with the timestamp of the snapshot's creation. This will allow the secondary to come back up to time without requiring a full initial sync. This script is customized by Ops Manager for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command at the system prompt, where `<mongod-port>` is the port of the `mongod` instance and `<oplog-size-in-gigabytes>` is the size of the replica set's oplog:

```
./seedSecondary.sh <mongod-port> <oplog-size-in-gigabytes>
```

Step 5: Restart the secondary as part of the replica set.

Use the following sequence:

1. Shut down the process using one of the following methods:

   - **Automated Deployment**:

     Shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

   - **Non-Automated Deployment on MongoDB 2.6 or Later**:

     ```
     use admin
     db.shutdownServer()
     ```

   - **Non-Automated Deployment on MongoDB 2.4 or earlier**:

     ```
     use admin
     db.shutdownServer( { force: true } )
     ```

2. Restart the process as part of a replica set:

```
mongod --dbpath /<path-to-data> --replSet <replica-set-name>
```

Step 6: Connect to the primary and add the secondary to the replica set.

Connect to the primary and use `rs.add()` to add the secondary to the replica set.

```
rs.add("<host>:<port>")
```

Repeat this operation for each member of the set.

Restore Each Config Server

Perform this procedure separately for each config server. Each config server must have a copy of the tar file with the config server data.

Step 1: Restore the snapshot to the config server.

Extract the data files to the location where the config server's `mongod` instance will access them. This is the location you will specify as the `dbPath` when running `mongod` for the config server.

```
tar -xvf <backup-restore-name>.tar.gz
mv <backup-restore-name> /data
```

Step 2: Start the config server.

The following example starts the config server using the new data:

```
mongod --configsvr --dbpath /data
```

Step 3: Update the sharded cluster metadata.

If the new shards do not have the same hostnames and ports as the original cluster, you must update the shard metadata. To do this, connect to each config server and update the data.

First connect to the config server with the `mongo` shell. For example:

```
mongo
```

Then access the `shards` collection in the config database. For example:

```
use config
db.shards.find().pretty()
```

The `find()` method returns the documents in the `shards` collection. The collection contains a document for each shard in the cluster. The `host` field for a shard displays the name of the shard's replica set and then the hostname and port of the shard. For example:

```
{ "_id" : "shard0000", "host" : "shard1/localhost:30000" }
```

To change a shard's hostname and port, use the MongoDB `update()` command to modify the documents in the `shards` collection.

Start the `mongos`

Start the cluster's `mongos` bound to your new config servers.

Restore a Replica Set from a Backup

On this page

Overview

You can restore a replica set from the artifacts captured by Ops Manager Backup. You can restore either a stored snapshot or a point in time in the last 24 hours between snapshots. If you restore from a point in time, Ops Manager Backup creates a custom snapshot for the selected point by applying the oplog to the previous regular snapshot. Point-in-time recovery takes longer than recovery from a stored snapshot.

When you select a snapshot to restore, Ops Manager creates a link to download the snapshot as a tar file. The link is available for one download only and times out after an hour. You can optionally have Ops Manager `scp` the tar file directly to your system. The `scp` delivery method requires additional configuration but provides faster delivery. Windows does not come with `scp` and require additional setup outside the scope of this manual.

You can restore either to new hardware or existing hardware. If you restore to existing hardware, use a different data directory than used previously.

Sequence

The sequence used here to restore a replica set is to download the restore file and distribute it to each server, restore the primary, and then restore the secondaries. For additional approaches to restoring replica sets, see the procedure from the MongoDB Manual to Restore a Replica Set from a Backup.

Prerequisites

Oplog Size

To seed each replica set member, you will use the `seedSecondary.sh` script included in the backup restore file. When you run the script, you will provide the replica set's oplog size, in gigabytes. If you do not have the size, see the section titled "Check the Size of the Oplog" on the Troubleshoot Replica Sets page of the MongoDB manual.

Client Requests

You must ensure that the MongoDB deployment does not receive client requests during restoration. You must either:

- restore to new systems with new hostnames and reconfigure your application code once the new deployment is running, *or*
- ensure that the MongoDB deployment will *not* receive client requests while you restore data.

Procedures

Select and Download the Snapshot

Step 1: Select the *Backups* tab and then select *Replica Set Status*.

Step 2: Click the name of the replica set to restore.

Ops Manager displays your selection's stored snapshots.

Step 3: Select the snapshot from which to restore.

To select a **stored snapshot**, click the *Restore this snapshot* link next to the snapshot.

To select a **custom snapshot**, click the *Restore* button at the top of the page. In the resulting page, select a snapshot as the starting point. Then select the *Use Custom Point In Time* checkbox and enter the point in time in the *Date* and *Time* fields. Ops Manager includes all operations up to but not including the point in time. For example, if you select 12:00, the last operation in the restore is 11:59:59 or earlier. Click *Next*.

Step 4: Select HTTP as the delivery method for the snapshot.

In the *Delivery Method* field, select *Pull via Secure HTTP (HTTPS)*.

Optionally, you can instead choose SCP as the delivery method. See: *Retrieve a Snapshot with SCP Delivery* for the SCP delivery option's configuration. If you choose SCP, you must provide the hostname and port of the server to receive the files and provide access to the server through a username and password or though an SSH key. Follow the instructions on the Ops Manager screen.

Step 5: Finalize the request.

Click *Finalize Request* and confirm your identify via two-factor verification. Then click *Finalize Request* again.

Step 6: Retrieve the snapshot.

Ops Manager creates a one-time link to a tar file of the snapshot. The link is available for one download and times out after an hour.

To download the snapshot, select the Ops Manager *Backup* tab and then select *Restore Jobs*. When the restore job completes, select the *download* link next to the snapshot.

If you **optionally** chose SCP as the delivery method, the files are copied to the server directory you specfied. To verify that the files are complete, *see the section on how to validate an SCP restore*.

Step 7: Copy the snapshot to each server to restore.

Restore the Primary

You must have a copy of the snapshot on the server that provides the primary:

Step 1: Shut down the entire replica set.

Shut down the replica set's `mongod` processes using one of the following methods, depending on your configuration:

- **Automated Deployment**:

  If you use Ops Manager Automation to manage the replica set, you must shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

- **Non-Automated Deployment on MongoDB 2.6 or Later**:

  Connect to each member of the set and issue the following:

```
use admin
db.shutdownServer()
```

- **Non-Automated Deployment on MongoDB 2.4 or earlier**:

  Connect to each member of the set and issue the following:

```
use admin
db.shutdownServer( { force: true } )
```

Step 2: Restore the snapshot data files to the primary.

Extract the data files to the location where the `mongod` instance will access them through the `dbpath` setting. If you are restoring to existing hardware, use a different data directory than used previously. The following are example commands:

```
tar -xvf <backup-restore-name>.tar.gz
mv <backup-restore-name> /data
```

Step 3: Start the primary with the new `dbpath`.

For example:

```
mongod --dbpath /<path-to-data> --replSet <replica-set-name> --logpath /<path-to-data>
↪/mongodb.log --fork
```

Step 4: Connect to the primary and initiate the replica set.

For example, first issue the following to connect:

```
mongo
```

And then issue `rs.initiate()`:

```
rs.initiate()
```

Step 5: Restart the primary as a standalone, *without* the `--replSet` option.

Use the following sequence:

1. Shut down the process using one of the following methods:

   - **Automated Deployment**:

     Shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

   - **Non-Automated Deployment on MongoDB 2.6 or Later**:

     ```
     use admin
     db.shutdownServer()
     ```

   - **Non-Automated Deployment on MongoDB 2.4 or earlier**:

     ```
     use admin
     db.shutdownServer( { force: true } )
     ```

2. Restart the process as a standalone:

   ```
   mongod --dbpath /<path-to-data> --logpath /<path-to-data>/mongodb.log --fork
   ```

Step 6: Connect to the primary and drop the oplog.

For example, first issue the following to connect:

```
mongo
```

And then issue `rs.drop()` to drop the oplog.

```
use local
db.oplog.rs.drop()
```

Step 7: Run the `seedSecondary.sh` script on the primary.

The seedSecondary.sh script re-creates the oplog collection and seeds it with the timestamp of the snapshot's creation. This will allow the secondary to come back up to time without requiring a full initial sync. This script is customized by Ops Manager for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command at the system prompt, where `<mongod-port>` is the port of the `mongod` instance and `<oplog-size-in-gigabytes>` is the size of the replica set's oplog:

```
./seedSecondary.sh <mongod-port> <oplog-size-in-gigabytes>
```

Step 8: Restart the primary as part of a replica set.

Use the following sequence:

1. Shut down the process using one of the following methods:

   - **Automated Deployment**:

     Shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

   - **Non-Automated Deployment on MongoDB 2.6 or Later**:

     ```
     use admin
     db.shutdownServer()
     ```

   - **Non-Automated Deployment on MongoDB 2.4 or earlier**:

     ```
     use admin
     db.shutdownServer( { force: true } )
     ```

2. Restart the process as part of a replica set:

   ```
   mongod --dbpath /<path-to-data> --replSet <replica-set-name>
   ```

Restore Each Secondary

After you have restored the primary you can restore all secondaries. You must have a copy of the snapshot on all servers that provide the secondaries:

Step 1: Connect to the server where you will create the new secondary.

Step 2: Restore the snapshot data files to the secondary.

Extract the data files to the location where the `mongod` instance will access them through the `dbpath` setting. If you are restoring to existing hardware, use a different data directory than used previously. The following are example commands:

```
tar -xvf <backup-restore-name>.tar.gz
mv <backup-restore-name> /data
```

Step 3: Start the secondary as a standalone, *without* the `--replSet` option.

Use the following sequence:

1. Shut down the process using one of the following methods:

   - **Automated Deployment**:

     Shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

   - **Non-Automated Deployment on MongoDB 2.6 or Later**:

     ```
     use admin
     db.shutdownServer()
     ```

   - **Non-Automated Deployment on MongoDB 2.4 or earlier**:

     ```
     use admin
     db.shutdownServer( { force: true } )
     ```

2. Restart the process as a standalone:

```
mongod --dbpath /<path-to-data> --logpath /<path-to-data>/mongodb.log --fork
```

Step 4: Run the `seedSecondary.sh` script on the secondary.

The seedSecondary.sh script re-creates the oplog collection and seeds it with the timestamp of the snapshot's creation. This will allow the secondary to come back up to time without requiring a full initial sync. This script is customized by Ops Manager for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command at the system prompt, where `<mongod-port>` is the port of the `mongod` instance and `<oplog-size-in-gigabytes>` is the size of the replica set's oplog:

```
./seedSecondary.sh <mongod-port> <oplog-size-in-gigabytes>
```

Step 5: Restart the secondary as part of the replica set.

Use the following sequence:

1. Shut down the process using one of the following methods:

   - **Automated Deployment**:

     Shut down through the Ops Manager console. See *Shut Down MongoDB Processes*.

   - **Non-Automated Deployment on MongoDB 2.6 or Later**:

     ```
     use admin
     db.shutdownServer()
     ```

   - **Non-Automated Deployment on MongoDB 2.4 or earlier**:

     ```
     use admin
     db.shutdownServer( { force: true } )
     ```

2. Restart the process as part of a replica set:

   ```
   mongod --dbpath /<path-to-data> --replSet <replica-set-name>
   ```

Step 6: Connect to the primary and add the secondary to the replica set.

Connect to the primary and use `rs.add()` to add the secondary to the replica set.

```
rs.add("<host>:<port>")
```

Repeat this operation for each member of the set.

6.6 Restore MongoDB Instances with Backup

Use the procedures in these tutorials to restore data from Backup artifacts to MongoDB instances. If you want to restore an entire MongoDB deployment, use the *Restore MongoDB Deployments* tutorials.

Restore from a Stored Snapshot with HTTP Pull Restore a replica set or sharded cluster from a stored snapshot with HTTP Pull delivery.

Restore from a Stored Snapshot with SCP Delivery Restore a replica set or sharded cluster from a stored snapshot with SCP delivery.

Restore from a Point in the Last Day Restore a replica set from a custom snapshot from any point within a 24-hour period of time.

Restore a Single Database Restore only a portion of a backup to a new `mongod` instance.

Seed a New Secondary Use Ops Manager Backup to seed a new secondary in an existing replica set.

Restore from a Stored Snapshot

On this page

Overview

With Backup, you can restore from a stored snapshot or build a *custom snapshot* reflecting a different point in the last 24 hours. For all backups, restoring from a stored snapshot is faster than restoring from a custom snapshot in the last 24 hours.

By default, Backup automatically takes and stores a snapshot every 6 hours. Snapshots remain available for restoration following the *snapshot retention policy*.

For replica sets, you will receive one `.tar.gz` file containing your data; for sharded clusters, you will receive a series of `.tar.gz` files.

Procedure

Step 1: Select the *Backups* tab, and then select either *Sharded Cluster Status* or *Replica Set Status*.

Step 2: Click the name of the sharded cluster or replica set to restore.

Ops Manager displays your selection's stored snapshots.

Step 3: Select the snapshot from which to restore.

To select a **stored snapshot**, click the *Restore this snapshot* link next to the snapshot.

To select a **custom snapshot**, click the *Restore* button at the top of the page. In the resulting page, select a snapshot as the starting point. Then select the *Use Custom Point In Time* checkbox and enter the point in time in the *Date* and *Time* fields. Ops Manager includes all operations up to but not including the point in time. For example, if you select 12:00, the last operation in the restore is 11:59:59 or earlier. Click *Next*.

Step 4: Select HTTP as the delivery method for the snapshot.

In the *Delivery Method* field, select *Pull via Secure HTTP (HTTPS)*.

Optionally, you can instead choose SCP as the delivery method. See: *Retrieve a Snapshot with SCP Delivery* for the SCP delivery option's configuration. If you choose SCP, you must provide the hostname and port of the server to receive the files and provide access to the server through a username and password or though an SSH key. Follow the instructions on the Ops Manager screen.

Step 5: Finalize the request.

Click *Finalize Request* and confirm your identify via two-factor verification. Then click *Finalize Request* again.

Step 6: Retrieve the snapshot.

To download the snapshot, select the Ops Manager *Backup* tab and then select *Restore Jobs*. When the restore job completes, select the *download* link next to the snapshot.

For a sharded clusters, Ops Manager provides several *download* links for the several `.tar.gz` files.

Step 7: Extract the data files from the .tar.gz archive created by the backup service.

```
tar -zxvf <tarball-name>.tar.gz
```

Step 8: Select the location where the mongod will access the data files.

The directory you choose will become the mongod's data directory. You can either create a new directory or use the existing location of the extracted data files.

If you create a new directory, move the files to that directory.

If you use the existing location of the extracted data files, you can optionally create a symbolic link to the location using the following command, where `<hash>-<rsname>-<time>` is the name of the snapshot and `<dbpath>` is the data directory:

```
ln -s <hash>-<rsname>-<time>/ <dbpath>
```

Step 9: Start the mongod with the new data directory as the dbpath.

In the mongod configuration, set the `dbpath` option to the path of the data directory that holds the data files from the Backup snapshot.

```
mongod --dbpath /data/db
```

Additional Information

Restore from a Point in the Last 24 Hours

Retrieve a Snapshot with SCP Delivery

On this page

In addition to the *PULL via Secure HTTP (HTTPS)* delivery method, Ops Manager Backup supports snapshot delivery using SCP. With *Push via Secure Copy (SCP)*, Ops Manager copies the snapshot directly to the target system.

For an overview of Ops Manager's internal behavior and a more detailed description of the restore types and delivery options, see: *Restore Flows*.

Requirements

In order for Ops Manager to be able to securely copy your snapshots to your system, you must grant Ops Manager access to the server, either by providing a username and password, or by providing a username with an SSH public key. See: *Select Backup File Delivery Method and Format* for instructions.

Procedure

Step 1: Select the *Backups* tab, and then select either *Sharded Cluster Status* or *Replica Set Status*.

Step 2: Click the name of the sharded cluster or replica set to restore.

Ops Manager displays your selection's stored snapshots.

Step 3: Select the snapshot from which to restore.

To select a **stored snapshot**, click the *Restore this snapshot* link next to the snapshot.

To select a **custom snapshot**, click the *Restore* button at the top of the page. In the resulting page, select a snapshot as the starting point. Then select the *Use Custom Point In Time* checkbox and enter the point in time in the *Date* and *Time* fields. Ops Manager includes all operations up to but not including the point in time. For example, if you select 12:00, the last operation in the restore is 11:59:59 or earlier. Click *Next*.

Step 4: Select *Push via Secure Copy (SCP)* as the delivery method.

In the *Select Restore Destination* screen's *Delivery Method* field, select *Push via Secure Copy (SCP)* to transfer the backup to your server via SCP.

Step 5: Enter the SCP details.

Enter the following:

Format: Select the file format. The *Individual DB Files* option copies the data files directly to the target directory.

With the *Individual DB Files* format, Ops Manager transmits the MongoDB data files directly to the target directory. The individual files format only requires sufficient space on the destination server for the data files.

In contrast, the *Archive (tar.gz)* option bundles the database files into a single `tar.gz` archive file, which you must extract before reconstruction your databases. This requires temporary space on the server hosting the Backup Daemon and sufficient space on the destination server to hold the archive file and extract it.

SCP Host: Your server.

SCP Port: SCP runs over port 22 by default.

SCP User: A username for your server.

Auth Method: Select either password or SSH as the authentication method. See: *Select Backup File Delivery Method and Format* for details about the authentication methods.

Password: If using a password for authentication, enter the password. To test the authentication, click *Test*.

Target Directory: The server location to which to transfer the snapshot.

Step 6: Finalize the request.

Click *Finalize Request* and confirm your identify via two-factor verification. Then click *Finalize Request* again.

Ops Manager pushes the snapshot to the location you specified.

Additional Information

The *Restore a Sharded Cluster from a Backup* and *Restore a Replica Set from a Backup* tutorials provide more detailed instructions for restoring a sharded cluster or replica set using snapshot.

Restore from a Point in the Last 24 Hours

Backup lets you restore data from a point within the last 24-hour period. Ops Manager creates a backup that includes all operations up to the point in time you select. The point in time is an upper *exclusive* bound: if you select a timestamp of 12:00, then the last operation in the restore will be no later than 11:59:59.

To restore from a point in time, see the procedure for the resource you are restoring:

- *Restore a Sharded Cluster from a Backup*
- *Restore a Replica Set from a Backup*

Restore a Single Database

On this page

Overview

Backup snapshots contain a complete copy of your `mongod` data directory. In MongoDB 2.6 and earlier, you can restore a portion of your data, such as a single database or collection, from a backup with the `mongodump` and `mongorestore` tools and the `--dbpath` option.

Important: MongoDB 3.0 introduced major changes to the MongoDB tools in order to support additional storage engines such as WiredTiger. In MongoDB 3.0, `mongodump` and `mongorestore` do not support the `--dbpath` option. As such, you cannot use this procedure for a partial data restore on a `mongod` running MongoDB 3.0.

If you anticipate you will want to restore a single database, *create a backup for just that database* by excluding the databases or collections that you do not want to back up using the *Blacklist* field in the *Advanced Settings*.

Procedure

Select and Download a Snapshot

Step 1: Select the *Backups* tab and then select *Replica Set Status*.

Step 2: Click the name of the replica set that contains the database to restore.

Ops Manager displays your selection's stored snapshots.

Step 3: Select the snapshot from which to restore.

To select a **stored snapshot**, click the *Restore this snapshot* link next to the snapshot.

To select a **custom snapshot**, click the *Restore* button at the top of the page. In the resulting page, select a snapshot as the starting point. Then select the *Use Custom Point In Time* checkbox and enter the point in time in the *Date* and *Time* fields. Ops Manager includes all operations up to but not including the point in time. For example, if you select 12:00, the last operation in the restore is 11:59:59 or earlier. Click *Next*.

Step 4: Select HTTP as the delivery method for the snapshot.

In the *Delivery Method* field, select *Pull via Secure HTTP (HTTPS)*.

Optionally, you can instead choose SCP as the delivery method. See: *Retrieve a Snapshot with SCP Delivery* for the SCP delivery option's configuration. If you choose SCP, you must provide the hostname and port of the server to receive the files and provide access to the server through a username and password or though an SSH key. Follow the instructions on the Ops Manager screen.

Step 5: Finalize the request.

Click *Finalize Request* and confirm your identify via two-factor verification. Then click *Finalize Request* again.

Step 6: Retrieve the snapshot.

Ops Manager creates a one-time link to a tar file of the snapshot. The link is available for one download and times out after an hour.

To download the snapshot, select the Ops Manager *Backup* tab and then select *Restore Jobs*. When the restore job completes, select the *download* link next to the snapshot.

If you **optionally** chose SCP as the delivery method, the files are copied to the server directory you specfied. To verify that the files are complete, *see the section on how to validate an SCP restore*.

Restore the Database

Step 1: Use the mongodump command to dump a single database.

Use the unpacked snapshot restore directory as the `dpath` switch and the single database name as the `--db` switch in the `mongodump` command:

```
mongodump --dbpath <path> --db <database-name>
```

Step 2: Use the mongorestore command to import the single database dump.

Enter this `mongorestore` command:

```
mongorestore --db <database-name> --drop
```

You also may specify the `--drop` switch to drop all collections from the target database before you restore them from the `bson` file created with `mongodump`.

Seed a New Secondary from Backup Restore

On this page

Overview

When a natural synchronization of a new secondary host costs too much time or resources, seeding a secondary from an Ops Manager Backup restore is a faster better alternative. Seeding also does not impact an active MongoDB instance to retrieve data.

Prerequisites

To seed a secondary from a backup restore file, you must have:

- A backup restore file.
- The `seedSecondary.sh` script included in the backup restore file.

When you run the `seedSecondary.sh` script as part of this tutorial, you must provide the replica set's oplog size, in gigabytes. If you do not have the size, see the section titled "Check the Size of the Oplog" on the Troubleshoot Replica Sets page of the MongoDB manual.

Considerations

An Ops Manager backup stores all database files in a single directory. If you run MongoDB with the `directoryPerDb` option, then after restore you must redistribute the files to the different databases. Ops Manager Backup does not provide restoration artifacts that use the `directoryPerDb` option.

The seedSecondary.sh file will not be in the backup restore if you have blacklisted dbs or collections or have resynced your backup after the snapshot (or for config servers). In these cases, including the script would cause an inconsistent secondary. In the case of a blacklist, your secondary would not include some collections which would cause problems for your deployment.

Seeding a new secondary from a backup restore requires an oplog window on the current primary that spans back to the snapshot's timestamp.

Procedure

Step 1: Remove the broken secondary from your replica set.

```
rs.remove("SECONDARYHOST:SECONDARYPORT")
```

Step 2: Login to the server on which to create the new secondary.

Step 3: Bring up new node as a standalone.

```
tar -xvf backup-restore-name.tar.gz
mv backup-restore-name data
mongod --port <alternate-port> --dbpath /data
```

Where `ALTERNATEPORT` is not the usual port your secondary runs on.

Step 4: Run seedSecondary.sh script to create oplog collection and seed with correct timestamp.

The seedSecondary.sh script re-creates the oplog collection and seeds it with correct timestamp.

To run the script, issue the following command at the system prompt, where `<mongod-port>` is the port of the `mongod` instance and `<oplog-size-in-gigabytes>` is the size of the replica set's oplog:

```
./seedSecondary.sh <mongod-port> <oplog-size-in-gigabytes>
```

Step 5: Shut down the new secondary on the alternate port.

Step 6: Start up the new secondary.

```
mongod --port <secondary-port> --dbpath /data  --replSet REPLICASETNAME
```

Step 7: Add the new secondary to the replica set on the primary host.

```
rs.add("<secondary-host>:<secondary-port>")
```

6.7 Backup Maintenance

Configure Backup Data Delivery Select Backup delivery method and file format.

Delete Backup Snapshots Manually remove unneeded stored snapshots from Ops Manager.

Stop, Disable, Restart Backup Procedures to stop, restart, or disable backup.

Resync Backup If your Backup oplog has fallen too far behind your deployment to catch up, you must resync the Backup service.

Select Backup File Delivery Method and Format

On this page

Overview

With Backup, you can restore from a *stored snapshot* or, if you are restoring a replica set, you may build a *custom snapshot* reflecting a different point in the last 24 hours. For all backups, restoring from a stored snapshot is faster than restoring from a custom snapshot in the last 24 hours.

Once you select a backup-enabled sharded cluster or replica set to restore, the next step is to select the delivery method and file format.

Procedures

Select Backup File via Secure HTTP (HTTPS)

In the *Select Restore Destination* window, select *Pull via Secure HTTP (HTTPS)* to create a one-time direct download link.

Select Backup File via Secure Copy (SCP)

In the *Select Restore Destination* window, select *Push via Secure Copy (SCP)*. You can grant access by supplying Ops Manager with a username and password to your server, or you can provide a username and grant access via SSH public key.

To grant access via SSH public key:

Step 1: Select the *Administration* tab, and then select *Group Settings*.

Step 2: Scroll to the *Public Key for SCP Restores* setting.

Step 3: In the *Passphrase* box, enter a passphrase and then click the *Generate a New Public Key* button.

Backup generates and displays a public key.

Step 4: Log in to your server using the same username you will supply in your restore request.

Step 5: Add the public key to the `authorized_keys` file for that user.

The `authorized_keys` file is often located in the `~/.ssh` directory.

Important: For security reasons, you should remove this public key from the `authorized_keys` file once you have obtained your backup file.

Note: For security reasons, you should remove this public key from the authorized hosts file once you have obtained your backup file.

Select Backup File Format

In the *Select Restore Destination* window, select *Individual DB Files* or *Archive (tar.gz)* as the *Format*:

- Select *Individual DB Files* to transmit MongoDB data files produced by Ops Manager Backup directly to the target directory. The individual database files are faster for Backup to construct, but require additional file space on the destination server. The data *is* compressed during transmission.
- Select *Archive (tar.gz)* to deliver database files in a single `tar.gz` file you must extract before reconstructing databases.

Next Steps

Restore from a Stored Snapshot

Restore from a Point in the Last 24 Hours

Delete Snapshots for Replica Sets and Sharded Clusters

On this page

Overview

To delete snapshots for replica sets and sharded clusters, use the Ops Manager console to find then select a backup snapshot to delete.

Constraints

You can delete any replica set or sharded cluster snapshot if it is not needed for replica set point-in-time restores.

Procedure

Step 1: Click the Ops Manager Backup tab.

Step 2: Select type of backup file to delete.

Select either *Replica Set Status* or *Sharded Cluster Status*.

Step 3: Click the name of the replica set or sharded cluster.

Displays replica set or sharded cluster details with a list of possible backup files to delete.

Step 4: Select backup file to delete.

On the list of snapshots, click the *Delete* link to the right of a snapshot.

Step 5: Confirm deletion.

Click the OK button on the *Delete Snapshot* interface to confirm deletion of the backup.

Stop, Start, or Disable the Ops Manager Backup Service

On this page

- *Procedures*

Overview

Stopping the Ops Manager Backup Service for a replica set or sharded cluster suspends the service for that resource. Ops Manager stops taking new snapshots but retains existing snapshots until their listed expiration date.

After stopping backups, you can restart the Backup Service at any time. Depending how much time has elapsed, the Backup Service may perform an initial sync.

Disabling the Ops Manager Backup Service, by contrast, immediately deletes all snapshots. Later, if you want back up the cluster or replica set, when you enable backup, Ops Manager behaves as if the resource has never been backed up. Enabling backups on a previously disabled resource always requires an initial sync.

Procedures

Stop the Backup Service for a Cluster or Replica Set

Step 1: Click the *Backup* tab and then click either *Sharded Cluster Status* or *Replica Set Status*, depending on the resource to stop backup for.

Step 2: Click *Stop* for the cluster or replica set.

If prompted, enter the two-factor authentication code, click *Verify*, and click *Stop* again.

Restart the Ops Manager Backup Service

Step 1: Click the *Backup* tab and then click either *Sharded Cluster Status* or *Replica Set Status*, depending on the resource to re-enable.

Step 2: Click *Restart* for the cluster or replica set.

Step 3: Select a *Sync source* and click *Restart.*

Disable the Ops Manager Backup Service

Step 1: Stop and then terminate each sharded cluster enrolled in Backup.

In Ops Manager, click the *Backup* tab and select *Sharded Cluster Status*.

For each cluster enrolled in Backup, click *Stop*. If prompted, enter the two-factor authentication code, click *Verify*, and click *Stop* again.

When the *Terminate* button appears, click *Terminate*. Click *Terminate* again.

Step 2: Stop and then terminate each replica set enrolled in Backup.

In Ops Manager, click the *Backup* tab and select *Replica Set Status*.

For each replica set enrolled in Backup, click *Stop*. If prompted, enter the two-factor authentication code, click *Verify*, and click *Stop* again.

When the *Terminate* button appears, click *Terminate*. Click *Terminate* again.

Step 3: Stop all Backup Agents.

On Linux

Issue the following command:

```
service mongodb-mms-backup-agent stop
```

On Windows

In Windows `Control Panel`, open `Administrative Tools`, and then open `Services`.

In the list of services, select the `MMS Backup Agent` service. Select the `Action` menu and select `Stop`.

Resync Backup

On this page

Overview

When a backup becomes out of sync with the MongoDB deployment, Ops Manager produces a `Backup requires resync` alert. If you receive a `Backup requires resync` alert, you must resync the backup for the specified MongoDB instance.

The following scenarios trigger a `Backup requires resync` alert:

- **The Oplog has rolled over.** This is by far the most common case for the `Backup requires resync` alert and occurs whenever the Backup Agent's tailing cursor cannot keep up with the deployment's oplog. This is similar to when a secondary falls too far behind the primary in a replica set. Without a resync, the Backups will not catch up.
- **Unsafe applyOps**. This occurs when a document that Backup does not have a copy of is indicated.
- **Data corruption or other illegal instruction.** This typically causes replication, and therefore the backup job, to break. When the daemon sees the broken job, it requests a resync.

During the resync, data is read from a secondary in each replica set and Ops Manager does not produce any new snapshots.

Note: For production deployments, you should resync all backups annually.

Considerations

To avoid the need for resyncs, ensure the Backup oplog does not fall behind the deployment's oplog. This requires that:

- adequate machine resources are provisioned for the agent, and that
- you restart the Ops Manager agents in a timely manner following maintenance or other downtime.

To provide a buffer for maintenance and for occasional activity bursts, ensure that the oplog on the primary is large enough to contain at least 24 hours of activity. For more information on the Backup oplog, see the *Backup FAQs*.

Procedures

Use the appropriate procedure below for your deployment type, sharded cluster or replica set.

Resync a Sharded Cluster

Step 1: Select the *Backup* tab and then select *Sharded Cluster Status*.

Step 2: Select *See Shards/Configs*.

The column expands to display the shards and config servers.

Step 3: Resync.

For the MongoDB instance to resync:

Click the *resync* link.

Accept the default selection in the *Sync source* drop-down list, and click the *RESYNC* button.

If prompted for a two-factor verification code, enter the code and click *VERIFY*. Then click the *RESYNC* button again.

Resync a Replica Set

Step 1: Select the *Backup* tab and then select *Replica Set Status*.

Step 2: In the replica set's *LAST SYNC* column, select *resync*.

Step 3: Select a *Sync source* and click the *RESYNC* button.

If prompted for a two-factor verification code, enter the code and click *VERIFY*. Then click the *RESYNC* button again.

7 Security

Security Options & Support Matrices Describes Ops Manager security features.

Firewall Configuration Describes the ports used by Ops Manager components.

Change the Ops Manager Ports Describes how to change the default ports used by Ops Manager on Windows and Unix-based systems.

SSL Connections with Ops Manager Run the Ops Manager Application over HTTPS and require users to have a valid certificate to connect to the Ops Manager web interface.

SSL Connections with Backing Instances Configure SSL connections to the backing MongoDB processes that host the databases that support Ops Manager.

SSL Connections with MongoDB Deployments Specify SSL use for host.

LDAP for Ops Manager Users Configure Ops Manager to use LDAP to store user data and permissions.

Configure Authentication for MongoDB Deployments Edit authentication credentials for host.

System-wide Two-Factor Authentication Configure two-factor authentication.

Manage Your Account's Two-Factor Authentication Describes two-factor authentication, which is required for Ops Manager.

7.1 Security Overview

Overview

Ops Manager provides security options to ensure the security of your Ops Manager agents, Ops Manager servers, and MongoDB deployments. Ops Manager supports the options described in the tables on this page.

Security Options

	Connections with Ops Manager	Connections with Ops Manager *Backing Instances*	Connections with MongoDB Deployments
Ops Manager	*not applicable*	• *SSL* • *MONGODB-CR* • *LDAP* • *Kerberos*	Ops Manager connects to MongoDB through the Monitoring, Backup, and Automation agents.
Monitoring Agent	• *SSL*	*not applicable*	• *SSL* • *MONGODB-CR* • *LDAP* • *Kerberos*
Backup Agent	• *SSL*	*not applicable*	• *SSL* • *MONGODB-CR* • *LDAP* • *Kerberos*
Automation Agent	• *SSL*	*not applicable*	• *MONGODB-CR*
Ops Manager user	• *SSL* • *Ops Manager access control* • *LDAP*	*not applicable*	For user access to MongoDB, see Authentication in the MongoDB manual.

Supported User Authentication Per Release

The following table shows the available user authentication mechanisms and the release the mechanism became available.

Method	Available beginning with...
Authentication against Ops Manager Application database	OnPrem 1.0
Authentication against LDAP	OnPrem 1.4

Supported MongoDB Security Features on Linux

This section describes supported security options on Linux.

Connections Between Ops Manager Servers and Backing Replica Sets

The following table shows the authentication and SSL options available between Ops Manager and the *Ops Manager Application Database* and *Backup Blockstore Database*. These options do not apply to the HEAD databases that reside on the Backup Daemon:

Username/Password Authentication	MongoDB SSL Connections	Kerberos Authentication	MongoDB SSL Connections with Client Certificates	x509 Authentication	LDAP Authentication
OnPrem 1.0+	Ops Manager 1.6+	OnPrem 1.3	Ops Manager 1.6+	Ops Manager 1.6+	OnPrem 1.5+

Connections Between Agents and MongoDB Deployments

The following table shows the authentication and SSL options available between the Ops Manager agents and the MongoDB deployments they manage and back up:

	Username/Password Authentication	MongoDB SSL Connections	Kerberos Authentication	MongoDB SSL Connections with Client Certificates	x509 Authentication	LDAP Authentication
Monitoring Agent	OnPrem 1.0	OnPrem 1.0	OnPrem 1.3	OnPrem 1.5+		OnPrem 1.5+
Backup Agent	OnPrem 1.4	OnPrem 1.4	OnPrem 1.4.1	OnPrem 1.5+		OnPrem 1.5+
Automation Agent	Ops Manager 1.6+					

Supported MongoDB Security Features on Windows

This section describes supported security options on Windows.

Connections Between Ops Manager Servers and Backing Replica Sets

The following table shows the authentication and SSL options available between Ops Manager and the *Ops Manager Application Database* and *Backup Blockstore Database*. These options do not apply to the HEAD databases that reside on the Backup Daemon:

Username/Password Authentication	MongoDB SSL Connections	Kerberos Authentication	MongoDB SSL Connections with Client Certificates	x509 Authentication	LDAP Authentication
OnPrem 1.5+	Ops Manager 1.6+		Ops Manager 1.6+	Ops Manager 1.6+	OnPrem 1.5+

Connections Between Agents and MongoDB Deployments

The following table shows the authentication and SSL options available between the Ops Manager agents and the MongoDB deployments they manage and back up:

	Username/Password Authentication	MongoDB SSL Connections	Kerberos Authentication	MongoDB SSL Connections with Client Certificates	x509 Authentication	LDAP Authentication
Monitoring Agent	OnPrem 1.5+	OnPrem 1.5+		OnPrem 1.5+		OnPrem 1.5+
Backup Agent	OnPrem 1.5+	OnPrem 1.5+		OnPrem 1.5+		OnPrem 1.5+
Automation Agent	Ops Manager 1.6+					

7.2 Firewall Configuration

Overview

Ops Manager components use the default ports and health-check endpoints described here.

Ports

Although the following components are logically distinct, the *Ops Manager HTTP Service*, *Backup HTTP Service*, and *Backup Alert Service* are all part of the Ops Manager Application and typically run on a single system. The Backup Daemon typically runs on a distinct system.

Ops Manager HTTP Service

`8080`
: Web server, available from browsers. The system running the Monitoring Agent must be able to access this port as well.

Backup HTTP Service

`8081`
: Available from browsers for HTTP restores, and from systems running the Backup Agent.

`8091`
: *Optional*

  Used for internal diagnostics. Only available on the localhost interface.

Backup Alert Service

`8650`
: A "kill-port" that the control script uses to signal a shutdown.

  Only available on the localhost interface.

8092

Optional

Used for internal diagnostics. Only available on the localhost interface.

Backup Daemon

8640

A "kill-port" that the control script uses to signal a shutdown.

Only available on the localhost interface.

27500–27503

The Backup Daemon uses this port range to on the localhost interface to run `mongod` instances to apply oplog entries to maintain the local copies of the backed up database.

8090

Optional

Used for internal diagnostics. Only available on the localhost interface.

Monitoring HTTP Endpoints

Ops Manager provides health-check endpoints for the monitoring of the Ops Manager components via a standard monitoring service, such as Zabbix or Nagios. These endpoints are only accessible on the `localhost` interface.

Backup HTTP Service Endpoint

The *Backup HTTP Service* on the Ops Manager Application exposes the following endpoint:

```
http://localhost:8091/health
```

The endpoint checks the connections from the service to the *Ops Manager Application Database* and the *Backup Blockstore Database*.

A successful response from the endpoint returns the following:

```
{
  "mms_db": "OK",
  "backup_db": "OK"
}
```

Backup Alert Service Endpoint

The *Backup Alert Service* on the *Ops Manager Application* exposes the following health-check endpoint:

```
http://localhost:8092/health
```

Backup Daemon Endpoint

The *Backup Daemon* on the Backup Daemon server exposes a health-check endpoint at:

```
http://localhost:8090/health
```

7.3 Change the Ops Manager Ports

On this page

Overview

By default, Ops Manager uses port `8080` for the Ops Manager HTTP Service and port `8081` for the Backup HTTP Service.

To use different ports, you need to configure Ops Manager to use the ports, update *`mms.centralUrl`* and *`mms.backupCentralUrl`*, and restart Ops Manager.

Procedures

Change the Default Port on Unix-based Systems

To change the ports on Unix-based operating systems, you must edit the port settings in `<install-directory>/conf/mms.conf`, update the *`mms.centralUrl`* and *`mms.backupCentralUrl`* values in the conf-mms-properties value, and restart Ops Manager.

Step 1: Open `mms.conf` for editing.

Open the `mms.conf` file with root access. `mms.conf` is located in the `<install_dir>/conf/` directory.

Step 2: Set the `BASE_PORT` value to the port Ops Manager should use.

By default, Ops Manager uses port `8080`. Change the `BASE_PORT` value to the desired port number.

```
BASE_PORT=11700
```

If you wish to change the port for Ops Manager connections over SSL, update `BASE_SSL_PORT`.

When changing the port, ensure that the chosen port is available for use.

Step 3: Set the `BACKUP_BASE_PORT` value to the port the Backup HTTP Service should use.

By default, the Backup HTTP Service uses port `8081`. To change the port, set `BACKUP_BASE_PORT` to the desired port number.

```
BACKUP_BASE_PORT=11701
```

If you wish to change the port for Ops Manager connections over SSL, update `BACKUP_BASE_SSL_PORT`.

When changing the port, ensure that the chosen port is available for use. You must use different ports for the `BASE_PORT` and `BACKUP_BASE_PORT`.

Step 4: Save the changes.

Step 5: Edit `mms.centralUrl` and `mms.backupCentralUrl` to use the new ports.

Open conf-mms.properties with root access for editing.

Set *`mms.centralUrl`* to the new port specified by the `BASE_PORT`:

```
mms.centralUrl=http://mms.example.com:11700
```

Set `mms.backupCentralUrl` to use the new port specified by the `BACKUP_BASE_PORT`:

```
mms.backupCentralUrl=http://mms.example.com:11701
```

Save the changes, and close the file.

Step 6: Restart Ops Manager.

Depending on how you installed Ops Manager, the command to restart Ops Manager will vary.

Installation with `DEB` or `RPM` package.

If you installed using a `DEB` or `RPM` package, use the following command to restart Ops Manager:

```
sudo service mongodb-mms restart
```

Installation from `.tar.gz` archive.

If you installed using a `.tar.gz` archive, use the following command to restart Ops Manager:

```
sudo /etc/init.d/mongodb-mms restart
```

Ops Manager will restart. The Ops Manager interface should be accessible at the new URL and port combination.

Change the Default Port on a Windows System

To change the ports on Windows, you need to edit two registry key values in the Windows registry, update the *`mms.centralUrl`* and *`mms.backupCentralUrl`* values in the *conf-mms.properties* file, and restart Ops Manager.

Step 1: Open the Registry Editor.

From the *Start* menu, click *Run*, and then type `regedit`, and click *OK*. The Registry Editor will open.

Step 2: Set the port that Ops Manager should use.

By default, Ops Manager uses port `8080`. You can change the port by editing the relevant registry value.

Open the following registry value:

```
SOFTWARE\Wow6432Node\Apache Software Foundation\Procrun 2.
↪0\MMS\Parameters\Java\Options
```

Step 3: Add `-Dbase-port=<portnumber>` to the `Options` registry value, setting the value to the desired port.

Add a new line in the `Options` file that resembles the following:

```
-Dbase-port=11700
```

When changing the port, ensure that the chosen port is available.

Step 4: To change the Backup HTTP Service's port, open the `\MMSBSlurp\Parameters\Java\Options` registry value.

Open the following registry value:

```
SOFTWARE\Wow6432Node\Apache Software Foundation\Procrun 2.
↪0\MMSBSlurp\Parameters\Java\Options
```

Step 5: Edit `-Dbase-port=8081` to point to the desired port.

`-Dbase-port=8081` is already listed in the `Options` registry value. Edit it to use the desired port:

```
-Dbase-port=11701
```

When changing the port, ensure that the chosen port is available.

Step 6: When you are done editing the settings, click *OK*, and exit the Registry Editor.

Step 7: Edit `mms.centralUrl` and `mms.backupCentralUrl` to use the new ports.

Open *conf-mms.properties* with root access for editing.

Update `mms.centralUrl` to use the new port specified by `-DBase-port` in the `\MMS\Parameters\Java\Options` registry value:

```
mms.centralUrl=http://mms.example.com:11700
```

Update `mms.backupCentralUrl` to use the new port specified by `-DBase-port` in the `\MMSBSlurp\Parameters\Java\Options` registry value:

```
mms.backupCentralUrl=http://mms.example.com:11701
```

Save the changes, and close the file.

Step 8: Restart Ops Manager.

To restart the Ops Manager HTTP service, open the *Control Panel*, then *System and Security*, then *Administrative Tools*, and finally, double click to open the *Services* window.

In the *Services* list, restart the `MongoDB Ops Manager HTTP Service` and `MongoDB Backup HTTP Service` by right-clicking on them in the list, and choosing *Restart*.

Ops Manager will restart. The Ops Manager interface should be accessible at the new URL and port combination.

7.4 Configure SSL Connections to Ops Manager

On this page

Overview

You can encrypt connections from the Monitoring and Backup Agents to Ops Manager and from website clients to the Ops Manager web interface. You can encrypt connections in either of two ways:

- Set up an HTTPs proxy in front of Ops Manager.
- Run the Ops Manager Application over HTTPS, as described here.

Run the Ops Manager Application Over HTTPS

To run the Ops Manager Application over HTTPS, you can configure Ops Manager with the server's certificate and private key.

Step 1: Open the `conf-mms.properties` file with root privileges.

For the file location, see *Ops Manager Configuration Files*.

Step 2: Configure the `mms.https.PEMKeyFile` and `mms.https.PEMKeyFilePassword` settings.

Set *`mms.https.PEMKeyFile`* to a PEM file containing an X509 certificate and private key, and set *`mms.https.PEMKeyFilePassword`* to the password for the certificate. For example:

```
mms.https.PEMKeyFile=<path_and_name_of_pem_file>
mms.https.PEMKeyFilePassword=<password_for_pem_file>
```

Set *mms.centralUrl* to the new HTTPS information. For example:

```
mms.centralUrl=https://mms.example.net:8443
```

8443 is the default port used by Ops Manager.

7.5 Configure the Connections to the Backing MongoDB Instances

On this page

Overview

If you set up your *backing MongoDB instances* to use access control or to run over SSL, then you must update Ops Manager's configuration files with the necessary information for accessing the MongoDB instances.

The conf-mms.properties file configures the connection from Ops Manager to the Ops Manager Application database.

If you are using Backup, you also need to update the conf-daemon.properties file so that the Backup Daemon can connect to the Ops Manager Application Database, and configure the Backup databases in the Ops Manager UI.

Prerequisites

This tutorial assumes that you have deployed the *Ops Manager Application Database* and *Backup Blockstore Database*, and that you have configured them to use access control and/or SSL. For information on deploying MongoDB with access control or to use SSL, see Security Concepts in the MongoDB manual.

Procedures

Configure Ops Manager to Connect to Backing Databases with Access Control

MongoDB deployments can use MONGODB-CR/SCRAM-SHA-1 username-password authentication, Kerberos authentication, LDAP authentication, or x.509 Client Certificate authentication to manage access to the database.

If your Ops Manager Application database uses access control, you must configure Ops Manager and the Backup Daemon to be able to connect to the database.

Step 1: Open the Ops Manager configuration files with root privileges.

Open the files listed below. *Ops Manager Configuration Files* describes the locations of each file:

- `conf-mms.properties`, which configures Ops Manager's connection to the *Ops Manager Application Database*.
- **Optional** `conf-daemon.properties`, which configures the Backup Daemon's connection to the Ops Manager Application Database. You only need to open `conf-daemon.propeties` if you have installed the *Backup Daemon*.

Step 2: Configure Ops Manager to connect to the Ops Manager Application database.

`mongo.mongoUri` contains the connection string used to access the *Ops Manager Application Database*.

The *`mongo.mongoUri`* reference provides examples of the connection string format for each authentication mechanism and details the required permissions for the connecting user.

For an Ops Manager Application database using Kerberos authentication, the *`mongo.mongoUri`* setting would resemble:

```
mongo.mongoUri=mongodb://username%40REALM.example.net@mydb1.example.net:40000/?
↪authMechanism=GSSAPI
```

Step 3: Optional: Configure the Backup Daemon to connect to the Ops Manager Application database.

If you have installed the Backup Daemon, you must configure *`mongo.mongoUri`* in `conf-daemon.properties` so that the Backup Daemon can connect to the Ops Manager Application database.

`mongo.mongoUri` contains the connection string used to access the *Ops Manager Application Database*.

The *`mongo.mongoUri`* reference provides examples of the connection string format for each authentication mechanism and details the required permissions for the connecting user.

For an Ops Manager Application database using Kerberos authentication, the *`mongo.mongoUri`* setting would resemble:

```
mongo.mongoUri=mongodb://username%40REALM.example.net@mydb1.example.net:40000/?
↪authMechanism=GSSAPI
```

`mongo.mongoUri` should be idential in both `conf-mms.properties` and `conf-daemon.properties`.

Step 4: Configure any other authentication mechanism-specific settings in both `conf-mms.properties` and `conf-daemon.properties`.

If you are using Kerberos authentication, you must *configure the Kerberos settings*, as in the following:

```
jvm.java.security.krb5.kdc=kdc.example.com
jvm.java.security.krb5.realm=EXAMPLE.COM
mms.kerberos.principal=mms/mmsweb.example.com@EXAMPLE.COM
mms.kerberos.keyTab=/path/to/mms.keytab
```

If you are using x.509 Client Certificate Authentication, you must also be connecting over SSL. See: *Configure SSL Connections to the Ops Manager Application Database* for the SSL configuration instructions.

Step 5: Restart Ops Manager and the Backup Daemon.

If the Ops Manager Application database is running over SSL, proceed to the *SSL configuration tutorial*.

Restart Ops Manager and the Backup Daemon using the appropriate command for your distribution:

Installed on Linux with DEB or RPM packages:

```
sudo service mongodb-mms restart
sudo service mongodb-mms-backup-daemon restart
```

Installed on Linux from an Archive:

```
<install_dir>/bin/mongodb-mms restart
<install_dir>/bin/mongodb-mms-backup-daemon restart
```

Installed on Windows:

- Open `Control Panel`, then `System and Security`, then `Administrative Tools`, and then `Services`. In the Services list, right-click on the `MongoDB Ops Manager HTTP Service` and select `Restart`.
- On the Backup Daemon server, open `Control Panel`, then `System and Security`, then `Administrative Tools`, and then `Services`. Right-click on the `MMS Backup Daemon Service` and select `Restart`.

Configure SSL Connections to the Ops Manager Application Database

Step 1: Open the Ops Manager configuration files with root privileges.

Open the files listed below. *Ops Manager Configuration Files* describes the locations of each file:

- `conf-mms.properties`, which configures Ops Manager's connection to the *Ops Manager Application Database*.
- **Optional** `conf-daemon.properties`, which configures the Backup Daemon's connection to the Ops Manager Application Database. You only need to open `conf-daemon.propeties` if you have installed the *Backup Daemon*.

Step 2: Configure Ops Manager to connect to the Ops Manager Application database over SSL.

Configure the following settings in `conf-mms.properties`:

`mongo.ssl`: Set this to `true` to indicate that the *Ops Manager Application Database* is using SSL.

`mongodb.ssl.CAFile`: Specify the PEM file that contains the root certificate chain from the Certificate Authority that signed the MongoDB server certificate.

`mongodb.ssl.PEMKeyFile`: If the MongoDB instance is running with `--sslCAFile` option, specify the PEM file containing an x.509 certificate and private key.

`mongodb.ssl.PEMKeyFilePassword`: If the client PEM file contains an encrypted private key, specify the password for PEM file. To encrypt this password in the configuration file, use the Ops Manager `credentialstool` tool. See *Encrypt MongoDB User Credentials*.

Step 3: Optional: Configure the Backup Daemon to connect to the Ops Manager Application database over SSL.

If you are using Backup, set the following settings in `conf-daemon.properties`:

Set *`mongo.ssl`* to `true`. This setting should match the *`mongo.ssl`* setting in `conf-mms.properties`.

Update the SSL settings with the SSL client certificate information to use to connect to the backing databases. *`mongodb.ssl.CAFile`*, *`mongodb.ssl.PEMKeyFile`*, and *`mongodb.ssl.PEMKeyFilePassword`* should match the settings in `conf-mms.properties`.

Step 4: Restart Ops Manager and the Backup Daemon.

Restart Ops Manager and the Backup Daemon using the appropriate command for your distribution:

Installed on Linux with DEB or RPM packages:

```
sudo service mongodb-mms restart
sudo service mongodb-mms-backup-daemon restart
```

Installed on Linux from an Archive:

```
<install_dir>/bin/mongodb-mms restart
<install_dir>/bin/mongodb-mms-backup-daemon restart
```

Installed on Windows:

- Open `Control Panel`, then `System and Security`, then `Administrative Tools`, and then `Services`. In the Services list, right-click on the `MongoDB Ops Manager HTTP Service` and select `Restart`.
- On the Backup Daemon server, open `Control Panel`, then `System and Security`, then `Administrative Tools`, and then `Services`. Right-click on the `MMS Backup Daemon Service` and select `Restart`.

7.6 Configure SSL for MongoDB

On this page

Overview

If your MongoDB deployment uses SSL, then you must configure the *Use SSL* setting for the deployment, as described here.

You also must configure the agents' SSL settings, as described in *Configure Monitoring Agent for SSL* and *Configure Backup Agent for SSL*.

Procedures

Edit Host SSL Use for Monitoring

Before editing these credentials, update the agent's configuration file to use SSL. *Configure Monitoring Agent for SSL*.

To specify SSL use:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Select the process's gear icon and select *Edit Host*.

Step 3: Select *SSL*.

Step 4: Turn `ON` or `OFF`.

Step 5: Close the dialog.

Edit Host Credential Information for Backup

Before editing these credentials, update the agent's configuration file to use SSL. *Configure Backup Agent for SSL*.

To specify SSL use:

Step 1: Select the *Backup* tab and then select *Replica Set Status* or *Sharded Cluster Status*.

Step 2: On the line listing the cluster or replica set, click the gear icon.

Step 3: Select *SSL*.

Step 4: Turn `ON` or `OFF`

Step 5: Close the dialog.

7.7 Configure Users and Groups with LDAP for Ops Manager

On this page

Important: You must start with a fresh installation of the latest version of Ops Manager. There is no method for converting an existing Ops Manager deployment to use LDAP user management.

Overview

You can configure Ops Manager to use a Lightweight Directory Access Protocol (LDAP) service to manage user authentication. Users log in using the standard Ops Manager interface. Ops Manager synchronizes user name and email addresses in Ops Manager user records with the values in LDAP user records.

Note: If your MongoDB deployment uses LDAP, you must also create MongoDB users for the Ops Manager agents. For more information, see *Configure Monitoring Agent for LDAP* and *Configure Backup Agent for LDAP Authentication*.

Additionally, and independently, the Ops Manager Application can also use LDAP to authenticate to its *backing instances*. To use LDAP with backing instances, set the `mongo.mongoUri` in addition to the *LDAP settings* described in the procedure below.

User Authentication

When a user logs in, Ops Manager searches LDAP for a matching user. To perform the search, Ops Manager logs into LDAP as the "search" user, using the credentials specified in the `mms.ldap.bindDn` and `mms.ldap.bindPassword` settings in the Ops Manager *configuration file*.

Ops Manager searches LDAP within the base distinguished name defined in the `mms.ldap.user.baseDn` setting of the Ops Manager configuration file and matches the user according to the LDAP attribute defined in the `mms.ldap.user.searchAttribute` setting. If a matching user is found, Ops Manager authenticates the supplied password against the LDAP password.

Access Control

LDAP groups let you control access to Ops Manager. You map LDAP groups to Ops Manager roles and assign the LDAP groups to the users who should have those roles.

To use LDAP groups effectively, *create additional groups* within Ops Manager to control access to specific deployments in your organization. For example, create separate Ops Manager groups for development environments and for production environments. Provide access to a deployment by mapping an LDAP group to a role in the Ops Manager group.

This tutorial describes how to map LDAP groups to global Ops Manager roles and to group-level Ops Manager roles. You do the former through the Ops Manager *configuration file* and the latter through the Ops Manager interface.

Note: Changes made to LDAP groups can take up to an hour to appear in Ops Manager.

Global Owner Access

Your LDAP installation must include a group that you can assign to the Ops Manager `mms.ldap.global.role.owner` setting. The first user to log into Ops Manager with LDAP authentication must belong to this LDAP group. This user will also create the initial Ops Manager group.

For example, if LDAP has an `admin` group for use by Ops Manager administrators, set the `mms.ldap.global.role.owner` property to `admin` during the appropriate step in the procedure below.

Prerequisites

The *Ops Manager Application* must be installed and configured. You must either start with a new Ops Manager installation or reset your installation to a clean state. For assistance, contact your MongoDB account manager.

The LDAP server must be installed, configured, and accessible to Ops Manager.

Procedure

To configure LDAP authentication, define user records in LDAP, configure *LDAP settings* in Ops Manager, and then associate LDAP groups with Ops Manager group-level roles, as described here:

Step 1: Define LDAP user records.

Step 2: Update the service class setting in `conf-mms.properties`.

Set the *`mms.userSvcClass`* setting in the Ops Manager *conf-mms.properties configuration file* to the value of the LDAP service class. For example:

```
mms.userSvcClass=com.xgen.svc.mms.svc.user.UserSvcLdap
```

Step 3: Enter LDAP user settings in `conf-mms.properties`.

Enter values for the following settings. The LDAP group you specify for the *`mms.ldap.global.role.owner`* setting must have a user assigned to it. You will sign in as this user later in this procedure. For more information on setting or an example value, click the setting or see *LDAP User Settings*:

- *`mms.ldap.url`*.
- *`mms.ldap.bindDn`*.
- *`mms.ldap.bindPassword`*.
- *`mms.ldap.user.baseDn`*.
- *`mms.ldap.user.searchAttribute`*.
- *`mms.ldap.user.group`*.
- *`mms.ldap.global.role.owner`*

Optional settings:

- *`mms.ldap.user.firstName`*.
- *`mms.ldap.user.lastName`*.
- *`mms.ldap.user.email`*.

Step 4: Associate LDAP groups with global Ops Manager roles in `conf-mms.properties`.

In addition to the *`mms.ldap.global.role.owner`* setting that you assigned in the previous step, assign the remaining global-role settings. Each provides the members of the specified LDAP group or groups with an Ops Manager *global role*. Global roles provide access to all the Ops Manager *groups* in the Ops Manager deployment.

You can specify an LDAP group using any format, including Common Name (`cn`) or Distinguished Name (`dn`). The format you choose must be consistent across settings and consistent with the format used in the LDAP user records in the attribute specified in the *`mms.ldap.user.group`* setting.

For more information and example values, click the settings or see *LDAP Global Role Settings*:

- *`mms.ldap.global.role.automationAdmin`*
- *`mms.ldap.global.role.backupAdmin`*
- *`mms.ldap.global.role.monitoringAdmin`*
- *`mms.ldap.global.role.userAdmin`*
- *`mms.ldap.global.role.readOnly`*

Optional:

- *`mms.ldap.group.separator`*

Step 5: Log in as a global owner and create the first Ops Manager group.

Log into Ops Manager as an LDAP user that is part of the LDAP group specified in the Ops Manager *`mms.ldap.global.role.owner`* setting.

Upon successful login, Ops Manager displays your groups page.

Step 6: Associate LDAP groups with group-level roles.

On the Ops Manager *My Groups* page (in the *Administration* tab), click the *Add Group* button.

Enter a name for the new Ops Manager group and enter the LDAP groups that should provide the permissions for each *group-level role*.

Select the checkbox to agree to the terms of service.

Click *Add Group*.

Step 7: Add your MongoDB deployments.

Specify the LDAP authentication settings when *adding a MongoDB deployment*.

7.8 Configure MongoDB Authentication and Authorization

On this page

Overview

Your MongoDB deployments can use the access control mechanisms described here. You specify the authentication settings when *adding the deployment*. You can also edit settings after adding a deployment, as described on this page.

If a deployment uses access control, the Monitoring and Backup Agents must authenticate to the deployment as MongoDB users with appropriate access. See the following:

- *Required Access for Monitoring Agent*
- *Required Access for Backup Agent*

Access Control Mechanisms

MONGODB-CR

MongoDB Challenge-Response (`MONGODB-CR`) is the MongoDB default for authentication and authorization. To enable, see *MongoDB access control*.

To create MongoDB users for the Ops Manager agents, see *Add Monitoring Agent User for MONGODB-CR* and *Configure Backup Agent for MONGODB-CR*.

To configure the Ops Manager Application to authenticate to the *backing instances* using `MONGODB-CR`, see `mongo.mongoUri`.

LDAP

To use LDAP for access control, see *LDAP*.

Kerberos

If your MongoDB deployment uses Kerberos for authentication, you must create the Kerberos Principal for the Ops Manager agents, create a MongoDB user for that Kerberos Principal, edit the agent's configuration file. If you are running both the Monitoring Agent and the Backup Agent on the same server, then both agents must connect as the same Kerberos Principal.

To create a Kerberos Principal and the associated MongoDB user as well as edit the configuration file, see *Configure the Monitoring Agent for Kerberos* and *Configure the Backup Agent for Kerberos*.

Specify Kerberos as the MongoDB process's authentication mechanism when *adding the deployment* or in the procedure on this page for editing a deployment.

To enable Kerberos authentication between the Ops Manager Application and the *Ops Manager Application Database*, configure the *Kerberos Settings* and the `mongo.mongoUri` setting. You must configure all required Kerberos settings to enable Kerberos authentication.

Edit Host Credentials

If you configure authentication credentials, the Ops Manager agents must authenticate to the deployment as a MongoDB user with the proper access. Configure the agents with the proper credentials **before** configuring the credentials on the deployment. See:

- *Configure Monitoring Agent for Access Control*.
- *Configure Backup Agent for Access Control*.

Edit Host Credential Information for Monitoring

Before editing these credentials, set up the agent as a user in MongoDB with appropriate access. See *Configure Monitoring Agent for Access Control*.

To edit authentication credentials:

Step 1: Select the *Deployment* tab and then the *Deployment* page.

Step 2: Select the process's gear icon and select *Edit Host*.

Step 3: Select the *Credentials* tab.

Step 4: At the bottom of the dialog box, click the *Change* button.

Step 5: Enter the credentials.

Edit the following information, as appropriate:

Auth Mechanism	The authentication mechanism used by the host. Can specify *MONGODB-CR*, *LDAP (PLAIN)*, or *Kerberos(GSSAPI)*.
Current DB Username	If the authentication mechanism is MONGODB-CR or LDAP, the username used to authenticate the Monitoring Agent to the MongoDB deployment. See *Add Monitoring Agent User for MONGODB-CR*, *Configure Monitoring Agent for LDAP*, or *Configure the Monitoring Agent for Kerberos* for setting up user credentials.
Current DB Password	If the authentication mechanism is MONGODB-CR or LDAP, the password used to authenticate the Monitoring Agent to the MongoDB deployment. See *Add Monitoring Agent User for MONGODB-CR*, *Configure Monitoring Agent for LDAP*, or *Configure the Monitoring Agent for Kerberos* for setting up user credentials.
Update other hosts in replica set/sharded cluster as well	Only for cluster or replica set. If checked, apply the credentials to all other hosts in the cluster or replica set.

Step 6: Click the *Submit* button.

Step 7: Close the dialog box.

Edit Host Credential Information for Backup

Before editing these credentials, set up the agent as a user in MongoDB with appropriate access. See *Configure Backup Agent for Access Control*.

To edit authentication credential information:

Step 1: Select the *Backup* tab and then select *Replica Set Status* or *Sharded Cluster Status*.

Step 2: On the line listing the cluster or replica set, click the gear icon.

Step 3: Select *Edit Credentials*.

Step 4: Enter the credentials.

Edit the following information, as appropriate:

Auth Mechanism	The authentication mechanism used by the host. Can specify *MONGODB-CR*, *LDAP (PLAIN)*, or *Kerberos(GSSAPI)*.
Current DB Username	If the authentication mechanism is MONGODB-CR or LDAP, the username used to authenticate the Monitoring Agent to the MongoDB deployment. See *Configure Backup Agent for MONGODB-CR*, *Configure Backup Agent for LDAP Authentication*, or *Configure the Backup Agent for Kerberos* for setting up user credentials.
Current DB Password	If the authentication mechanism is MONGODB-CR or LDAP, the password used to authenticate the Monitoring Agent to the MongoDB deployment. See *Configure Backup Agent for MONGODB-CR*, *Configure Backup Agent for LDAP Authentication*, or *Configure the Backup Agent for Kerberos* for setting up user credentials.
My deployment supports SSL for MongoDB connections	If checked, the Monitoring Agent must have a trusted CA certificate in order to connect to the MongoDB instances. See *Configure Monitoring Agent for SSL*.

Step 5: Click *Save*.

7.9 Manage Two-Factor Authentication for Ops Manager

On this page

Overview

When enabled, two-factor authentication requires a user to enter a verification code to log in and to perform certain protected operations. Operations that require two-factor authentication include:

- restoring and deleting snapshots,
- stopping and terminating Backup for a sharded cluster or replica set,
- inviting and adding users,
- generating new two-factor authentication backup codes, and
- saving phone numbers for two-factor authentication.

Administrators with access to the Ops Manager Application's `<install_dir>/conf/conf-mms.properties` *configuration file* on your servers can enable two-factor authentication through the file's `mms.multiFactorAuth.require` setting. Administrators can also enable two-factor authentication to use Twilio to send verification codes to uses.

Users configure two-factor authentication on their accounts through their Ops Manager *user profiles*, where they select whether to receive their verification codes through voice calls, text messages (SMS), or the Google Authenticator application. If your organization does not use Twilio, then users can receive codes only through Google Authenticator.

Administrators can reset accounts for individual users as needed. Reseting a user's account clears out the user's existing settings for two-factor authentication. When the user next performs an action that requires verification, Ops Manager forces the user to re-enter settings for two-factor authentication.

Procedures

Enable Two-factor Authentication

Step 1: Open the Ops Manager Application's `conf-mms.properties` file.

The `conf-mms.properties` file is located in the `<install_dir>/conf/` directory. See *Ops Manager Configuration Files* for more information.

Step 2: Set the `mms.multiFactorAuth.require` property to `true`.

```
mms.multiFactorAuth.require=true
```

Step 3: Restart the Ops Manager Application.

```
sudo service mongodb-mms start
```

Enable Twilio Integration

Step 1: Configure Twilio integration.

Configure Twilio integration through the *Twilio settings* in the Ops Manager Application's `conf-mms.properties` file.

Step 2: Restart the Ops Manager Application.

For example:

```
sudo service mongodb-mms start
```

Reset a User's Two-factor Authentication Account

Reseting the user's account clears out any existing two-factor authentication information. The user will be forced to set it up again at the next login.

You must have the `global user admin` or `global owner` *role* to perform this procedure.

Step 1: Open Ops Manager Administration.

To open Administration, click the *Admin* link in the Ops Manager banner.

Step 2: Select the *Users* page.

Step 3: Locate the user and click the pencil icon on the user's line.

Step 4: Select the *Clear Two Factor Auth* checkbox.

7.10 Manage Your Two-Factor Authentication Options

On this page

Overview

When enabled, Ops Manager requires two-factor authentication to help users control access to their Ops Manager accounts. To log into Ops Manager, a user must provide their password (i.e. "something you know"), as well as a second time-sensitive verification code, delivered during authentication (i.e. "something you have"). By requiring both factors, Ops Manager can grant authentication requests with a higher degree of confidence.

Ops Manager users receive verification codes through text messages (SMS), automated voice calls or an application that implements the Time-based One-time Password Algorithm (TOTP), such as the Google Authenticator application. Users can configure two-factor authentication mechanisms when signing up for Ops Manager or in the *Administration* tab's *Account* page in Ops Manager.

Authentication with Text or Voice Messages

Users can receive verification codes through text or voice by providing phone numbers when setting up their Ops Manager profiles. When a user needs a code, Ops Manager sends the code using text (SMS) or through an automated phone call that reads out the code.

Certain network providers and countries may impose delays on SMS messages. Users who experience delays should consider Google Authenticator for verification codes.

Note: From India, use Google Authenticator for two-factor authentication. Google Authenticator is more reliable than authentication with SMS text messages to Indian mobile phone numbers (i.e. country code 91).

Authentication with Applications

Authentication using Google Authenticator

Google Authenticator is a smartphone application that uses TOTP to generate verification codes. When a user needs a code, the application generates a time-based code based on a private key that was shared between Ops Manager and the user's Google Authenticator application during the initial pairing process.

The Google Authenticator application **does not** require a Google account and does not connect a user's Ops Manager account to Google in any way. The has both iOS and Android versions, and the user does not need to associate the application with a Google account. Ops Manager two-factor authentication using Google Authenticator is not in any way integrated with Google's own account authentication mechanisms, and Ops Manager does **not** provide two-factor authentication codes to Google.

Authentication using Another Implementation of TOTP

There are implementations of the Time-based One-time Password Algorithm (TOTP) other than Google Authenticator. For example, the Authenticator application for Windows Phones.

Ensure that whichever devices runs the TOTP application has it's own set of robust authentication requirements.

For other implementations of TOTP, consider the list of TOTP implementations on Wikipedia.

Two-Factor Authentication on a Shared Account

A global team that shares the same Ops Manager account can use Google Authenticator and use the same seed code for all team members. To generate a common seed code that all team members can use, select the *Can't scan the barcode?* link when *Configuring Two-Factor Authentication with Google Authenticator*.

Procedures

To enable or disable two-factor authentication for the entire Ops Manager environment, see *Manage Two-Factor Authentication for Ops Manager*.

Configure Two-Factor Authentication with Text or Voice

Step 1: In Ops Manager, select the *Administration* tab and then *Account*.

Step 2: Select the pencil icon for *Two Factor Authentication*.

Or, if this is the first time you are setting up an account, click the *Configure* button to the right side of the *Account* page and follow the instructions.

Step 3: Select *Use Voice/SMS*.

Step 4: Enter the phone number for the phone that will receive the codes.

If you are outside of the United States or Canada, you must include `011` and your country code. Alternatively, you can sign up for a Google Voice number and use that number for your authentication.

Step 5: Select how to receive the codes.

Select either *Text message (SMS)* or *Voice call (US/Canada only)*.

Step 6: Click *Send Code*.

Ops Manager sends the codes to your phone.

Step 7: Enter the code in the box provided in Ops Manager and click *Verify*.

Step 8: Click *Save Changes*.

Configure Two-Factor Authentication with Google Authenticator

Step 1: Install Google Authenticator from either the Google Play store or the iOS Apple Store, depending on your device.

Step 2: Run Google Authenticator.

Step 3: Click `Begin Setup`.

Step 4: When prompted, select how you will enter the shared private key.

Under `Manually Add an Account`, select either `Scan a barcode` or `Enter provided key`. Stay on this screen while you use the next steps to access the barcode or key in Ops Manager.

Step 5: In Ops Manager, select the *Administration* tab and then *Account*.

Step 6: Select the pencil icon for *Two Factor Authentication*.

Or, if this is the first time you are setting up an account, click the *Configure* button to the right side of the *Account* page and follow the instructions.

Step 7: Select *Use Google Authenticator*.

Ops Manager provides a barcode and a *Can't scan the barcode?* link.

Step 8: Scan or enter the shared private key.

If your smartphone can scan barcodes, then scan the barcode. Otherwise, click *Can't scan the barcode?* and type the provided *Key* into your smartphone.

Step 9: Enter the Google Authenticator code in Ops Manager.

After you scan the barcode or enter the key, Google Authenticator generates a 6-digit code. Enter that in the box provided in Ops Manager and click *Verify*.

Step 10: Click *Save Changes*.

Generate New Recovery Codes

As a backup, you can generate recovery codes to use in place of a sent code when you do not have access to a phone or your Google Authenticator application. Each recovery code is single-use, and you should save these codes in a secure place. When you generate new recovery codes, you invalidate previously generated ones.

Step 1: In Ops Manager, select the *Administration* tab and then *Account*.

Step 2: Select the pencil icon for *Two Factor Authentication*.

Or, if this is the first time you are setting up an account, click the *Configure* button to the right side of the *Account* page and follow the instructions.

Step 3: Select *Generate New Recovery Codes*.

Keep the codes in a safe place. Each code can be used in conjunction with your username and password to not only access Ops Manager but to reset your security settings on Ops Manager.

8 Administration

Manage Your Account Administer your account and groups.

Administer the System Administer the Ops Manager system. This access is available only to administrators with global access.

Manage Groups Create and manage Ops Manager groups.

Manage Ops Manager Users and Roles Control access to Ops Manager groups and group resources by creating Ops Manager users and assigning roles.

Manage MongoDB Users and Roles Control access to your MongoDB deployments by enabling user authentication, creating MongoDB users, and assigning roles.

Configure Available MongoDB Versions Configure the versions that are available for your Automation Agents to download.

Backup Alerts Describes alert messages concerning the Ops Manager Backup system.

Start and Stop Ops Manager Application Manage the Ops Manager Application.

Back Up Ops Manager Options for backing up Ops Manager.

8.1 Manage Your Account

On this page

- *API Keys & Whitelists Page*
- *My Groups Page*
- *Group Settings Page*
- *Users Page*
- *Agents Page*
- *Billing/Subscriptions*
- *Payment History*

The *Administration* tab lets you personalize your console and activate or deactivate a variety of features. The *Administration* tab provides the pages described here.

Account Page

The *Account* page allows users to update their personal information.

- *User Name*: Displays the user's name. You cannot change your username.
- *Email Address*: Displays the email address Ops Manager associates with your account. You can change your email address by clicking on the "*pencil*" icon.
- *Mobile Phone Number*: The number to use to receive SMS alerts, including two-factor authentication codes.
- *Password*: Allows you to change your Ops Manager password. Passwords must fulfill Ops Manager's *password requirements*.
- *Two-Factor Authentication*: Ops Manager requires two factor authentication for login. For details, see *Manage Your Two-Factor Authentication Options*.

  To delete or reset two-factor authentication, contact your Ops Manager system administrator.

Personalization Page

The *Personalization* page allows users to configure the console to suit their needs and preferences. The available fields depend on the user's *role*.

- *My Time Zone*: Sets your local time zone.
- *My Date Format*: Allows you to select your preferred date format.
- *Page Shown When Switching Groups*: Sets which page of the Ops Manager console you will see when you select a different group. If you select *Current*, Ops Manager will not change pages when you select a different group.
- *Display Opcounters On Separate Charts*: Allows you to control the presentation of Opcounter Charts. If enabled, Ops Manager charts each opcounter type separately. Otherwise, each opcounter type is overlaid together in a single chart.
- *Display Chart Annotations*: Toggles the presence of chart annotations. Chart annotations overlay information about significant system events on the charts. For example, with chart annotations Ops Manager will draw a red vertical line over the charts.
- *Receive |mms| Newsletters*: Allows you to opt-in to, or opt-out of receiving e-mail newsletters about Ops Manager.

API Keys & Whitelists Page

API Keys & Whitelists lets you *generate an API key* for your Ops Manager group. Use the API key to support automated installation of your Monitoring Agent with scripts included with the agent installation files.

You can also define an *API Whitelist* of IP addresses permitted to invoke operations that require a higher level of security. See *Enable the Public API* for more information.

My Groups Page

My Groups displays the Ops Manager groups you belong to. From here you can *add a group*.

Group Settings Page

These settings apply to all users in the group. To set group settings, see *Manage Group Settings*.

Users Page

Users displays the group's users and their *roles*. From here you can *add and manage users*.

Agents Page

The *Agent* page displays the status of your installed agent and provides links to download new agents, in both `.zip` and `.tar.gz` formats. The software is dynamically assembled with your API key. Instructions are included to set up and start the downloaded agent, as well as create a new user for the agent if MongoDB authentication is used.

The *Agents* page includes the following information about your installed agents:

Field	Description
Status	The time of the last ping from the agent.
Type	The type of agent.
Hostname	The hostname for the agent and any warnings, such as that the agent is down or out-of-date.
State	Indicates whether the agent is active.
Ping Count	The number of pings (i.e. data payloads) sent by the agent since midnight GMT. Typically agents send pings every minute.
Version	The version of the agent software running on this agent instance.
Log	Click *view logs* to view the agent's log.

If you have more than one Monitoring Agent, only one agent actively monitors MongoDB instances at a time. See *Monitoring Architecture* for more information.

Billing/Subscriptions

For users with the *appropriate permissions*, this page displays the account's payment method and billing history.

Payment History

For users with the *appropriate permissions*, this page displays the account's payment history.

8.2 Administer the System

On this page

The *Administration* section of the Ops Manager application provides a tool to view system status, set cron jobs, manage users and groups, and perform other administrative functions.

If you have administrative privileges for the Ops Manager deployment, click the *Admin* link in the top right corner of Ops Manager to access the system administration tool.

This page describes the *Admin* section's tabs and pages.

General Tab

The *General* tab provides overview information, messages, reports, and access to users.

Overview Page

The *Overview* page provides reports on system use and activity.

This page displays the summary table and *Total Pages Viewed* and *Total Chart Requests* charts.

A summary table reports totals for:

- groups
- active groups
- active hosts
- users
- users active
- ping spillover queue
- increment spillover queue

Additionally two charts report:

- total page views
- total chart requests

Users Page

The *Users* page displays a list of user information for all people with permission to use the Ops Manager application as well as provides an interface to manage users. Use the search box on the upper right corner to find a user record.

Users Information

For each user, the users table reports:

- the username
- the available *roles*, if any
- the date and time of the last login.
- the date and time of the last access event.
- the total number of login.
- the user's configured timezone.
- the creation date and time.

Edit a User Record

To edit a user record:

1. Click the pencil icon at the far right of the user record.
2. In the *Edit User* interface, you can:
    - Edit the user's email.
    - Select the user's groups and *roles*.
    - Lock or unlock the account.
    - Specify the user's global roles.
3. When finished, click the *Save* button to save any changes and exit.

Groups Page

The *Groups* page lists all groups created with their date created and the last date and time an agent for the group pinged Ops Manager. At the top right of the page is a search box to find a group by name. Click a group name to view the group's details.

Logs Page

The *Logs* page in *General* tab lists backup logs by job and class with messages grouped and counted by last hour, 6 hours, and last 2 days. Click a count number to see all messages in the group.

Messages Page

You can display a short message on any page of the Ops Manager application to notify or remind users, such as impending maintenance windows. Messages may be *active*, i.e. visible, on all pages or a subset of pages.

The *Messages* page provides information on existing messages as well as an interface to add new messages or manage existing messages.

UI Messages Table

The *UI Messages* table holds all available messages. Use the search box on the top right to filter the list of messages.

For each message, the table reports:

- which page or page prefix the message will appear on.
- the text of the message.
- whether the message is active or inactive. Active messages are also highlighted in orange.
- the creation date and time for the message.
- the date and time of the last modification for the message.

Add a Message

To add a message which appears on one page, or on all pages,

1. Click the:guilabel:*Add Message* button.
2. In the *Add Message* interface, enter
   - the message, and
   - the page URL or optionally, page prefix in the *Add Message* interface.

     The page prefix allows you to specify a path of a single page or the URL prefix of a group of pages. The prefix must begin with a / character.
3. Click the *Active* checkbox to make the message live. Optionally, you can leave the box unchecked to disable the message.
4. Click the *Add* button to add the message.

Once added, active messages take 60 seconds before they display.

Disable a Message

To disable a message, click the orange square button to the right of any alert listed on the *Messages* page. The orange button will change to a grey button.

To re-enable a disabled message, click the grey button; the grey button will change to back to an orange button.

Delete a Message

To delete a message, click the garbage can icon to the right of any alert listed on the *Messages* page.

Audits Page

The *Audits* page displays all events tracked by Ops Manager. This includes the group events as well as internal and system events, which are not tracked at a group level.

Backup Tab

The *Backup* tab provides information on Backup resources, including jobs, daemons, and blockstores.

Jobs Page

You can see all active and stopped Backup jobs on the *Jobs* page. For each backup job, the tab lists job group, name, last agent conf, last oplog, head time, last snapshot, what the agent is working on, the daemon, and the blockstore.

Ops Manager puts a yellow background on the following fields if they are delayed:

- *Last Agent Conf*, if older than 1 hour.
- *Last Oplog*, if older than 1 hour before the *Last Agent Conf*.
- *Head Time*, if older than 1 hour before *Last Oplog*.
- *Last Snapshot*, if older than the snapshot interval multiplied by 1.5.

Click the group name to go directly to the backup job page for the group. Click the job name to see the results of the job on a detail page. On the job detail page, click links to see the logs, conf call, and download a zipped archive file containing complete diagnostic information for the specified job.

Job Timeline Page

The *Job Timeline* page displays a graphical representation of information found on other *Admin* pages, in particular the *Jobs* page. The *Job Timeline* page displays critical timestamps (head, last snapshot, next snapshot) and offers a way to assign a daemon to a given job.

Click the *Auto refresh* checkbox to update the list automatically every 10 seconds. Click the *Refresh* button to refresh data manually.

To view the backup job JSON, click the *Show JSON* link under the *Group* heading for any backup job. When the JSON displays, click the *View raw runtime data* link under the code to view the raw data. To hide the daemon JSON, click the *Hide JSON* link.

To move the job to a new Backup Daemon, click the *Move head* link under the *Machine* heading for a backup job. Select the location and click the *Move head* button to move the job to a new Backup Daemon. Ops Manager does not automatically move jobs between daemons.

You can bind a backup job to a head by clicking *Set binding* under the *Machine* heading for a backup job. You can bind a job to a preferred Backup Daemon during *initial sync*. After initial sync completes, Ops Manager automatically assigns jobs to Backup Daemons.

Logs Page

The *Logs* page on the backup tab lists backup logs by job and class with messages grouped and counted by last 2 hours, last day, and last 3 days. Click a count number to see all messages in the group.

Restores Page

The *Restores* page displays the last 100 requested restores and their progress. To show all restores ever requested, click *Show All*.

Resource Usage Page

The *Resource Usage* page provides key size and throughput statistics on a per-job basis for all groups for which Backup is enabled. The page displays such throughput statistics as the size of the data set, how active it is, and how much space is being used on the blockstore.

To export the information, click *EXPORT AS CSV*.

Grooms Page

New in version 1.4.

Ops Manager performs periodic garbage collection on blockstores through groom jobs that remove unused blocks to reclaim space. Unused blocks are those that are no longer referenced by a live snapshot. An Ops Manager scheduling process determines when grooms are necessary.

This page lists active and recent groom jobs.

A groom job forces the backup process to:

- write all new data to a new location,
- copy all existing, needed data from the old location to the new one,
- update references, to maintain data relationships, and
- drop the old database.

During groom operations, you may notice that blockstore file size will fluctuate, sometimes dramatically.

You can manually direct Ops Manager to move blocks between Blockstores through the *Groom Priority*.

Groom Priority

New in version 1.6.

The *Groom Priority* page allows you to manually schedule jobs to move a backup's blocks to a different blockstore and to manually schedule *grooms*. The page lists each backup by its replica set and highlights a backup in blue if a groom is in progress.

To move a backup's chunks to a different blockstore, select the destination blockstore in the backup's *Blockstore List* column. You might want to do this, for example, if you add a new blockstore and would like to balance data.

Typically, you should not need to manually schedule groom jobs. Ops Manager *runs the jobs automatically*. If you do need to initiate a job, click the *Schedule* button for the backup's replica set.

Daemons Page

The *Daemons* page lists all active Backup Daemons and provides configuration options. Ops Manager automatically detects Backup Daemons and displays them here. You can reconfigure daemons from this page. Changes can take up to 5 minutes to take effect.

Use the *Pre-Configure New Daemon* button at the bottom of the page if you want to add a daemon but do not want it to take new jobs. Type the `<machine>:<roothead path>` in the text field above the *Pre-Configure New Daemon* button. Click the button to configure the new Backup Daemon.

For each daemon, the page lists the server name, configuration, head space used, head space free, the number of replica sets backed up, the percentage of time the Backup Daemon Service was busy, and job runtime counts by 1 minute, 10 minutes, 60 minutes, less than or equal to 180 minutes, and greater than 180 minutes.

The page includes the following fields and links to manage and configure daemons:

Field	Description
Show Detailed JSON	Displays the Backup Daemon JSON. When the JSON displays, the *View raw runtime data* link appears under the code, allowing you to view the raw data.
Move all heads	Moves all the Backup Daemon jobs that live on this daemon to a new daemon location. Click the link, then select the location, and then click the *Move all heads* button.
Delete Daemon	Deletes a Backup Daemon.
Assignment	Allows the daemon to take more jobs. If a disk fills on daemon server, you can deselect this so the server isn't overloaded.
Backup Jobs	Allows the daemon to take more backup jobs. Deselect this, for example, when performing maintenance on a daemon's server.
Restore Jobs	Allows the daemon to take more restore jobs.
Resource Usage	Collects information on Blockstore use (i.e., on how many blocks are still referenced by snapshots). Ops Manager uses this information to generate the *Resource Usage Page* page and to prioritize garbage collection.
Garbage Collection	Enables garbage collection.
Head Disk Type	Indicates whether the disk type is SSD or HDD. If you change a daemon to SSD, then only jobs with an oplog greater than 1GB/hour will go to this daemon. Any jobs with an oplog less than 1GB/hour can only go to a daemon marked HDD.

Blockstores Page

The *Blockstores* page lists all Backup blockstores and provides the ability to add a new blockstore.

Edit an Existing Blockstore

To update an existing Blockstore, edit the *<hostname>:<port>* field, *MongoDD Auth Username* field, and *MongoDB Auth Password* field. If the username and password had used the `credentialstool` for encryption, click the *Encrypted Credentials* checkbox.

Add a Blockstore

To add a blockstore, first *deploy the dedicated MongoDB instance(s)* to host the blockstore.

Then, on the *Blockstores* page, enter the `<id>` and `<hostname:port>` in the text fields above the *Add New Blockstore* button. For replica sets, enter a comma-separated list for `<hostname:port>`. Finally, click the button to save and add the blockstore. A newly configured blockstore becomes active when you select the assignment checkbox.

See also:

Configure Multiple Blockstores in Multiple Data Centers describes how to configure multiple blockstores in different data centers.

Delete a Blockstore

To delete a blockstore, click the *Delete blockstore* link for the blockstore.

Sync Stores Page

This page configures the backing replica sets that store sync slices. Sync slices are temporary backups of your deployment batched into 10MB slices. The Backup Agent creates sync slices during an *initial sync*, and the Backup Daemon uses them to create the permanent backup. Ops Manager deletes sync slices once the initial sync completes.

Do **not** modify a sync store's connection string to point to a different replica set. Doing so requires all Ops Manager components to be restarted. Existing data will not be copied automatically.

From this page you can:

- Add a member to a replica set by adding the host to the replica set connection string in the *<hostname>:<port>* field. Do **not** modify the connection string to point to a different replica set.
- Change the authentication credentials or connection options for a sync store by modifying the appropriate fields.
- Add an additional sync store by clicking the *Add New Sync Store* button at the bottom of the page.
- Enable and disable new assignments to sync stores using the checkboxes in the *Assignment Enabled* column.

Oplog Stores Page

This page configures the backing replica sets that store oplog slices. Oplog slices are compressed batches of the entries in the tailed oplogs of your backed-up deployments.

Do **not** modify the oplog store's connection string to point to a different replica set. Doing so requires all Ops Manager components to be restarted. Existing data will not be copied automatically.

From this page you can:

- Add a member to a replica set by adding the host to the replica set connection string in the *<hostname>:<port>* field. Do **not** modify the connection string to point to a different replica set.
- Change the authentication credentials or connection options for a oplog store by modifying the appropriate fields.
- Add an additional oplog store by clicking the *Add New Oplog Store* button at the bottom of the page.
- Enable and disable new assignments to oplog stores using the checkboxes in the *Assignment Enabled* column.

Control Panel Tab

The *Control Panel* tab provides access to Cron jobs and email history.

Email History Page

The *Email History* page displays all alerts emails sent, the subject, and the recipient. To view an email, click the subject link. To filter the list, click *FILTER BY RECIPIENT*.

Cron Jobs Page

The *Cron Jobs* page displays all scheduled processes and their statuses. To disable a job, uncheck its box in the *ENABLED* column. To run a job, click *RUN NOW*.

8.3 Manage Groups

On this page

Overview

Groups associate Ops Manager users with MongoDB in the Ops Manager application. A group provides access to a distinct Ops Manager environment or deployment. Each group has one Monitoring Agent and Backup Agent.

You create the first Ops Manager group when you register the first Ops Manager user. Create additional groups to manage segregated systems or environments. For example, your deployment might have two or more environments separated by firewalls. In this case, you would need two or more separate Ops Manager groups.

Ops Manager API and shared secret keys are unique to each group. Each group requires its own Monitoring Agent with the appropriate API and shared secret keys. Within each group, the agent needs to be able to connect to all hosts it monitors in the group.

The user who creates a group automatically has the *Owner* role and can manage user access.

Working with Multiple Environments

If you have multiple MongoDB systems in distinct environments and cannot monitor all systems with a single agent, you will need to add a new group. Having a second group makes it possible to run two agents.

You may also use a second group and agent to monitor a different set of MongoDB instances in the same environment if you want to segregate the hosts within the Ops Manager console. A user can only view data from the hosts monitored in a single group at once.

After adding a second group, the Ops Manager interface will have a drop-down list that will allow you to change groups. Selecting a new group will refresh the current page with the data available from the servers in this group.

Procedures

Create a Group

When you create a new group, you are automatically added as the first user to the group. The group is automatically assigned a set of *alert configurations*.

Step 1: Select the *Administration* tab and then the *My Groups* page.

Step 2: Click the *ADD GROUP* button.

Step 3: Type a name for the new group and click *CONTINUE*.

For security and auditing reasons, you cannot use a name used earlier. Once you name a group, the group's name cannot be changed.

Step 4: Assign Hosts.

After you create the group, Ops Manager will log you into the group and displays a *Welcome* page. To create the group's first deployment, click either *BEGIN SETUP* to use the Ops Manager set-up wizard or click *Advanced Setup* to add a deployment manually.

Remove a Group

Please contact your Ops Manager administrator to remove a company or group from your Ops Manager account.

Manage Group Settings

Step 1: Select the *Administration* tab and then select *Group Settings*.

Step 2: Modify group settings as desired.

See *Group Settings* options for a full description of the group settings.

If you have *Global Owner* access, the interface displays a *second Group Settings* link under the *Admin Only* section. For information on these settings, see *Admin Only Group Settings*.

Group Settings

The following settings in the *Administration* tab's *Group Settings* page apply to all users in the group:

- *Group Time Zone*: Sets your group's time zone.
- *Collect Logs For All Hosts*: Activates or deactivates the collection of log data for all hosts. This overwrites the statuses set on the individual hosts.
- *Collect Profiling Information for All Hosts*: Activates or deactivates the collection of profiling information for all hosts. Ops Manager Monitoring can collect data from MongoDB's profiler to provide statistics about performance and database operations. Ensure exposing profile data to Ops Manager Monitoring is consistent with your information security practices. Also be aware the profiler can consume resources which may adversely affect MongoDB performance.
- *Collect Database Specific Statistics*: Allows you to enable or disable the collection of database statistics. For more information, see "*How does \mms\ gather database statistics?*".
- *Enable Public API*: Allows you to *use the Public API* with the group.
- *Reset Duplicates*: Allows you to reset and remove all detected duplicate hosts. This is useful if your server environment has drastically changed and you believe a host is incorrectly marked as a duplicate.
- *Preferred Hostnames*: Allows you to specify the hostname to use for servers with multiple aliases. This prevents servers from appearing multiple times under different names. By default, the Monitoring Agent tries to connect by resolving hostnames. If the agent cannot connect by resolving a hostname, you can force the Monitoring Agent to prefer an IP address over its corresponding hostname for a specific IP address. To override this default behavior, set an IP address as a preferred hostname. If your IP addresses have a common prefix, create a preferred hostname with the *ends-with* button or click the *regexp* button to use a regular expression.
- *Public Key for SCP Restores*: If you use Ops Manager *Backup*, this setting allows you to generate a public key for SCP backup restoration. If you restore a snapshot through SCP, Ops Manager uses the key to transmit the snapshot. For more information on restores, see *Restore MongoDB Instances with Backup*, and *how to validate an SCP restore* and other SCP FAQs.

- *PagerDuty Service Key*: Adds a service key for a PagerDuty account. This is the default key used if you *create an alert configuration* that uses PageDuty.
- *HipChat Settings*: Adds a room and API token for a HipChat account. These are the default settings used if you *create an alert configuration* that uses HipChat.

Admin Only Group Settings

The following group settings in the *Admin Only* section of the *Administration* tab could, in certain situations, affect more than the group. For example, setting logging to a high verbosity would cause system logs to roll over faster. Only users with the *Global Owner* role can edit these settings:

- *Mongos Deactivation Threshold*: Change the amount of time before Ops Manager stops monitoring an unreachable `mongos`. By default, the Monitoring Agent stops monitoring an unreachable `mongos` after 24 hours. Set this to the amount of time in hours to wait before deactivation.
- *Monitoring Agent Log Level*: Change the verbosity of the Monitoring Agent log.
- *Automation Agent Log Level*: Change the verbosity of the Automation Agent log.

8.4 Manage Ops Manager Users and Roles

Ops Manager requires users to log in to verify who they are and to determine their access to features. Ops Manager provides built-in user management for controlling authentication and access.

You can alternatively use LDAP to control authentication and access. If you do, you must start with a **fresh installation** of Ops Manager. See *Configure Users and Groups with LDAP for Ops Manager*.

The section describes the following:

Manage Ops Manager Users Create and manage Ops Manager users and assign roles.

Ops Manager Roles Describes the user roles available within the Ops Manager.

Manage Ops Manager Users

On this page

Overview

Ops Manager users provide access to Ops Manager groups. You can create a new user in a group to give access to that group. You can later give the user access to additional groups. When you create a user, you assign the user a role in the group. A role determines the actions the user can perform and the data the user can access.

Procedures

Add Users

Step 1: Click the *Administration* tab and then select the *Users* page.

Step 2: Click the *Add User* button.

Step 3: Enter the new user's email address and role.

Step 4: Click *Add/Invite*.

Step 5: If prompted, enter the two-factor verification code.

There might be a delay of a few seconds before you receive the prompt. Ops Manager will prompt you for a two-factor verification code if you have not verified recently.

Step 6: Click the *Send Email Invitation* button.

Users can create accounts using the account registration page of your Ops Manager installation.

View Ops Manager Users

To view users, click the *Administration* tab and then *Users* page. The *Users* page lists users who have access to your Ops Manager group, their roles, their time zones, and other information. The page also lists any invitations to join the group waiting for a reply, as well as any requests from users who want to join the group. A user can request to join a group when first registering for Ops Manager.

View Invitations

When you invite a user to join a group, Ops Manager then sends an email to the prospective new user. To view invitations sent but not yet accepted, click the *Administration* tab and then select the *Users* page. The page lists any users with pending invitations. To cancel an invitation, click *CANCEL INVITE*.

View Requests

Users can request access when they create their Ops Manager account, as on the registration page.

To view requests, click the *Administration* tab and then select the *Users* page. The *Users* page lists any pending requests to join your group. To approve or deny a request, click the appropriate button.

Remove Ops Manager Users

Step 1: Click the *Administration* tab and then select the *Users* page.

Step 2: Remove the user.

Click the user's gear icon and select *Delete User*.

Assign Roles to Ops Manager Users

Assign *roles* to Ops Manager users to limit the actions they can perform and the data they can view.

To assign roles to users in a *group*, you must have either the *User Admin* role or *Owner* role in the group. The user who creates a group automatically has the *Owner* role.

To assign roles to any user in any group, you must have either the *Global User Admin* role or *Global Owner* role.

You can assign roles either through Ops Manager, as described here, or *through an LDAP server* after you have *set up LDAP integration* and created LDAP groups for your Ops Manager roles.

To assign roles inside of Ops Manager, select the *Administration* tab and then the *Users* page. Click the user's gear icon and select *Edit User*. Click the appropriate checkboxes to assign *roles*.

Assign Ops Manager Roles with LDAP

To assign roles through an LDAP server, you must first *set up LDAP integration* and create LDAP groups for your Ops Manager roles. You must also have the permissions described in *Assign Roles to Ops Manager Users*.

For LDAP authentication, the welcome form includes the ability to assign LDAP groups to the Ops Manager group-level and global roles.

There is no planned upgrade path from existing Ops Manager user authentication to using LDAP. You will need to recreate users, groups, and roles manually with your LDAP service, as described in the *Configure Users and Groups with LDAP for Ops Manager* document.

1. *Configure LDAP authentication*.
2. Create groups on your LDAP server for each of the available Ops Manager group-level and global roles.

To assign LDAP groups to Ops Manager roles:

1. Click the *Admin* link at the top right of the Ops Manager page.
2. Click *General* and then click *Groups*.
3. Click the pencil icon at the far right of a group name. Edit the Roles interface by adding the appropriate LDAP group name to its corresponding Ops Manager group name.

Because Ops Manager does not update role assignments stored in your LDAP server, assign roles by assigning users to groups in your LDAP server.

Configure global roles in the `conf-mms.properties` *configuration file*.

Ops Manager Roles

On this page

Overview

Ops Manager roles allow you to grant users different levels of access to Ops Manager. You can grant a user the privileges needed to perform a specific set of tasks and no more.

If you use LDAP authentication for Ops Manager, you must create LDAP groups for each available role described below then assign users to LDAP groups. There is no round trip synchronization between your LDAP server and Ops Manager.

To assign user roles, see *Assign Roles to Ops Manager Users*. You cannot assign your own roles.

Read Only

The **Read Only** role has the lowest level of privileges. The user can generally see everything in a group, including all activity, operational data, users, and user roles. The user, however, cannot modify or delete anything.

User Admin

The **User Admin** role grants access to do the following:

- Add an existing user to a group.
- Invite a new user to a group.
- Remove an existing group invitation.
- Remove a user's request to join a group, which denies the user access to the group.
- Remove a user from a group.
- Modify a user's roles within a group.
- Update the billing email address.

Monitoring Admin

The **Monitoring Admin** role grants all the privileges of the **Read Only** role and grants additional access to do the following:

- Manage alerts (create, modify, delete, enable/disable, acknowledge/unacknowledge).
- Manage hosts (add, edit, delete, enable deactivated).
- Manage group-wide settings.
- Download Monitoring Agent.

Backup Admin

The **Backup Admin** role grants all the privileges of the **Read Only** role and grants access to manage *backups*, including the following:

- Start, stop, and terminate backups.
- Request restores.
- View and edit excluded namespaces.
- View and edit host passwords.
- Modify backup settings.
- Generate SSH keys.
- Download the Backup Agent.

Automation Admin

The **Automation Admin** role grants all the privileges of the **Read Only** role and grants access to the following management actions:

- View deployments.
- Provision machines.
- Edit configuration files.
- Modify settings.
- Download the Automation Agent.

Owner

The **Owner** role has the privileges of all the other roles combined, as well as additional privileges available only to Owner. In addition to the privileges of other roles, an Owner can:

- Set up the *Backup* service.
- Update billing information.
- Enable the Public API.

Group Roles

The following roles grant privileges within a group:

Global Roles

Global roles have all the same privileges as the equivalent Group roles, except that they have these privileges for all groups. They also have some additional privileges as noted below.

Global Read Only

The **Global Read Only** role grants *read only* access to all groups. The role additionally grants access to do the following:

- View *backups* and other statistics through the *admin* UI.
- Global user search.

Global User Admin

The **Global User Admin** role grants *user admin* access to all groups. The role additionally grants access to do the following:

- Add new groups.
- Manage UI messages.
- Send test emails, SMS messages, and voice calls.
- Edit user accounts.
- Manage LDAP group mappings.

Global Monitoring Admin

The **Global Monitoring Admin** role grants *monitoring admin* access to all groups. The role additionally grants access to do the following:

- View system statistics through the *admin* UI.

Global Backup Admin

The **Global Backup Admin** role grants *backup admin* access to all groups. The role additionally grants access to do the following:

- View system statistics through the *admin* UI.
- Manage blockstore, daemon, and oplog store configurations.
- Move jobs between daemons.
- Approve backups in awaiting provisioning state.

Global Automation Admin

The **Global Automation Admin** role grants *automation admin* access to all groups. The role additionally grants access to view system statistics through the *admin* UI.

Global Owner

The **Global Owner** role for an Ops Manager account has the privileges of all the other roles combined.

8.5 Manage MongoDB Users and Roles

You can enable MongoDB access control and manage MongoDB users and roles directly from the Ops Manager interface.

Enable MongoDB Role-Based Access Control Control access to MongoDB databases.

Manage MongoDB Users and Roles Add MongoDB users and assign roles.

Manage Custom Roles Create custom roles.

Enable MongoDB Role-Based Access Control

On this page

Overview

MongoDB uses Role-Based Access Control (RBAC) to determine access to a MongoDB system. When run with access control, MongoDB requires users to authenticate themselves to determine their access. MongoDB limits each user to the resources and actions allowed by the user's roles. If you leave access control disabled, any client can access any database in your deployments and perform any action.

When you enable MongoDB access control, you enable it for all the deployments in your Ops Manager group. The group shares one set of users for all deployments, but each user has permissions only for specific resources.

Access control applies to the Ops Manager agents as well as to clients. When you enable access control, Ops Manager creates the appropriate users for the agents.

Considerations

Once you enable access control, you must *create MongoDB users* so that clients can access your databases. Always use the Ops Manager interface to manage users and roles. Do not do so through direct connection to a MongoDB instance.

When you enable access control, Ops Manager creates a user with global privileges used only by the Automation Agent. The first user you create can be any type of user, as the Automation Agent guarantees you will always have access to user management.

For more information on MongoDB access control, see the Authentication and Authorization pages in the MongoDB manual.

Enable MongoDB Access Control

Step 1: Select the *Deployment* tab and then select *Authentication & Users*.

Step 2: Turn on authentication.

Click the button for *Auth Enabled* to toggle it to *Yes*. This automatically creates users for your Monitoring, Backup, and Automation agents.

Step 3: Click *Review & Deploy*.

Step 4: Review your changes, and click *Confirm & Deploy*.

Next Steps

To create your first users and assign privileges, see *Manage MongoDB Users and Roles*.

Manage MongoDB Users and Roles

On this page

Overview

When *MongoDB access control is enabled*, you provide client access to MongoDB by creating users and assigning user roles. The users you create apply to all MongoDB instances in your Ops Manager group, but each user has a specified authentication database. Together, the user's name and database serve as a unique identifier for that user.

You can specify access using MongoDB's built-in roles and also by *creating custom roles*. Ops Manager provides the interface for doing so.

You can create users before enabling accessing control or after, but they don't go into effect until you enable access control. The MongoDB instances won't require user credentials until access control is enabled.

To authenticate, a client must specify the username, password, database, and authentication mechanism. For example, from the `mongo` shell, a client would specify the `--username`, `--password`, `--authenticationDatabase`, and `--authenticationMechanism` options.

MongoDB users are separate from Ops Manager *users*. MongoDB users have access to MongoDB databases. Ops Manager users access to Ops Manager groups.

Considerations

If you want Ops Manager to ensure that all deployments in a group have the same database users and roles, use only the Ops Manager interface to manage users.

However, if you want certain deployments in a group to have certain users or roles not set at the group level, you can add them through direct connection to the MongoDB instances."

Add a MongoDB User

Step 1: Select the *Deployment* tab and then select *Authentication & Users*.

Step 2: Select the *Add User* button.

Step 3: In the *Identifier* fields, enter the database on which the user authenticates and enter a username.

Together, the database and username uniquely identify the user. Though the user has just one authentication database, the user can have privileges on other database. You grant privileges when assigning the user roles.

Step 4: In the *Roles* drop-down list, select the user's roles.

You can assign both user-defined roles and built-in roles.

Step 5: Enter the user's password and click *Add User*.

Step 6: Click *Review & Deploy*.

Step 7: Review your changes, and click *Confirm & Deploy*.

Edit a User's Roles

Step 1: Select the *Deployment* tab and then select *Authentication & Users*.

Step 2: Click the user's gear icon and select *Edit*.

Step 3: Edit the user's information.

In the *Roles* drop-down list, you can both add and delete roles. You can add both user-defined roles and built-in roles.

Step 4: Click *SAVE CHANGES*.

Step 5: Click *Review & Deploy*.

Step 6: Review your changes, and click *Confirm & Deploy*.

Remove a MongoDB User

Step 1: Select the *Deployment* tab and then select *Authentication & Users*.

Step 2: Click the user's gear icon and select *Remove*.

Step 3: To confirm, click *Delete User*.

Step 4: Click *Review & Deploy*.

Step 5: Review your changes, and click *Confirm & Deploy*.

Manage Custom Roles

On this page

Overview

Roles grant users access to MongoDB resources. By default, MongoDB provides a number of built-in roles, but if these roles cannot describe a desired privilege set, you can create custom roles.

When you create a role, you specify the database to which it applies. Ops Manager stores your custom roles on all MongoDB instances in your Ops Manager group but uniquely identifies a role by the combination of the database name and role name. If a database with that name exists on multiple deployments within your Ops Manager group, the role applies to each of those databases. If you create a role on the `admin` database, the role applies to all `admin` databases in the deployment.

Roles consist of privileges that grant access to specific actions on specific resources. On most databases, a resource is the database or a collection, but on the `admin` database a resource can be all databases, all collections with a given name across databases, or all deployments.

A role can inherit privileges from other roles in its database. A role on the `admin` database can inherit privileges from roles in other databases.

MongoDB roles are separate from Ops Manager *roles*.

Prerequisite

MongoDB access control *must be enabled* to apply roles. You can create roles before enabling accessing control or after, but they don't go into effect until you enable access control.

Considerations

Use only the Ops Manager interface to manage users and roles. Do not do so through direct connection to a MongoDB instance.

Procedures

Create a Custom MongoDB Role

Step 1: Select the *Deployment* tab and then select *Authorization & Roles*.

Step 2: Select the *ADD ROLE* button.

Step 3: In the *Identifier* fields, enter the database on which to define the role and enter a name for the role.

A role applies to the database on which it is defined and can grant access down to the collection level. The role's database and name uniquely identify the role.

Step 4: Select the role's privileges.

You can add privileges in two ways:

Give a role the privileges of another role.

To give a role all the privileges of one or more existing roles, select the roles in the *Inherits From* field. The field provides a drop-down list that includes both MongoDB built-in roles and any custom roles you have already created.

Add a privilege directly.

To add specific privileges to the role, click *ADD PRIVILEGES FOR A RESOURCE*.

In the *Resource* field, specify the resource to which to apply the role. To specify the whole database, leave the field blank. To specify a collection, enter the collection name. If the resource is on the `admin` database, you can click *ADMIN* and apply the role outside the `admin` database.

In the *Available Privileges* section, select the actions to apply. For a description of each action, see Privilege Actions in the MongoDB manual.

Step 5: Click *ADD PRIVILEGES*.

Step 6: Click *ADD ROLE*.

Step 7: Click *Review & Deploy*.

Step 8: Review your changes, and click *Confirm & Deploy*.

Edit a Custom Role

You can change a custom role's privileges. You cannot change its name or database.

Step 1: Select the *Deployment* tab and then select *Authorization & Roles*.

Step 2: Click the role's gear icon and select *Edit*.

Step 3: Remove privileges.

To remove an inherited role, click the *x* next to the role.

To remove a privilege, click the trash icon next to the privilege.

Step 4: Add privileges.

You can add privileges to the role in two ways:

Give a role the privileges of another role.

To give a role all the privileges of one or more existing roles, select the roles in the *Inherits From* field. The field provides a drop-down list that includes both MongoDB built-in roles and any custom roles you have already created.

Add a privilege directly.

To add specific privileges to the role, click *ADD PRIVILEGES FOR A RESOURCE*.

In the *Resource* field, specify the resource to which to apply the role. To specify the whole database, leave the field blank. To specify a collection, enter the collection name. If the resource is on the `admin` database, you can click *ADMIN* and apply the role outside the `admin` database.

In the *Available Privileges* section, select the actions to apply. For a description of each action, see Privilege Actions in the MongoDB manual.

Step 5: Click *ADD PRIVILEGES*.

Step 6: Click *SAVE CHANGES*.

Step 7: Click *Review & Deploy*.

Step 8: Review your changes, and click *Confirm & Deploy*.

View Privileges for a Role

To view a role's privileges, select the *Deployment* tab, then the *Roles* page, and then select *view privileges* next to the role.

Each privilege pairs a resource with a set of Privilege Actions. All roles are assigned a database. Each built-in role is assigned to either `admin` database or every database.

Remove a Custom Role

Step 1: Select the *Deployment* tab and then select *Authorization & Roles*.

Step 2: Click the role's gear icon and select *Remove*.

Step 3: To confirm, click *DELETE ROLE*.

Step 4: Click *Review & Deploy*.

Step 5: Review your changes, and click *Confirm & Deploy*.

8.6 Configure Available MongoDB Versions

On this page

Overview

You can manage the versions that are available to your Automation Agents to download. MongoDB versions are downloaded by the Automation Agents "lazily." Automation Agents will not download selected versions unless the version is in use by a MongoDB process on your server.

Procedure

Step 1: Click the *Deployment* tab and then *Version Manager*.

Step 2: Select the checkboxes for the versions of MongoDB the Automation Agent will use.

Step 3: Click *Review & Deploy* at the top of the page.

Step 4: Click *Confirm & Deploy*.

8.7 Backup Alerts

On this page

If a problem with the Ops Manager Backup system occurs, Ops Manager sends an alert to system administrators. This page describes possible alerts and provides steps to resolve them.

Backup Agent Down

This alert is triggered if a Backup Agent for a group with at least one active replica set or cluster is down for more than 1 hour.

To resolve this alert:

1. Open the group in Ops Manager by typing the group's name in the *GROUP* box.
2. Select the *Backup* tab and then the *Backup Agents* page to see what server the Backup Agent is hosted on.
3. Check the Backup Agent log file on that server.

Backups Broken

If Ops Manager Backup detects an inconsistency, the Backup state for the replica set is marked as "broken."

To debug the inconsistency:

1. Check the corresponding Backup Agent log. If you see a "Failed Common Points" test, one of the following may have happened.
   - A significant rollback event occurred on the Backup replica set.
   - The oplog for the Backup replica set was resized or deleted.
   - High oplog churn caused the agent to lose the tail of the oplog.

   In such cases, you must resync the Backup replica set, as described in the procedure *Resync Backup*.

2. Check the corresponding job log for an error message explaining the problem. In Ops Manager, click *Admin*, then *Backup*, and then *Jobs*. Then click the name of the job and then *Logs*. Contact MongoDB Support if you need help interpreting the error message.

Cluster Snapshot Failed

This alert is generated if Ops Manager Backup cannot successfully take a snapshot for a sharded cluster backup. The alert text should contain the reason for the problem. Common problems include the following:

- There was no reachable mongos. To resolve this issue, ensure that there is at least one `mongos` showing on the Ops Manager *Deployment* page.
- The balancer could not be stopped. To resolve this issue, check the log files for the first config server to determine why the balancer will not stop.
- Could not insert a token in one or more shards. To resolve this issue, ensure connectivity between the Backup Agent and all shards.

Bind Failure

This alert is generated if a new replica set cannot be bound to a Backup Daemon. The alert test should contain a reason for the problem. Common problems include:

- No primary is found. At the time the binding occurred, no primary could be detected by the Monitoring Agent. Ensure that the replica set is healthy.
- Not enough space is available on any Backup Daemon.

In both cases, resolve the issue and then re-initiate the initial sync. Alternatively, the job can be manually bound through the Ops Manager Admin interface. In Ops Manager, click *Admin*, then *Backup*, and then *Job Timeline*.

For information on initial sync, see Replica Set Data Synchronization.

Snapshot Behind Snitch

This alert is triggered if the latest snapshot for a replica set is significantly behind schedule. Check the job log in the Ops Manager Admin interface for any obvious errors. In Ops Manager, click *Admin*, then *Backup*, and then *Jobs*. Then click the name of the job and then *Logs*.

8.8 Start and Stop Ops Manager Application

On this page

Start the Ops Manager Server

Note: If you installed from a `tar.gz` or `zip` archive, you must create a symlink located at the path `/etc/init.d/mongodb-mms` that points to the `<install_dir>/bin/mongodb-mms`.

After configuring your Monitoring deployment, you can start the Ops Manager server with this command:

```
sudo /etc/init.d/mongodb-mms start
```

In some situations, starting MongoDB *may* take several minutes to pre-allocate the journal files. This is normal behavior.

You can now use the Monitoring instance by visiting the URL specified in the `mms.centralUrl` parameter (e.g. http://mms.example.com:8080) to continue configuration:

Unlike the SaaS version of Ops Manager, Monitoring stores user accounts in the local MongoDB instance. When you sign into the Monitoring instance for the first time, the system will prompt you to register and create a new "group" for your deployment.

Because there are no Monitoring Agents attached to your account, the first page you see in Monitoring will provide instructions for downloading the Monitoring Agent. Click the "download agent" link to download a pre-configured agent for your account. Continue reading this document for installation and configuration instructions for the Ops Manager agent.

Stop the Ops Manager Server

Enter the following command:

```
sudo /etc/init.d/mongodb-mms stop
```

Startup Log File Output

The Ops Manager server logs its output to a `logs` directory inside the installation directory. You can view this log information with the following command:

```
sudo less <install_dir>/logs/mms0.log
```

If the server starts successfully, you will see content in this file that resembles the following:

```
[main] INFO  ServerMain:202 - Starting mms...
[main] WARN  AbstractConnector:294 - Acceptors should be <=2*availableProcessors:␣
↪SelectChannelConnector@0.0.0.0:0
[null] LoginService=HashLoginService identityService=org.eclipse.jetty.security.
↪DefaultIdentityService@1eb3319f
[main] INFO  AppConfig:46 - Starting app for env: hosted
[main] INFO  MmsAppConfig:67 - Not loading backup components
[main] INFO  GraphiteSvcImpl:67 - Graphite service not configured, events will be␣
↪ignored.
[main] INFO  TwilioSvcImpl:48 - Twilio service not configured, SMS events will be␣
↪ignored.
[main] INFO  OpenDMKSnmpTrapAgentSvcImpl:91 - SNMP heartbeats hosts not configured,␣
↪no heartbeat traps will be sent.
[main] INFO  ServerMain:266 - Started |mms| in: 24979 (ms)
```

Optional: Run as Different User

1. Edit `<install_dir>/conf/mms.conf`:

```
|mms|_USER=foo_user
```

2. Change Ownership of `<install_dir>` for new user:

```
sudo chown -R foo_user:foo_group <install_dir>
```

3. Restart Ops Manager server:

```
sudo <install_dir>/bin/mongodb-mms restart
```

Optional: Ops Manager Application Server Port Number

1. Edit `<install_dir>/conf/conf-mms.properties`:

```
mms.centralUrl=http://mms.acmewidgets.com:<newport>
```

2. Edit `<install_dir>/conf/mms.conf`

```
BASE_PORT=<newport>
```

3. Restart Ops Manager server:

```
sudo <install_dir>/bin/mongodb-mms restart
```

8.9 Back Up Ops Manager

On this page

Deploying the Ops Manager Backup Blockstore and Application databases as replica sets is key to protect the databases from failure. Ensure you configure and deploy your replica sets for failover and redundancy. See Replica Set High Availability and Replica Set Deployment Architectures for more about replica set architecture.

Beyond using MongoDB's replication capabilities, you can create backups for the Backup Blockstore database and Application database, both for longterm storage of snapshots, and for backing up Ops Manager for disaster recovery purposes.

To restore Ops Manager, you only need backups of the Application database and the Backup Blockstore database. Ops Manager's other components are stateless: you can rebuild them from the installation packages if need be.

Important: Your Backup installation cannot back up Ops Manager. If you wish to use Ops Manager to back up Ops Manager, you will need *two* Ops Manager installations: one to back up your MongoDB deployment, and another to back up your Ops Manager databases.

Back Up with the Public API

The Ops Manager Public API provides the ability to programmatically restore snapshots on your desired schedule, and store them offline. Ideally, you should save the backups to a tape drive, appending the new snapshots daily until the drive is full, and then store the drive offsite.

Programmatically restoring snapshots has the same impact on the Backup Blockstore database as a typical restore.

You can then *restore your MongoDB instance from the stored snapshots*.

This method backs up the snapshots only: you **cannot** use the backups to restore Ops Manager in the event that the Blockstore is lost.

Shut Down and Back Up

You can perform a full backup of all snapshots contained in Ops Manager, as well as the point-in-time restores that Backup stores by shutting down Ops Manager and all of the hosts monitored and backed up by Ops Manager, and then backing up their file systems while they're offline.

The 'Shut Down and Back Up' approach has the advantage of providing all of Ops Manager's backup snap shots **as well as** the point-in-time restores that it stores. However, the process involves significant downtime and during the shut down, Ops Manager is completely unavailable. As such, if a failure occurs in one of your MongoDB instances, you may lose the data that was not backed up by Ops Manager prior to the shutdown.

Online Backup

Online backup is the riskiest option, as it involves backing up the databases while Ops Manager is running. If done correctly, this provides you with all of the backup snapshots contained in Ops Manager, as well as the point-in-time restores for the period leading up to the backup.

However, the process is complex and there is high potential for errors. To perform the online backup for databases using the **MMAPv1** storage engine, you must run `fsyncLock` on each `mongod` process that is part of the Application and Backup Blockstore replica sets in the correct order, back up the underlying file system, and then call `fsyncUnlock`. If you forget to call `fsyncUnlock` on a `mongod` instance, you risk significant long-term problems and failures.

Note: The `fsyncLock` command is not appropriate for databases that use the **WiredTiger** storage engine. See db.fsyncLock() in the MongoDB manual.

---

9 API

Public API Principles Overview of the Public API.

Public API Resources The resources exposed by the Public API.

Public API Tutorials Enable the API, and create and modify a deployment.

9.1 Public API Principles

On this page

Overview

The Ops Manager Public API follows the principles of the REST architectural style to expose a number of internal resources which enable programmatic access to Ops Manager's features.

The API has the following features:

- **JSON entities** - All entities are expressed in JSON.
- **Digest authentication** - To ensure that your API key is never sent over the network, API requests are authenticated using HTTP Digest Authentication.
- **Browsable interface** - Using a consistent linking mechanism, you can browse the entire API by starting at the root resource and following links to related resources.

HTTP Methods

All resources support a subset of these common HTTP Methods:

- `GET` - Retrieve the JSON representation of a resource.
- `POST` - Create a new resource using the provided JSON representation.
- `PUT` - Replace a resource with the provided JSON representation.
- `PATCH` - Update the specified fields in a resource using the provided JSON representation.
- `DELETE` - Remove a resource.

JSON

All entities are represented in JSON. The following rules and conventions apply:

- When sending JSON to the server via `POST` or `PUT`, make sure to specify the correct content type request header: `Content-Type: application/json`

- Invalid fields will be *rejected* rather than *ignored*. If, for example, you attempt to create a new entity and misspell one of the fields, or if you attempt to update an existing entity and include a field that cannot be modified, the server will respond with a 400 status code and an error message stating which field was invalid.
- All dates are returned as ISO-8601 formatted strings designated in UTC. When sending dates to the server (ie, as query parameters or fields in `POST` or `PATCH` request entities), use ISO-8601 formatted dates. If you do not specify a time zone, UTC is assumed. However, it is highly recommended that you include a time zone designator to avoid any ambiguity.
- In some cases, a timestamp will be returned as a BSON timestamp, most notably in the backup resources. These are represented in JSON documents as an object with two fields: `date`, an ISO-8601 formatted date string in UTC with granularity to the second, and `increment` a 32-bit integer.
- Fields that contain numeric values in a particular unit will be named so as to disambiguate the unit being used. For example, a host's uptime is returned in millseconds, so the name of the host entity field is `uptimeMsec`.
- Fields that do not have a current value will be returned with an appropriate default value. For example, Ops Manager will not have any statistics for a newly discovered host, so any statistics-related fields will have a value of zero. Fields that do not have a sensible default value will be omitted from the entity. For example, a host that is not using authentication will omit the `username` field from the returned entity.
- The fields in the JSON documents returned by the server are in no particular order, and it may change. Do not depend on the order of the fields.

Linking

Each resource includes one or more links to sub-resources and/or related resources. For example, a host has a link to the group it belongs to, the replica set it belongs to, and so on. Links are placed in the `links` field of an entity, which is an array of link relation objects. Each link relation has two fields:

- `rel` - Name (or type) of the relation. Many of these are considered Extension Relation Types and will be prefixed by *http://mms.mongodb.com*.
- `href` - The target URL.

All entities include at least one link relation called `self`, which is simply its own URL. When an entity is part of a list (ie, when requesting all hosts in a group), then it only includes the `self` link relation. Here's an example of a portion of a **host** resource with a few links:

```
{
  "id": "xxx",
  "groupId": "yyy",
  "hostname": "mongodb.foo.com",
  "port": 27017,
  // additional host properties...
  "links": [
    {
      "rel": "self",
      "href": "https://mms.mongodb.com/api/public/v1.0/groups/xxx/hosts/yyy"
    },
    {
      "rel": "http://mms.mongodb.com/group",
      "href": "https://mms.mongodb.com/api/public/v1.0/groups/xxx"
    }
  ]
}
```

For more information, refer to the Web Linking Specification. Note that although the specification describes a format for including links in the HTTP response headers, doing so is not a requirement. To make the API easily browsable, it

includes the links in the response body rather than in the response headers.

Lists

Some resources return a list of entities. For example, you can request a list of all **hosts** in a **group**. When a list of entities is expected in a response, the results will be returned in batches bounded by two query parameters:

- `pageNum` - Page number (1-based). Defaults to 1 if not specified.
- `itemsPerPage` - Maximum number of items to return, up to a maximum of 100. Defaults to 100 if not specified.

The response entity contains three fields:

- `totalCount` - The total number of items in the entire result set. For example, if a group has a total of 57 hosts, and you make a request with `pageNum=6` and `itemsPerPage=10`, then `totalCount` will be 57.
- `results` - The result set, which is an array of entity documents.
- `links` - Contains one to three link relations: `previous` for the previous page of results (omitted for the first page); `next` for the next page of results (omitted for the last page); `self` for the current page (always present).

If you make a request for a list of entities and there are no results, then the API will respond with a 200 status code and the `results` array will be empty. It does *not* respond with a 404 in this case, since the list of entities may not be empty at some point in the future. However, had you requested a list of entities in a context that does not exist (ie, the list of hosts for a non-existent group), then this *will* result in a 404 response status.

Here's an example respoonse for the second page of 10 hosts in a group with a total of 57 hosts:

```
{

  "totalCount": 57,
  "results": [
    {
      "id": "yyy",
      "groupId": "xxx",
      // additional host properties...
    },
    // additional host documents...
  ],
  "links": [
    {
      "rel": "previous",
      "href": "https://www.mongodb.com/api/public/v1.0/groups/xxx/hosts?
↪itemsPerPage=10&pageNum=1"
    },
    {
      "rel": "next",
      "href": "https://www.mongodb.com/api/public/v1.0/groups/xxx/hosts?
↪itemsPerPage=10&pageNum=3"
    }
  ]
}
```

Envelopes

Some clients may not be able to access the HTTP response headers and/or status code. In that case, you can request that the response include an "envelope," which is simply an extra layer of information in the JSON document that

contains any relevant details that would normally be in the response headers. By default, the API will *not* include the response in an envelope. To request one, simply add the query parameter `envelope=true`.

For responses that contain a single entity, the envelope will contain two fields:

- `status` - The HTTP status code.
- `content` - The requested entity.

For responses that contain a list of entities, there is already an envelope that wraps the results, so specifying `envelope=true` in this case will only add the `status` field to the existing envelope.

Pretty Printing

By default, extraneous whitespace is stripped from the JSON returned by the server. To ask for pretty-printed JSON, simply append the `pretty=true` query parameter to any request. Note that all the examples in this document show pretty-printed JavaScript for clarity, although the example URLs do not contain this additional query parameter.

Response Codes

Responses utilize the standard HTTP response codes, including:

Code	Meaning	Notes
200	OK	The request was successful. This is typically the response to a successful `GET` request.
201	Created	A new resource was created. This is typically the response to a succcessful `POST` request.
202	Accepted	A request for an asynchronous operation was accepted.
400	Bad Request	Something was wrong with the client request.
401	Unauthorized	Authentication is required but was not present in the request. Typically this means that the digest authentication information was omitted from the request.
403	Forbidden	Access to the specified resource is not permitted. Usually means that the user associated with the given API Key is not allowed to access the requested resource.
404	Not Found	The requested resource does not exist.
405	Method Not Allowed	The HTTP method is not supported for the specified resource. Keep in mind that each resource may only support a subset of HTTP methods. For example, you are not allowed to `DELETE` the root resource.
409	Conflict	This is typically the response to a request to create or modify a property of an entity that is unique when an existing entity already exists with the same value for that property. For example, attempting to create a group with the same name as an existing group is not allowed.
5xx	Various server errors	Something unexpected went wrong. Try again later and consider notifying Ops Manager Support.

Errors

When a request results in an error, the response body will contain a document with additional details about what went wrong. The document contains three fields:

- `error` - The error code, which is simply the HTTP status code.
- `reason` - A short description of the error, which is simply the HTTP status phrase.
- `detail` - A more detailed description of the error.

For example, here is the response body for a request for a host that does not exist:

```
{
  "error": 404,
  "reason": "Not Found",
  "detail": "No host exists with ID yyy in group xxx."
}
```

Authentication

As previously mentioned, the Ops Manager API uses HTTP Digest Authentication. The details of digest authentication are beyond the scope of this document, but it essentially requires a username and a password which are hashed using a unique server-generated value called a **nonce**. The username is the username of a registered Ops Manager account, and the password is an API Key associated to that username.

Keep the following points in mind:

- The server-generated nonce is used by the client to hash the username and password before sending them back to the server to authenticate a request. The nonce is only valid for a short amount of time as per the digest authentication specification. This is to prevent replay attacks, so you can't cache a nonce and use it forever.
- Some resource methods require even more security and are additionally protected by a whitelist, which is a list of client IP addresses associated to a user account that are permitted to access these protected resources.
- The Ops Manager UI has a concept of **roles**, which allow more fine-grained control of the operations a user is allowed to perform. The API resources also enforce the same authorization rules, so the resources and methods that can be accessed by an API Key are governed by the roles granted to the associated user. For example, to `DELETE` a host, the user that owns the API key used to make the request must be a `Monitoring Admin` or `Owner` in the group that the host belongs to.
- Many resources are tied to a group, as evidenced by URLs of the form `.../api/public/v1.0/groups/<GROUP-ID>/...`. For these resources, the user tied to the API key must be a member of the group *or* must be assigned to one of the `GLOBAL` roles. Otherwise the server will respond with a 403 (Forbidden) status.

Automation

The API provides endpoints that let you *modify a group's deployment* and *retrieve deployment status*. You can modify a deployment by sending a new automation configuration to Ops Manager. The automation configuration is where you describe and configure the MongoDB processes to be deployed. Ops Manager refers to this as the deployment's "goal state." When you submit a new automation configuration through the API, the Automation Agents adjust the current state of the system to match the goal state.

Important: There is no protection in the API to prevent concurrent modifications. If two administrators both start with a configuration based on the current version, make their own modifications, and then submit their modifications, the later modification wins.

Additional Information

See *Public API Resources* for a complete reference of all resources available in the Ops Manager Public API.

9.2 Public API Resources

The Ops Manager *Public API* exposes the following resources:

Root

On this page

This is the starting point (or the homepage, if you will) for the Ops Manager API. From here, you can traverse the `links` to reach all other API resources.

Sample Entity

```
{
  "throttling": false,
  "links": [ ... ]
}
```

Entity Fields

Name	Type	Description
throttling	boolean	Tells whether or not Ops Manager is throttling data. This can be used as a simple indicator of the current health of Ops Manager, since throttling is generally enabled when Ops Manager is in an unhealthy state.

Links

Relation	Description
self	Me
groups	Groups accessible to the current API user.
user	The current API user.

Example

Retrieve the root resource:

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0"
```

```
HTTP/1.1 200 OK

{
  "throttling" : false,
  "links" : [ ... ]
}
```

Hosts

On this page

You can typically access a host using a variety of names. DNS records and entries in the `/etc/hosts` file determine what names you can use to access a given host.

When you add a host to Ops Manager, Ops Manager automatically discovers various valid hostname and port combinations for each monitored `mongod` and `mongos` process. Ops Manager then ranks the hostnames to choose a "primary" hostname. Hostnames with the most periods are ranked highest, while the loopback address (`127.0.0.1`) and `localhost` lowest. Ops Manager treats the "losing" hostnames as host aliases.

When Ops Manager processes a ping from the Monitoring agent, the algorithm for assigning a primary hostname repeats. As a result, the primary hostname may change over time. You can also *specify preferred hostnames* in Ops Manager's group settings to override the hostname algorithm.

Operations

- `GET /api/public/v1.0/groups/GROUP-ID/hosts` - Get all hosts in a group. Use the `clusterId` query parameter to only get the hosts that belong to the specified cluster. The resulting list is sorted alphabetically by `hostname:port`.
- `GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID` - Get a single host by ID.
- `GET /api/public/v1.0/groups/GROUP-ID/hosts/byName/HOSTNAME:PORT` - Get a single host by its hostname and port combination. You can specify either the primary hostname or an alias.
- `POST /api/public/v1.0/groups/GROUP-ID/hosts` - Create a new host in the group. Note that after a new host is created, Ops Manager will not know much about it except what is provided. Thus, the document returned in the response will be missing many values until they are discovered, which could take several minutes. Only these fields may be specified when creating a host:
  - `hostname` - Required.
  - `port` - Required.
  - `username` - Required if `authMechanismName` is `MONGODB_CR`.Otherwise it's illegal.
  - `password` - Required if `authMechanismName` is `MONGODB_CR`. Otherwise it's illegal.
  - `sslEnabled` - Default is `false` if omitted.
  - `logsEnabled` - Default is `false` if omitted.
  - `alertsEnabled` - Default is `true` if omitted.
  - `profilerEnabled` - Default is `false` if omitted.
  - `muninPort` - Default is 0 and Munin stats are not collected if omitted.
  - `authMechanismName` - Default is `NONE` if omitted. If set to `MONGODB_CR` then you must provide the `username` and `password`.
- `PATCH /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID` - Update an existing host using the fields provided. Unspecified fields will preserve their current values.
  - Only these fields may be specified: `username password sslEnabled logsEnabled alertsEnabled profilerEnabled muninPort authMechanismName`
  - If `authMechanismName` is `NONE` then any existing value for `username` and `password` will be cleared out. For `MONGODB_CR` you must provide both `username` and `password`.
- `DELETE /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID` - Remove a host.

Sample Entity


```
{
  "id": "680ab316473d6b28f966364b947134fc",
  "groupId": "2847387cd717dabc348a",
  "hostname": "localhost",
  "port": 27017,
  "typeName": "SHARD_SECONDARY",
```

```
  "lastPing": "2014-02-15T16:03:47Z",
  "ipAddress": "127.0.0.1",
  "version": "2.4.3",
  "deactivated": false,
  "hasStartupWarnings": true,
  "sslEnabled": false,
  "logsEnabled": false,
  "lastReactivated": "2013-12-15T09:17:23Z",
  "uptimeMsec": 48918394,
  "lastRestart": "2014-01-16T12:34:01Z",
  "shardName": "sh1",
  "replicaSetName": "rs1",
  "replicaStateName": "RECOVERING",
  "created": "2013-11-05T03:04:05Z",
  "hostEnabled": true,
  "journalingEnabled": false,
  "alertsEnabled": true,
  "hidden": false,
  "muninEnabled": false,
  "profilerEnabled": false,
  "lowUlimit": false,
  "muninPort": 4949,
  "authMechanismName": "MONGODB_CR",
  "username": "mongo",
  "aliases": [ "127.0.0.1:27017" ],
  "links": [ ... ]
}
```


Entity Fields

Name	Type	Description
id	string	Unique identifier.
groupId	string	ID of the group that owns this host.
hostname	string	Primary hostname. A host typically has several aliases, so the primary is the best available name as decided by Ops Manager.
port	integer	Port that MongoDB process (`mongod` or `mongos`) listens on.
typeName	enum	Type for this host. Possible values are: • `STANDALONE` • `REPLICA_PRIMARY` • `REPLICA_SECONDARY` • `REPLICA_ARBITER` • `RECOVERING` • `MASTER` • `SLAVE` • `SHARD_MONGOS` • `SHARD_CONFIG` • `SHARD_STANDALONE` • `SHARD_PRIMARY` • `SHARD_SECONDARY` • `NO_DATA` The host's type for new hosts added to Ops Manager will be `NO_DATA` until the Monitoring Agent receives its first ping.

Continued on next page

Table 1 – continued from previous page

Name	Type	Description
lastPing	date	When the last ping for this host was received.
ipAddress	string	IP address of this host.
version	string	Version of MongoDB running on this host.
deactivated	boolean	Has this host been deactivated by Ops Manager? A host will be marked as deactivated when Ops Manager hasn't received a ping from it in several days.
hasStartupWarnings	boolean	Are there startup warnings for this host?
sslEnabled	boolean	Is SSL enabled for this host?
logsEnabled	boolean	Is Ops Manager collecting logs for this host?
lastReactivated	date	The last time this has was manually reactivated.
uptimeMsec	long	Number of milliseconds since this host's last restart.
lastRestart	date	Date this host was last restarted.
shardName	string	Name of the shard this host belongs to. Only present if the host is part of a sharded cluster.
replicaSetName	string	Name of the replica set this host belongs to. Only present if this host is part of a replica set.
replicaStateName	string	Current state of this host within a replica set. Only present if this host is part of a replica set. See Replica Set Member States for possible values.
created	date	Date this host was created or first discovered by Ops Manager.
hostEnabled	boolean	Is this host currently enabled? Hosts can be manually disabled in the Ops Manager UI.
journalingEnabled	boolean	Is journaling enabled for this host?
alertsEnabled	boolean	Are alerts enabled for this host?
muninEnabled	boolean	Are Munin stats being collected for this host?
hidden	boolean	Is this host currently hidden? When Ops Manager deactivates a host, it will also mark it as hidden.
profilerEnabled	boolean	Is Ops Manager collecting profile information from this host?
lowUlimit	boolean	Does this host have a low `ulimit` setting?
muninPort	integer	What port should be used to collect Munin stats from this host?
authMechanismName	string	The authentication mechanism used to connect to this host. Possible values are: • `MONGODB_CR` • `GSSAPI` • `NONE`
username	string	Username for connecting to this host. Only present when the `authMechanismName` is `MONGODB_CR`.
password	string	Password for connecting to this host. If a host's `authMechanismName` is `MONGODB_CR`, then you must include this field when creating the host or updating its credentials. However, it will never be exposed when a host entity is returned.
aliases	array of strings	A list of alternate hostname:port combinations that Ops Manager has discovered for the host.

Links

Relation	Description
self	Me
cluster	The cluster this host belongs to. Only present if the host is part of a replica set or master/slave.
parentCluster	The parent cluster. Only present if the host is part of a sharded cluster.
group	The group that this host belongs to.

Examples

Create a New Host

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST
↪"https://mms.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e/hosts" --
↪data '
{
  "hostname": "localhost",
  "port": 27017
}'

HTTP/1.1 201 Created
Location: https://mms.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e/
↪hosts/680ab316473d6b28f966364b947134fc

{
  "id" : "4059580c20c4581872ef24d0b8f5dca0",
  "groupId" : "5196d3628d022db4cbc26d9e",
  "hostname" : "localhost",
  "port" : 27017,
  "deactivated" : false,
  "hasStartupWarnings" : false,
  "sslEnabled" : false,
  "logsEnabled" : false,
  "created" : "2014-04-22T19:56:50Z",
  "hostEnabled" : true,
  "journalingEnabled" : false,
  "alertsEnabled" : true,
  "hidden" : false,
  "muninEnabled" : false,
  "profilerEnabled" : false,
  "lowUlimit" : false,
  "authMechanismName" : "NONE",
  "links" : [ ... ]
}
```

Update a Host

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X PATCH
↪"https://mms.mongodb.com/api/public/v1.0/groups/533c5895b91030606f21033a/hosts/
↪680ab316473d6b28f966364b947134fc" --data '
{
  "sslEnabled": true,
  "username": "mongo",
  "password": "M0ng0DB!:)"
}'

HTTP/1.1 200 OK

{
  "id" : "680ab316473d6b28f966364b947134fc",
  "groupId" : "533c5895b91030606f21033a",
  "hostname" : "localhost",
  "port" : 26000,
```

```
  "deactivated" : false,
  "hasStartupWarnings" : false,
  "sslEnabled" : true,
  "logsEnabled" : false,
  "created" : "2014-04-22T19:56:50Z",
  "hostEnabled" : true,
  "journalingEnabled" : false,
  "alertsEnabled" : true,
  "hidden" : false,
  "muninEnabled" : false,
  "profilerEnabled" : false,
  "lowUlimit" : false,
  "authMechanismName" : "MONGODB_CR",
  "username" : "mongo",
  "links" : [ ... ]
}
```

Get One Host

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪533c5895b91030606f21033a/hosts/56e9378f601dc49360a40949c8a6df6c"

HTTP/1.1 200 OK

{
  "id" : "56e9378f601dc49360a40949c8a6df6c",
  "groupId" : "533c5895b91030606f21033a",
  "hostname" : "mymongo.test.com",
  "port" : 26000,
  "deactivated" : false,
  "hasStartupWarnings" : false,
  "sslEnabled" : true,
  "logsEnabled" : false,
  "created" : "2014-04-22T19:56:50Z",
  "hostEnabled" : true,
  "journalingEnabled" : false,
  "alertsEnabled" : true,
  "hidden" : false,
  "muninEnabled" : false,
  "profilerEnabled" : false,
  "lowUlimit" : false,
  "authMechanismName" : "MONGODB_CR",
  "username" : "mongo",
  "aliases": [ "mymongo:26000", "12.34.56.78:26000" ]
  "links" : [ ... ]
}
```

Get All Hosts

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪533c5895b91030606f21033a/hosts"

HTTP/1.1 200 OK
```

```
{
  "totalCount" : 2,
  "results" : [
    {
      "id" : "56e9378f601dc49360a40949c8a6df6c",
      "groupId" : "533c5895b91030606f21033a",
      "hostname" : "mymongo.test.com",
      "port" : 26000,
      "deactivated" : false,
      "hasStartupWarnings" : false,
      "sslEnabled" : true,
      "logsEnabled" : false,
      "created" : "2014-04-22T19:56:50Z",
      "hostEnabled" : true,
      "journalingEnabled" : false,
      "alertsEnabled" : true,
      "hidden" : false,
      "muninEnabled" : false,
      "profilerEnabled" : false,
      "lowUlimit" : false,
      "authMechanismName" : "MONGODB_CR",
      "username" : "mongo",
      "aliases": [ "mymongo:26000", "12.34.56.78:26000" ]
      "links" : [ ... ]
    },
    {
      ...
    }
  ]
}
```

Delete a Host

```
curl -u "username:apiKey" --digest -i -X DELETE "https://mms.mongodb.com/api/public/
↪v1.0/groups/533c5895b91030606f21033a/hosts/680ab316473d6b28f966364b947134fc"

HTTP/1.1 200 OK
```

Metrics

On this page

Operations

- `GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/metrics` - Get a list of all available metrics for the host. Each entity in the list will be a partial metric entity. No actual data points are returned, but each entity contains a `self` link which you may follow to retrieve the full metric entity.
- `GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/metrics/METRIC-NAME` - Get the data points for the specified host and metric. If no additional query parameters are given, then the minute-level data for the past hour is returned. The `METRIC-NAME` may be any of the supported values listed for the `metricName` field, above. Note that if the provided metric is a database-level metric (ie, `DB_LOCK_PERCENTAGE`) or a hardware metric for a specific device (ie, its name begins with `MUNIN_IOSTAT`), then the response entity will contain a list of links to all available database (or hardware device) metrics. You may also provide additional query parameters:
  - `granularity` - The size of the epoch. Acceptable values are: `MINUTE HOUR DAY`.
  - `period` - The ISO-8601 formatted time period that specifies how far back in the past to query. For example, to request the last 36 hours of hour-level data, you must specify: `granularity=HOUR&period=P1DT12H`.
- `GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/metrics/DB-METRIC-NAME/DB-NAME` - Get the data points for the specified host, database metric, and database name. The only available database-level metric is `DB_LOCK_PERCENTAGE`. The same query parameters described above are also supported.
- `GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/metrics/HW-METRIC-NAME/DEVICE-NAME` - Get the data points for the specified host, hardware metric, and device name. The device-specific hardware metrics include the supported values for the `metricName` field that begin with `MUNIN_IOSTAT_`. The same query parameters described above are also supported.

Sample Entity

```
{
  "hostId": "680ab316473d6b28f966364b947134fc",
  "groupId": "2847387cd717dabc348a",
  "metricName" : "OPCOUNTERS_UPDATE",
  "units" : "RAW",
  "granularity" : "MINUTE",
  "dataPoints" : [ {
    "timestamp" : "2014-08-26T16:42:00Z",
    "value" : 10.3911
  }, {
    "timestamp" : "2014-08-26T16:43:00Z",
    "value" : 14.938
  }, {
    "timestamp" : "2014-08-26T16:44:00Z",
    "value" : 12.8882
  },
  ...
  ],
  "links" : [ ... ]
}
```

Entity Fields

Name	Type	Description
hostId	string	ID of the host to which this metric pertains.
groupId	string	ID of the group that owns this host.
metricName	enum	The name of the metric. Possible values are: • ASSERT_MSG • ASSERT_REGULAR • ASSERT_USER • ASSERT_WARNING • BACKGROUND_FLUSH_AVG • COMPUTED_MEMORY • CONNECTIONS • CURSORS_TOTAL_CLIENT_CURSORS_SIZE • CURSORS_TOTAL_OPEN • CURSORS_TOTAL_TIMED_OUT • DB_DATA_SIZE_TOTAL • DB_LOCK_PERCENTAGE • DB_STORAGE_TOTAL • EFFECTIVE_LOCK_PERCENTAGE • EXTRA_INFO_PAGE_FAULTS • GLOBAL_ACCESSES_NOT_IN_MEMORY • GLOBAL_LOCK_CURRENT_QUEUE_READERS • GLOBAL_LOCK_CURRENT_QUEUE_TOTAL • GLOBAL_LOCK_CURRENT_QUEUE_WRITERS • GLOBAL_PAGE_FAULT_EXCEPTIONS_THROWN • INDEX_COUNTERS_BTREE_ACCESSES • INDEX_COUNTERS_BTREE_HITS • INDEX_COUNTERS_BTREE_MISSES • INDEX_COUNTERS_BTREE_MISS_RATIO • JOURNALING_COMMITS_IN_WRITE_LOCK • JOURNALING_MB • MEMORY_MAPPED • MEMORY_RESIDENT • MEMORY_VIRTUAL • MUNIN_CPU_IOWAIT • MUNIN_CPU_IRQ • MUNIN_CPU_NICE • MUNIN_CPU_SOFTIRQ • MUNIN_CPU_STEAL • MUNIN_CPU_SYSTEM • MUNIN_CPU_USER • MUNIN_IOSTAT_OP_READ • MUNIN_IOSTAT_OP_WRITE • MUNIN_IOSTAT_TIME_READ • MUNIN_IOSTAT_TIME_WRITE • NETWORK_BYTES_IN • NETWORK_BYTES_OUT • NETWORK_NUM_REQUESTS • OPCOUNTERS_CMD • OPCOUNTERS_DELETE • OPCOUNTERS_GETMORE • OPCOUNTERS_INSERT • OPCOUNTERS_QUERY • OPCOUNTERS_REPL_CMD • OPCOUNTERS_REPL_INSERT • OPCOUNTERS_REPL_UPDATE • OPCOUNTERS_REPL_DELETE • OPCOUNTERS_UPDATE

Links

Relation	Description
self	Me
group	The group that the host belongs to.
host	The host to which the metric pertains.

Examples

Get All Available Metrics

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪51b9361d5ae9048f0aab01f4/hosts/04cf770dc43c9ff21747ecf71ff9ee78/metrics"

HTTP/1.1 200 OK

{
  "totalCount" : 53,
  "results" : [ {
    "hostId" : "04cf770dc43c9ff21747ecf71ff9ee78",
    "groupId" : "51b9361d5ae9048f0aab01f4",
    "metricName" : "ASSERT_REGULAR",
    "units" : "RAW",
    "links" : [ {
      "rel" : "self",
      "href" : "https://mms.mongodb.com/api/public/v1.0/groups/
↪51b9361d5ae9048f0aab01f4/hosts/04cf770dc43c9ff21747ecf71ff9ee78/metrics/ASSERT_
↪REGULAR"
    } ]
  }, {
    "hostId" : "04cf770dc43c9ff21747ecf71ff9ee78",
    "groupId" : "51b9361d5ae9048f0aab01f4",
    "metricName" : "ASSERT_WARNING",
    "units" : "RAW"
    "links" : [ {
      "rel" : "self",
      "href" : "https://mms.mongodb.com/api/public/v1.0/groups/
↪51b9361d5ae9048f0aab01f4/hosts/04cf770dc43c9ff21747ecf71ff9ee78/metrics/ASSERT_
↪WARNING"
    } ]
  }, ...  ],
  "links" : [ ... ]
}
```

Get a Single Metric

The following example gets hour-level data for the past 12 hours.

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪51b9361d5ae9048f0aab01f4/hosts/04cf770dc43c9ff21747ecf71ff9ee78/metrics/OPCOUNTERS_
↪QUERY?granularity=HOUR&period=PT12H"
```

```
HTTP/1.1 200 OK

{
  "groupId" : "51b9361d5ae9048f0aab01f4",
  "hostId" : "04cf770dc43c9ff21747ecf71ff9ee78",
  "metricName" : "OPCOUNTERS_QUERY",
  "units" : "RAW",
  "granularity" : "MINUTE",
  "dataPoints" : [ {
    "timestamp" : "2014-08-29T14:00:00Z",
    "value" : 381.2
  }, {
    "timestamp" : "2014-08-29T15:00:00Z",
    "value" : 407.23
  }, {
    "timestamp" : "2014-08-29T16:00:00Z",
    "value" : 365.3124
  }, ... ],
  "links" : [ ... ]
}
```

Clusters

On this page

MongoDB supports two different kinds of clusters: replica sets and sharded clusters. Since a shard within a sharded cluster is typically a replica set, a sharded cluster is a cluster of clusters. This relationship is reflected in the way Ops Manager models clusters, and it might lead to unexpected results from the **Clusters** resource. As an example, consider a deployment with one sharded cluster containing four shards, and each shard is a three-node replica set. In this scenario, the **Clusters** resource will return five entities: one that represents the sharded cluster, and four to represent the replica sets (shards). However, if each shard in this fictitious deployment was a standalone `mongod` instead of a replica set, then the **Clusters** resource would only return one entity representing the sharded cluster.

Operations

- `GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID` - Get a single cluster by ID.
- `GET /api/public/v1.0/groups/GROUP-ID/clusters` - Get all clusters in a group. Note that if Ops Manager hasn't received a ping from a cluster in several days, it will be considered inactive and will be filtered from this list. Use the `parentClusterId` query parameter to get all clusters with the specified parent cluster ID. The list of entities is sorted in ascending order by the date that Ops Manager discovered the cluster.
- `PATCH /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID` - Update a cluster by

ID. The only property that you may modify is the `clusterName`, since all other properties of a cluster are discovered by Ops Manager. Additionally, this operation is only permitted on clusters of type `SHARDED` and `SHARDED_REPLICA_SET`.

Sample Entity

```
{
  "id": "yyy",
  "groupId": "xxx",
  "typeName": "REPLICA_SET",
  "clusterName": "Cluster 0",
  "shardName": "shard001",
  "replicaSetName": "rs1",
  "lastHeartbeat": "2014-02-26T17:32:45Z",
  "links": [ ... ]
}
```

Entity Fields

Name	Type	Description
id	string	Unique identifier.
groupId	string	ID of the group that owns this cluster.
typeName	enum	Specifies what kind of cluster this is. Possible values are: • `MASTER_SLAVE` • `REPLICA_SET` • `SHARDED` • `SHARDED_REPLICA_SET`
clusterName	string	Display name of the cluster. Only applies to sharded clusters. Note that `mongod` itself doesn't allow you to name a cluster; this name is supplied by (and editable within) Ops Manager. For a replica set within a sharded cluster, the cluster name is the name of its parent cluster.
shardName	string	Name of the shard. Only present for a cluster of type `SHARDED` or `REPLICA_SET` that is part of a sharded cluster.
replicaSetName	string	Name of the replica set. Only present for a cluster of type `REPLICA_SET`.
lastHeartbeat	date	The approximate last time Ops Manager processed a ping from this cluster.

Links

Relation	Description
self	Me
parentCluster	The parent cluster. Only present if the type is `SHARDED` or `REPLICA_SET` within a sharded cluster.
group	The group that this cluster belongs to.
clusters	The member shards that belong to this cluster. Only present if the type is `SHARDED_REPLICA_SET`.
hosts	The member hosts that belong to this cluster. Present for all types except `SHARDED_REPLICA_SET`. Note: to get the hosts of a sharded cluster, follow the `clusters` link and get the `hosts` for each shard.

Examples

Get a Cluster

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪533c5895b91030606f21033a/clusters/533d7d4730040be257defe88"

HTTP/1.1 200 OK

{
  "id" : "533d7d4730040be257defe88",
  "typeName" : "SHARDED_REPLICA_SET",
  "clusterName" : "Animals",
  "lastHeartbeat" : "2014-04-03T15:26:58Z",
  "links" : [ ... ]
}
```

Get all Clusters

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪533c5895b91030606f21033a/clusters"

HTTP/1.1 200 OK

{
  "totalCount" : 3,
  "results" : [ {
    "id" : "533d7d4730040be257defe88",
    "typeName" : "SHARDED_REPLICA_SET",
    "clusterName" : "Animals",
    "lastHeartbeat" : "2014-04-03T15:26:58Z",
    "links" : [ ... ]
  }, {
    "id" : "533d7d4630040be257defe85",
    "typeName" : "REPLICA_SET",
    "clusterName" : "Animals",
    "shardName" : "cats",
    "replicaSetName" : "cats",
    "lastHeartbeat" : "2014-04-03T15:24:54Z",
    "links" : [ ... ]
  }, {
    "id" : "533d7d4630040be257defe83",
    "typeName" : "REPLICA_SET",
    "clusterName" : "Animals",
    "shardName" : "dogs",
    "replicaSetName" : "dogs",
    "lastHeartbeat" : "2014-04-03T15:26:30Z",
    "links" : [ ... ]
  } ],
 "links" : [ ... ]
}
```

Update a Cluster

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X PATCH
↪"https://mms.mongodb.com/api/public/v1.0/groups/533c5895b91030606f21033a/clusters/
↪533d7d4730040be257defe88" --data '
{
  "clusterName": "Zoo"
}'

HTTP/1.1 200 OK

{
  "id" : "533d7d4730040be257defe88",
  "typeName" : "SHARDED_REPLICA_SET",
  "clusterName" : "Zoo",
  "lastHeartbeat" : "2014-04-03T15:26:58Z",
  "links" : [ ... ]
}
```

Groups

On this page

Operations

- `GET /api/public/v1.0/groups/GROUP-ID` - Get a single group by ID.
- `GET /api/public/v1.0/groups` - Get all groups for the current user.
- `POST /api/public/v1.0/groups` - Create a new group. Only the `name` field may be specified. The `publicApiEnabled` field will be set to `true` for groups created with the API. The response entity will include the `agentApiKey` for the group.
- `GET /api/public/v1.0/groups/GROUP-ID/users` - Get all users in a group.
- `DELETE /api/public/v1.0/groups/GROUP-ID/users/USER-ID` - Remove a user from a group.
- `POST /api/public/v1.0/groups/GROUP-ID/users` - Add existing user(s) to a group.
  - You must send an array of entities, even if you're only adding a single user.
  - For each user being added, specify the user ID and role(s) to be assigned.
  - If a user is specified that is already part of the group, then their existing role(s) will be overwritten.

- `DELETE /api/public/v1.0/groups/GROUP-ID` - Delete a group. Once a group is deleted, its name cannot be reclaimed. Thus, if you create a group named **My Group** and then delete it, you will not be able to create another group named **My Group**.

Sample Entity

```
{
  "id": "xxx",
  "name": "My Group",
  "hostCounts": {
    "arbiter": 2,
    "config": 1,
    "primary": 4,
    "secondary": 8,
    "mongos": 2.
    "master": 0,
    "slave": 0
  },
  "lastActiveAgent": ISODate("2014-02-05T07:23:34Z"),
  "activeAgentCount": 1,
  "replicaSetCount": 3,
  "shardCount": 2,
  "publicApiEnabled": true,
  "agentApiKey": "cbd728abd6a6d6c6b6d7826345dbcff0e",
  "links": [ ... ]
}
```

Entity Fields

Name	Type	Description
id	string	Unique identifier.
name	string	Display name for the group.
hostCounts	object	The total number of hosts by type. The embedded fields should be self-explanatory.
lastActiveAgent	date	Date that a ping was last received from one of the group's Monitoring Agents.
activeAgentCount	integer	Number of Monitoring Agents sending regular pings to Ops Manager.
replicaSetCount	integer	Total number of replica sets for this group.
shardCount	integer	Total number of shards for this group.
publicApiEnabled	boolean	Is the Public API enabled for this group? This is a read-only field that will always be true for groups created with the API. Note that for groups created in the Ops Manager UI, the only way to set this flag to `true` is by enabling the Public API for the group in the **Settings** tab.
agentApiKey	string	The API key for your agent. This field is only present in the response entity to a `POST` request. Thus, the API key will only be exposed at group creation time.

Links

Relation	Description
self	Me
hosts	All hosts in the group.
users	All users in the group.
clusters	All clusters in the group.
alerts	All open alerts for the group.
alertConfigs	All alert configurations for the group.
backupConfigs	All backup configurations for the group.

Examples

Get a Group

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e"

HTTP/1.1 200 OK

{
  "id" : "5196d3628d022db4cbc26d9e",
  "name" : "API Example",
  "hostCounts" : {
    "arbiter" : 0,
    "config" : 1,
    "primary" : 3,
    "secondary" : 4,
    "mongos" : 2,
    "master" : 0,
    "slave" : 0
  },
  "lastActiveAgent" : "2014-04-03T18:18:12Z",
  "activeAgentCount" : 1,
  "replicaSetCount" : 3,
  "shardCount" : 2,
  "links" : [ ... ]
}
```

Get All Groups for Current User

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups"

HTTP/1.1 200 OK

{
  "totalCount" : 6,
  "results" : [ {
    "id" : "5196d3628d022db4cbc26d9e",
    "name" : "API Example",
    "hostCounts" : {
      "arbiter" : 0,
```

```
      "config" : 1,
      "primary" : 3,
      "secondary" : 4,
      "mongos" : 2,
      "master" : 0,
      "slave" : 0
    },
    "lastActiveAgent" : "2014-04-03T18:18:12Z",
    "activeAgentCount" : 1,
    "replicaSetCount" : 3,
    "shardCount" : 2,
    "links" : [ ... ]
  }, {
    // etc.
  } ],
  "links" : [ ... ]
}
```

Create a Group

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X POST
↪"https://mms.mongodb.com/api/public/v1.0/groups" --data '
{
  "name": "API Example 2"
}'

HTTP/1.1 201 Created
Location: https://mms.mongodb.com/api/public/v1.0/groups/533daa30879bb2da07807696

{
  "id" : "533daa30879bb2da07807696",
  "name" : "API Example 2",
  "activeAgentCount" : 0,
  "replicaSetCount" : 0,
  "shardCount" : 0,
  "publicApiEnabled": true,
  "agentApiKey": "cbd747d7b7b711de45aa3ff0e",
  "hostCounts" : {
    "arbiter" : 0,
    "config" : 0,
    "primary" : 0,
    "secondary" : 0,
    "mongos" : 0,
    "master" : 0,
    "slave" : 0
  },
  "links" : [ ... ]
}
```

Add Users to a Group

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X POST
↪"https://mms.mongodb.com/api/public/v1.0/groups/533daa30879bb2da07807696/users" --
↪data '
```

```
[
  {
    "id": "5329c8dfe4b0b07a83d67e7d",
    "roles": [{
      "roleName": "GROUP_READ_ONLY"
    }]
  },
  {
    "id": "5329c906e4b0b07a83d691ba",
    "roles": [{
      "roleName": "GROUP_MONITORING_ADMIN"
    }, {
      "roleName": "GROUP_BACKUP_ADMIN"
    }]
  }
]'

HTTP/1.1 200 OK
```

Delete a Group

```
curl -u "username:apiKey" --digest -i -X DELETE "https://mms.mongodb.com/api/public/
↪v1.0/groups/533daa30879bb2da07807696"

HTTP/1.1 200 OK
```

Get Users in a Group

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪5356823bc0edc2788a835ed0/users"

HTTP/1.1 200 OK

{
  "totalCount" : 2,
  "results" : [ {
    "id" : "5357e25a300490374243f425",
    "username" : "user1@foo.com",
    "emailAddress" : "user1@foo.com",
    "firstName" : "User",
    "lastName" : "One",
    "roles" : [ {
      "groupId" : "5356823bc0edc2788a835ed0",
      "roleName" : "GROUP_USER_ADMIN"
    } ],
    "links" : [ ... ]
  }, {
    "id" : "5356823b3004dee37132bb7b",
    "username" : "user2@foo.com",
    "emailAddress" : "user2@foo.com",
    "firstName" : "User",
    "lastName" : "Deux",
    "roles" : [ {
      "groupId" : "5356823bc0edc2788a835ed0",
```

```
      "roleName" : "GROUP_OWNER"
    }, {
      "groupId" : "5356823bc0edc2788a835ecd",
      "roleName" : "GROUP_OWNER"
    } ],
    "links" : [ ... ]
  } ],
  "links" : [ ... ]
}
```

Delete a User from a Group

```
curl -u "username:apiKey" --digest -i -X DELETE "https://mms.mongodb.com/api/public/
↪v1.0/groups/5356823bc0edc2788a835ed0/users/5357e25a300490374243f425"

HTTP/1.1 200 OK
```

Users

On this page

Overview

The `users` resource allows you to create and update users and retrieve user information. This resource also provides an endpoint for creating the first user in a system and retrieving an API key for use in future API calls. The endpoint for creating the first user is the only endpoint you can use without first having an API key.

Endpoints

Get a User by ID

Get a single user by ID. You can always retrieve your own user account. Otherwise, you must be a *global user* or you must have the *user admin role* in at least one group that is common between you and the user you are retrieving.

```
GET /api/public/v1.0/users/USER-ID
```

Get a User by Name

Get a single user by name. You can always retrieve your own user account. Otherwise, you must be a *global user* or you must have the *user admin role* in at least one group that is common between you and the user you are retrieving.

```
GET /api/public/v1.0/users/byName/USER-NAME
```

Get All Users in a Group

Retrieve all users in a group.

```
GET /api/public/v1.0/groups/GROUP-ID/users
```

Create the First User

This endpoint is available only when the Ops Manager instance has no users. This is the only API call you can make without first having an API key.

The user created through this endpoint is automatically granted the *GLOBAL_OWNER* role. The returned document includes the new user's API key, which you can use to make further API calls.

The endpoint does not create a group, but you can use the new user and API key to create a group through the *Groups* resource in the API. You cannot login to Ops Manager until after you have created a group.

```
POST /api/public/v1.0/unauth/users
```

Create a User

Create a new user. All fields are required.

```
POST /api/public/v1.0/users
```

Update a User

Update an existing user using the fields provided. Unspecified fields will preserve their current values. You cannot specify the password for security reasons.

```
PATCH /api/public/v1.0/users/USER-ID
```

Sample Entity

```
{
  "id": "xxx",
  "username": "somebody",
  "password": "abc123",
  "emailAddress": "somebody@somewhere-else.com",
  "mobileNumber": "2125551234",
  "firstName": "John",
  "lastName": "Doe",
```

```
  "roles": [
      {
        "groupId": "8491812938cbda83918c",
        "roleName": "GROUP_OWNER"
      },
      {
        "groupId": "4829cbda839cbdac3819",
        "roleName": "GROUP_READ_ONLY"
      }
  ],
  "links": [ ... ]
}
```

Entity Fields

Name	Type	Description
id	string	Unique identifier.
username	string	Ops Manager username.
password	string	Password. This field is NOT included in the entity returned from the server. It can only be sent in the entity body when creating a new user.
emailAddress	string	Email address.
mobileNumber	string	Mobile number. This field can only be set or edited using the Ops Manager UI because it is tied to two factor authentication.
firstName	string	First name.
lastName	string	Last name.
roles	object array	Role assignments.
roles.groupId	string	The groupId in which the user has the specified role. Note that for the "global" roles (those whose name starts with `GLOBAL_`) there is no `groupId` since these roles are not tied to a group.
roles.roleName	enum	The name of the role. Possible values are: • `GLOBAL_AUTOMATION_ADMIN` • `GLOBAL_BACKUP_ADMIN` • `GLOBAL_MONITORING_ADMIN` • `GLOBAL_OWNER` • `GLOBAL_READ_ONLY` • `GLOBAL_USER_ADMIN` • `GROUP_AUTOMATION_ADMIN` • `GROUP_BACKUP_ADMIN` • `GROUP_MONITORING_ADMIN` • `GROUP_OWNER` • `GROUP_READ_ONLY` • `GROUP_USER_ADMIN`

Links

Relation	Description
self	Me
whitelist	The user's whitelist.

Examples

Get a User by ID

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/users/
↪533dc19ce4b00835ff81e2eb"

HTTP/1.1 200 OK

{
  "id" : "533dc19ce4b00835ff81e2eb",
  "username" : "jane",
  "emailAddress" : "jane@qa.example.com",
  "firstName" : "Jane",
  "lastName" : "D'oh",
  "roles" : [ {
    "groupId" : "533daa30879bb2da07807696",
    "roleName" : "GROUP_USER_ADMIN"
  } ],
  "links": [ ... ]
}
```

Get a User by Name

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/
↪users/byName/jane"

HTTP/1.1 200 OK

{
  "emailAddress" : "jane@qa.example.com",
  "firstName" : "Jane",
  "id" : "533dc19ce4b00835ff81e2eb",
  "lastName" : "D'oh",
  "roles" : [ {
    "groupId" : "533daa30879bb2da07807696",
    "roleName" : "GROUP_USER_ADMIN"
  } ],
  "links": [ ... ]
}
```

Get All Users in a Group

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪533daa30879bb2da07807696/users"

HTTP/1.1 200 OK

{
  "totalCount": 3,
  "results": [ {
    "id" : "5329c8dfe4b0b07a83d67e7d",
    "username" : "jdoe",
```

```
    "emailAddress" : "jdoe@example.com",
    "firstName" : "John",
    "lastName" : "Doe",
    "roles" : [ {
      "groupId" : "5329cb6e879bb2da07806511",
      "roleName" : "GROUP_OWNER"
    }, {
      "groupId" : "5196d3628d022db4cbc26d9e",
      "roleName" : "GROUP_READ_ONLY"
    }, {
      "groupId" : "533daa30879bb2da07807696",
      "roleName" : "GROUP_READ_ONLY"
    } ],
    "links": [ ... ]
  }, {
    // etc.
  } ],
  "links": [ ... ]
}
```

Create a User

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X POST
↪"https://mms.mongodb.com/api/public/v1.0/users" --data '
{
  "username": "jane.doe@mongodb.com",
  "emailAddress": "jane.doe@mongodb.com",
  "firstName": "Jane",
  "lastName": "Doe",
  "password": "M0ng0D8!:)",
  "roles": [{
    "groupId": "533daa30879bb2da07807696",
    "roleName": "GROUP_USER_ADMIN"
  }]
}'

HTTP/1.1 201 Created
Location: https://mms.mongodb.com/api/public/v1.0/users/533dc19ce4b00835ff81e2eb

{
  "id" : "533dc19ce4b00835ff81e2eb",
  "username" : "jane.doe@mongodb.com",
  "emailAddress" : "jane.doe@mongodb.com",
  "firstName" : "Jane",
  "lastName" : "Doe",
  "roles" : [ {
    "groupId" : "533daa30879bb2da07807696",
    "roleName" : "GROUP_USER_ADMIN"
  } ],
  "links" : [ ... ]
}
```

Create the First User

```
curl -H "Content-Type: application/json" -i -X POST "http://<ops-manager-url>:<port>/
↪api/public/v1.0/unauth/users" --data '
{
  "username": "jane.doe@mongodb.com",
  "emailAddress": "jane.doe@mongodb.com",
  "password": "Passw0rd.",
  "firstName": "Jane",
  "lastName": "Doe"
}'

HTTP/1.1 200 OK

{
  "user": {
    "username": "jane.doe@mongodb.com",
    "roles": [
      {
        "roleName": "GLOBAL_OWNER"
      }
    ],
    "lastName": "Doe",
    "id": "533dc19ce4b00835ff81e2eb",
    "firstName": "Jane",
    "emailAddress": "jane.doe@mongodb.com",
    "links": [ ... ]
  },
  "apiKey": "1234abcd-ab12-cd34-ef56-1234abcd1234"
}
```

Create a User

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X POST
↪"https://mms.mongodb.com/api/public/v1.0/users" --data '
{
  "username": "jane",
  "emailAddress": "jane.doe@mongodb.com",
  "firstName": "Jane",
  "lastName": "Doe",
  "password": "M0ng0D8!:)",
  "roles": [{
    "groupId": "533daa30879bb2da07807696",
    "roleName": "GROUP_USER_ADMIN"
  }]
}'

HTTP/1.1 201 Created
Location: https://mms.mongodb.com/api/public/v1.0/users/533dc19ce4b00835ff81e2eb

{
  "id" : "533dc19ce4b00835ff81e2eb",
  "username" : "jane",
  "emailAddress" : "jane.doe@mongodb.com",
  "firstName" : "Jane",
  "lastName" : "Doe",
```

```
  "roles" : [ {
    "groupId" : "533daa30879bb2da07807696",
    "roleName" : "GROUP_USER_ADMIN"
  } ],
  "links" : [ ... ]
}
```

Update a User

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X PATCH
↪"https://mms.mongodb.com/api/public/v1.0/users/533dc19ce4b00835ff81e2eb" --data '
{
  "emailAddress": "jane@qa.example.com",
  "lastName": "D'oh"
}'

HTTP/1.1 200 OK

{
  "id" : "533dc19ce4b00835ff81e2eb",
  "username" : "jane",
  "emailAddress" : "jane@qa.example.com",
  "firstName" : "Jane",
  "lastName" : "D'oh",
  "roles" : [ {
    "groupId" : "533daa30879bb2da07807696",
    "roleName" : "GROUP_USER_ADMIN"
  } ],
  "links" : [ ... ]
}
```

Alerts

On this page

Operations

- `GET /api/public/v1.0/groups/GROUP-ID/alerts` - Gets all alerts with the specified status. Use the `status` query parameter with one of these possible values: `OPEN CLOSED`
- `GET /api/public/v1.0/groups/GROUP-ID/alerts/ALERT-ID` - Get a single alert by ID.

- `GET /api/public/v1.0/groups/GROUP-ID/alerts/ALERT-ID/alertConfigs` - Get the alert configuration(s) that triggered this alert.
- `PATCH /api/public/v1.0/groups/GROUP-ID/alerts/ALERT-ID` - Update an existing alert. The only field you may modify is the `acknowledgedUntil` field.
  - To acknowledge an alert "forever" set the date to 100 years in the future.
  - To unacknowledge a previously acknowledged alert, set the date in the past.

Sample Entity


```
{
  "id": "yyy",
  "groupId": "xxx",
  "alertConfigId": "xxx",
  "typeName": "HOST_METRIC",
  "eventTypeName": "OUTSIDE_METRIC_THRESHOLD",
  "status": "OPEN",
  "acknowledgedUntil": "2014-03-01T12:00:00Z",
  "created": "2014-02-01T12:34:12Z",
  "updated": "2014-02-02T01:23:45Z",
  "resolved": null,
  "lastNotified": "2014-02-04T02:43:13Z",
  "currentValue": {
    "number": 123.45,
    "units": "MEGABYTES"
  },
  "links": [ ... ]
}
```

Entity Fields

Name	Type	Description
id	string	Unique identifier.
groupId	string	ID of the group that this alert was opened for.
alertConfigId	string	ID of the alert configuration that triggered this alert.
typeName	enum	The type of alert. Possible values are: • AGENT • BACKUP • HOST • HOST_METRIC • REPLICA_SET
eventTypeName	enum	The name of the event that triggered the alert. The possible values here depend on the typeName: • **AGENT** - Possible values: – MONITORING_AGENT_DOWN – BACKUP_AGENT_DOWN • **HOST** - Possible values: – HOST_DOWN – HOST_RECOVERING – VERSION_BEHIND – HOST_EXPOSED • **HOST_METRIC** - Possible values: – OUTSIDE_METRIC_THRESHOLD • **BACKUP** - Possible values: – OPLOG_BEHIND – RESYNC_REQUIRED
status	enum	The current state of the alert. Possible values are: • OPEN • CLOSED
acknowledgedUntil	date	The date through which the alert has been acknowledged. Will not be present if the alert has never been acknowledged.
acknowledgementComment	string	The comment left by the user who acknowledged the alert. Will not be present if the alert has never been acknowledged.
acknowledgingUsername	string	The username of the user who acknowledged the alert. Will not be present if the alert has never been acknowledged.
created	date	When the alert was opened.
updated	date	When the alert was last updated.
resolved	date	When the alert was closed. Only present if the status is CLOSED.
lastNotified	date	When the last notification was sent for this alert. Only present if notifications have been sent.
metricName	enum	The name of the metric whose value went outside the threshold. Only present for alerts of type HOST_METRIC. Possible values are: • ASSERT_REGULAR • ASSERT_WARNING • ASSERT_MSG • ASSERT_USER • OPCOUNTER_CMD • OPCOUNTER_QUERY • OPCOUNTER_UPDATE • OPCOUNTER_DELETE
		• OPCOUNTER_INSERT • OPCOUNTER_REPL_UPDATE • OPCOUNTER_REPL_DELETE • OPCOUNTER_REPL_INSERT • MEMORY_RESIDENT

Links

Relation	Description
self	Me
group	The group that this alert was triggered for.
alertConfig	The alert configuration that triggered this alert.
alertConfigs	A list of alert configurations that triggered this alert. This list will only contain a single element and is present for backward compatibility. New code should use the `alertConfig` link instead.
host	The host that triggered this alert. Only present for alerts of type `HOST`.

Examples

Get an Alert

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e/alerts/533cb4b8e4b0f1820cdabc7f"

HTTP/1.1 200 OK

{
  "id" : "533cb4b8e4b0f1820cdabc7f",
  "groupId" : "5196d3628d022db4cbc26d9e",
  "typeName" : "BACKUP",
  "eventTypeName" : "OPLOG_BEHIND",
  "status" : "CLOSED",
  "created" : "2014-04-03T01:09:12Z",
  "updated" : "2014-04-03T01:14:12Z",
  "resolved" : "2014-04-03T01:14:12Z",
  "links" : [ ... ]
}
```

Get Open Alerts

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e/alerts?status=OPEN"

HTTP/1.1 200 OK

{
  "totalCount": 1,
  "results": [ {
    "alertConfigId":"55e756a6e4b0fa1210c695a2",
    "id" : "533dc45ee4b00835ff81ec2a",
    "groupId" : "5196d3628d022db4cbc26d9e",
    "hostId":"51714c7fa22cbe473a9573d3629ff53c",
    "hostnameAndPort":"example.test.4182.mongodbdns.com:27017",
    "typeName" : "HOST_METRIC",
    "eventTypeName" : "OUTSIDE_METRIC_THRESHOLD",
    "status" : "OPEN",
    "created" : "2014-04-03T20:28:14Z",
    "updated" : "2014-04-03T20:28:14Z",
    "lastNotified" : "2014-04-03T20:28:23Z",
```

```
    "metricName": "ASSERTS_REGULAR",
    "currentValue" : {
      "number" : 0.0,
      "units" : "RAW"
    },
    "links" : [ ... ]
  } ],
  "links" : [ ... ]
}
```

Get Alert Configurations that Triggered an Alert

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e/alerts/533cb4b8e4b0f1820cdabc7f/alertConfigs"

HTTP/1.1 200 OK

{
  "totalCount": 3,
  "results": [ {
    "id" : "5271259ee4b00ece6b4754ef",
    "groupId" : "5196d3628d022db4cbc26d9e",
    "typeName" : "BACKUP",
    "eventTypeName" : "RESYNC_REQUIRED",
    "created" : "2013-10-30T15:28:30Z",
    "updated" : "2014-02-12T16:11:05Z",
    "enabled" : true,
    "matchers" : [ ],
    "notifications" : [ {
      "typeName" : "EMAIL",
      "intervalMin" : 60,
      "delayMin" : 0,
      "emailAddress" : "somebody@somewhere.com"
    } ],
    "links" : [ ... ]
  } ],
  "links" : [ ... ]
}
```

Acknowledge an Alert

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X PATCH
↪"https://mms.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e/alerts/
↪533dc45ee4b00835ff81ec2a" --data '
{
  "acknowledgedUntil": "2014-04-15T00:00:00-0400",
  "acknowledgementComment": "This is normal. Please ignore."
}'

HTTP/1.1 200 OK

{
  "id" : "533dc45ee4b00835ff81ec2a",
  "groupId" : "5196d3628d022db4cbc26d9e",
```

```
  "typeName" : "HOST_METRIC",
  "eventTypeName" : "OUTSIDE_METRIC_THRESHOLD",
  "status" : "OPEN",
  "acknowledgedUntil" : "2014-04-15T04:00:00Z",
  "acknowledgementComment" : "This is normal. Please ignore.",
  "acknowledgingUsername" : "someuser@yourcompany.com",
  "created" : "2014-04-03T20:28:14Z",
  "updated" : "2014-04-03T20:33:14Z",
  "lastNotified" : "2014-04-03T20:33:23Z",
  "metricName": "ASSERTS_REGULAR",
  "currentValue" : {
    "number" : 0.0,
    "units" : "RAW"
  },
  "links" : [ ... ]
}
```

Alert Configurations

On this page

Operations

```
GET /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID
```

- Get a single alert configuration by ID.

```
GET /api/public/v1.0/groups/GROUP-ID/alertConfigs
```

- Get all alert configurations for a group.

```
GET /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID/alerts
```

- Get all open alerts that were triggered by an alert configuration.

```
POST /api/public/v1.0/groups/GROUP-ID/alertConfig
```

- Create a new alert configuration. All fields are required except `created` and `updated`.

```
PUT /api/public/v1.0/groups/GROUP-ID/alertConfigs
```

- Update an existing alert configuration. Partial updates are not supported except for one field (see `PATCH` below), so you must send the entire entity.

```
PATCH /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID
```

- Use to enable/disable an alert configuration by setting the `enabled` field.

```
DELETE /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID
```

- Remove an alert configuration.

Sample Entity


```
{
  "id": "yyy",
  "groupId": "xxx",
  "typeName": "HOST_METRIC",
  "eventTypeName": "OUTSIDE_METRIC_THRESHOLD",
  "created": "2014-02-01T12:34:12Z",
  "updated": "2014-02-02T01:23:45Z",
  "enabled": true,
  "matchers": [{
    "fieldName": "HOSTNAME",
    "operator": "STARTS_WITH",
    "value": "my-host-prefix"
  }, {
    "fieldName": "PORT",
    "operator": "EQUALS",
    "value": "27017"
  }],
  "notifications": [{
    "typeName": "EMAIL",
    "intervalMin": 5,
    "delayMin": 0,
    "emailAddress": "somebody@somewhere.com"
  }, {
    "typeName": "HIP_CHAT",
    "intervalMin": 5,
    "delayMin": 0,
    "notificationToken": "123456abcdef",
    "roomName": "|mms| Test Chat Room"
  }, {
    "typeName": "GROUP",
    "intervalMin": 5,
    "delayMin": 0,
    "groupId": "2847387cd717dabc348a",
    "groupName": "test1",
    "emailEnabled": true,
    "smsEnabled": true
  }, {
    "typeName": "USER",
    "intervalMin": 5,
    "delayMin": 0,
    "username": "john.doe",
    "emailEnabled": true,
    "smsEnabled": true
  }, {
    "typeName": "SMS",
    "intervalMin": 5,
```

```
    "delayMin": 0,
    "mobileNumber": "(212) 212-1212"
  }, {
    "typeName": "SNMP",
    "intervalMin": 5,
    "delayMin": 0,
    "snmpAddress": "somedomain.com:161"
  }, {
    "typeName": "PAGER_DUTY",
    "intervalMin": 5,
    "delayMin": 0,
    "serviceKey": "123456abcdef"
  }],
  "metricThreshold": {
    "metricName": "MEMORY_RESIDENT",
    "operator": "GREATER_THAN",
    "threshold": 7,
    "units": "GIGABYTES",
    "mode": "TOTAL"
  },
  "links": [ ... ]
}
```


Entity Fields

Name	Type	Description
id	string	Unique identifier.
groupId	string	ID of the group that owns this alert configuration.
typeName	enum	The type of this alert configuration. Supports the same values as the `typeName` field of the **alerts** resource.
eventTypeName	enum	The type of event that will trigger an alert. Supports the same values as the `eventTypeName` field of the **alerts** resource.
created	date	When this alert configuration was created.
updated	date	When this alert configuration was last updated.
enabled	boolean	Is this alert configuration enabled?
matchers	object array	Rules to apply when matching an object against this alert configuration. Only entities that match *all* these rules will be checked for an alert condition.
matchers.fieldName	string	The name of the field in the target object to match on. The available fields depend on the `typeName`: • **AGENT** - Not applicable. • **BACKUP** - Not applicable. • **HOST** and **HOST_METRIC** - Possible values are: – `HOSTNAME` – `PORT` – `HOSTNAME_AND_PORT` – `REPLICA_SET_NAME` – `TYPE_NAME` • **REPLICA_SET** - Possible values are: – `REPLICA_SET_NAME` – `SHARD_NAME` – `CLUSTER_NAME`

Continued on next page

Table 2 – continued from previous page

Name	Type	Description
matchers.operator	enum	The operator to test the field's value. Possible values are: • EQUALS • NOT_EQUALS • CONTAINS • NOT_CONTAINS • STARTS_WITH • ENDS_WITH • REGEX
matchers.value	string	The value to test with the specified operator. When matching on the TYPE_NAME field for a HOST or HOST_METRIC alert, the possible typeName values are: • PRIMARY • SECONDARY • STANDALONE • CONFIG • MONGOS
notifications	object array	Notifications to send when an alert condition is detected.
notifications.typeName	string	The type of alert notification. Possible values are: • GROUP • USER • SMS (Only available to Ops Manager installations) • EMAIL • PAGER_DUTY • HIPCHAT • SNMP
notifications.delayMin	integer	The number of minutes to wait after an alert condition is detected before sending out the first notification.
notifications.intervalMin	integer	The number of minutes to wait between successive notifications for unacknowledged alerts that are not resolved.
notifications.emailAddress	string	The email address to which to send notification. Only present for notifications of type EMAIL.
notifications.notificationToken	string	The HipChat API token. Only present for notifications of type HIP_CHAT.
notifications.roomName	string	HipChat room name. Only present for notifications of type HIP_CHAT.
notifications.emailEnabled	boolean	Should email notifications be sent? Only present for notifications of type GROUP and USER.
notifications.smsEnabled	boolean	Should SMS notifications be sent? Only present for notifications of type GROUP and USER.
notifications.username	string	The name of an Ops Manager user to which to send notifications. Only a user in the group that owns the alert configuration is allowed here. Only present for notifications of type USER.
notifications.snmpAddress	string	Hostname and port to send SNMP traps to. At this time Ops Manager is only able to send SNMP traps to the standard SNMP port (161). Only present for SNMP notifications. Ops Manager uses SNMP v2c.
notifications.emailAddress	string	Mobile number to send SMS messages to. Only present for notifications of type SMS.
notifications.snmpAddress	string	Hostname and port to send SNMP traps to. Note that SNMP is only supported for Ops Manager; also, at this time Ops Manager is only able to send SNMP traps to the standard SNMP port (161).
notifications.serviceKey	string	PagerDuty service key.

Continued on next page

Table 2 – continued from previous page

Name	Type	Description
metricThreshold	object	The threshold that will cause an alert to be triggered. Only present for alerts of the `HOST_METRIC`.
metricThreshold.metricName	string	The name of the metric to check. Supports the same values as the `metricName` field of the **alerts** resource.
metricThreshold.operator	string	The operator to apply when checking the current metric value against the threshold value. Possible values are: • `GREATER_THAN` • `LESS_THAN`
metricThreshold.threshold	number	The threshold value outside of which an alert will be triggered.
metricThreshold.units	string	The units for the threshold value. Supports the same values as the `currentValue.units` field of the **alerts** resource.
metricThreshold.mode	string	The mode to use when computing the current metric value. Possible values are: • `AVERAGE` • `TOTAL`

Links

Relation	Description
self	Me
group	The group that owns this alert configuration.
alerts	Open alerts triggered by this alert configuration.

Examples

Get All Alert Configurations in a Group

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e/alertConfigs"

HTTP/1.1 200 OK

{
  "totalCount": 3,
  "results": [ {
    "id" : "5271259ee4b00ece6b4754ef",
    "groupId" : "5196d3628d022db4cbc26d9e",
    "typeName" : "BACKUP",
    "eventTypeName" : "RESYNC_REQUIRED",
    "created" : "2013-10-30T15:28:30Z",
    "updated" : "2014-02-12T16:11:05Z",
    "enabled" : true,
    "matchers" : [ ],
    "notifications" : [ {
      "typeName" : "EMAIL",
      "intervalMin" : 60,
      "delayMin" : 0,
      "emailAddress" : "somebody@somewhere.com"
    } ],
```

```
    "links" : [ ... ]
  }, {
    "id" : "5329c8dfe4b0b07a83d67e7e",
    "groupId" : "5196d3628d022db4cbc26d9e",
    "typeName" : "AGENT",
    "eventTypeName" : "MONITORING_AGENT_DOWN",
    "created" : "2014-03-19T16:42:07Z",
    "updated" : "2014-03-19T16:42:07Z",
    "enabled" : true,
    "matchers" : [ ],
    "notifications" : [ {
      "typeName" : "GROUP",
      "intervalMin" : 5,
      "delayMin" : 0,
      "emailEnabled" : true,
      "smsEnabled" : false
    } ],
    "links" : [ ... ]
  }, {
    "id" : "533dc40ae4b00835ff81eaee",
    "groupId" : "5196d3628d022db4cbc26d9e",
    "typeName" : "HOST_METRIC",
    "eventTypeName" : "OUTSIDE_METRIC_THRESHOLD",
    "created" : "2014-04-03T20:26:50Z",
    "updated" : "2014-04-03T20:26:50Z",
    "enabled" : true,
    "matchers" : [ {
      "field" : "hostnameAndPort",
      "operator" : "EQUALS",
      "value" : "mongo.babypearfoo.com:27017"
    } ],
    "notifications" : [ {
      "typeName" : "SMS",
      "intervalMin" : 5,
      "delayMin" : 0,
      "mobileNumber" : "2343454567"
    } ],
    "metricThreshold" : {
      "metricName" : "ASSERT_REGULAR",
      "operator" : "LESS_THAN",
      "threshold" : 99.0,
      "units" : "RAW",
      "mode" : "AVERAGE"
    },
    "links" : [ ... ]
  } ]
}
```

Get an Alert Configuration

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e/alertConfigs/533dc40ae4b00835ff81eaee"

HTTP/1.1 200 OK

{
```

```
  "id" : "533dc40ae4b00835ff81eaee",
  "groupId" : "5196d3628d022db4cbc26d9e",
  "typeName" : "HOST_METRIC",
  "eventTypeName" : "OUTSIDE_METRIC_THRESHOLD",
  "created" : "2014-04-03T20:26:50Z",
  "updated" : "2014-04-03T20:26:50Z",
  "enabled" : true,
  "matchers" : [ {
    "field" : "hostnameAndPort",
    "operator" : "EQUALS",
    "value" : "mongo.babypearfoo.com:27017"
  } ],
  "notifications" : [ {
    "typeName" : "SMS",
    "intervalMin" : 5,
    "delayMin" : 0,
    "mobileNumber" : "2343454567"
  } ],
  "metricThreshold" : {
    "metricName" : "ASSERT_REGULAR",
    "operator" : "LESS_THAN",
    "threshold" : 99.0,
    "units" : "RAW",
    "mode" : "AVERAGE"
  },
  "links" : [ ... ]
}
```

Get All Open Alerts Triggered by an Alert Configuration

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e/alertConfigs/533dc40ae4b00835ff81eaee/alerts"

HTTP/1.1 200 OK

{
  "totalCount" : 2,
  "results" : [ {
    "id" : "53569159300495c7702ee3a3",
    "groupId" : "4d1b6314e528c81a1f200e03",
    "typeName" : "HOST_METRIC",
    "eventTypeName" : "OUTSIDE_METRIC_THRESHOLD",
    "status" : "OPEN",
    "acknowledgedUntil" : "2014-05-01T14:00:00Z",
    "created" : "2014-04-22T15:57:13.562Z",
    "updated" : "2014-04-22T20:14:11.388Z",
    "lastNotified" : "2014-04-22T15:57:24.126Z",
    "metricName" : "ASSERT_REGULAR",
    "currentValue" : {
      "number" : 0.0,
      "units" : "RAW"
    },
    "links" : [ ... ]
  }, {
    "id" : "5356ca0e300495c770333340",
    "groupId" : "4d1b6314e528c81a1f200e03",
```

```
    "typeName" : "HOST_METRIC",
    "eventTypeName" : "OUTSIDE_METRIC_THRESHOLD",
    "status" : "OPEN",
    "created" : "2014-04-22T19:59:10.657Z",
    "updated" : "2014-04-22T20:14:11.388Z",
    "lastNotified" : "2014-04-22T20:14:19.313Z",
    "metricName" : "ASSERT_REGULAR",
    "currentValue" : {
      "number" : 0.0,
      "units" : "RAW"
    },
    "links" : [ ... ]
  } ],
  "links" : [ ... ]
}
```

Create a New Alert Configuration

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST
↪"https://mms.mongodb.com/api/public/v1.0/groups/4d1b6314e528c81a1f200e03/
↪alertConfigs" --data '
{
  "groupId" : "4d1b6314e528c81a1f200e03",
  "typeName" : "REPLICA_SET",
  "eventTypeName" : "RESYNC_REQUIRED",
  "enabled" : true,
  "notifications" : [ {
    "typeName" : "GROUP",
    "intervalMin" : 5,
    "delayMin" : 0,
    "smsEnabled" : false,
    "emailEnabled" : true
  } ]
}'

HTTP/1.1 201 Created
Location: https://mms.mongodb.com/api/public/v1.0/groups/4d1b6314e528c81a1f200e03/
↪alertConfigs/5357ce3e3004d83bd9c7864c

{
  "id" : "5357ce3e3004d83bd9c7864c",
  "groupId" : "4d1b6314e528c81a1f200e03",
  "typeName" : "REPLICA_SET",
  "created" : "2014-04-23T14:29:18Z",
  "updated" : "2014-04-23T14:29:18Z",
  "enabled" : true,
  "matchers" : [ ],
  "notifications" : [ {
    "typeName" : "GROUP",
    "intervalMin" : 5,
    "delayMin" : 0,
    "emailEnabled" : true,
    "smsEnabled" : false
  } ],
  "links" : [ ... ]
}
```

Update an Existing Alert Configuration

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X PUT
↪"https://mms.mongodb.com/api/public/v1.0/groups/4d1b6314e528c81a1f200e03/
↪alertConfigs/5357ce3e3004d83bd9c7864c" --data '
{
  "groupId" : "4d1b6314e528c81a1f200e03",
  "typeName" : "REPLICA_SET",
  "eventTypeName" : "RESYNC_REQUIRED",
  "enabled" : true,
  "matchers" : [ {
    "fieldName" : "REPLICA_SET_NAME",
    "operator" : "EQUALS",
    "value" : "rs1"
  } ],
  "notifications" : [ {
    "typeName" : "EMAIL",
    "emailAddress" : "sos@foo.com",
    "intervalMin" : 60,
    "delayMin" : 5
  }, {
    "typeName" : "GROUP",
    "intervalMin" : 120,
    "delayMin" : 60,
    "smsEnabled" : true,
    "emailEnabled" : false
  } ]
}'

HTTP/1.1 200 OK

{
  "id" : "5357ce3e3004d83bd9c7864c",
  "groupId" : "4d1b6314e528c81a1f200e03",
  "typeName" : "REPLICA_SET",
  "created" : "2014-04-23T14:52:29Z",
  "updated" : "2014-04-23T14:52:29Z",
  "enabled" : true,
  "matchers" : [ {
    "fieldName" : "REPLICA_SET_NAME",
    "operator" : "EQUALS",
    "value" : "rs1"
  } ],
  "notifications" : [ {
    "typeName" : "EMAIL",
    "intervalMin" : 60,
    "delayMin" : 5,
    "emailAddress" : "sos@foo.com"
  }, {
    "typeName" : "GROUP",
    "intervalMin" : 120,
    "delayMin" : 60,
    "emailEnabled" : false,
    "smsEnabled" : true
  } ],
  "links" : [ ... ]
}
```

Disable an Alert Configuration

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X PATCH
↪"https://mms.mongodb.com/api/public/v1.0/groups/4d1b6314e528c81a1f200e03/
↪alertConfigs/5357ce3e3004d83bd9c7864c" --data '
{
  "enabled" : false
}'

HTTP/1.1 200 OK

{
  "id" : "5357ce3e3004d83bd9c7864c",
  "groupId" : "4d1b6314e528c81a1f200e03",
  "typeName" : "REPLICA_SET",
  "created" : "2014-04-23T14:52:29Z",
  "updated" : "2014-04-23T14:56:25Z",
  "enabled" : false,
  "matchers" : [ {
    "fieldName" : "REPLICA_SET_NAME",
    "operator" : "EQUALS",
    "value" : "rs1"
  } ],
  "notifications" : [ {
    "typeName" : "EMAIL",
    "intervalMin" : 60,
    "delayMin" : 5,
    "emailAddress" : "sos@foo.com"
  }, {
    "typeName" : "GROUP",
    "intervalMin" : 120,
    "delayMin" : 60,
    "emailEnabled" : false,
    "smsEnabled" : true
  } ],
  "links" : [ ... ]
}
```

Delete an Alert Configuration

```
curl -i -u "username:apiKey" --digest -X DELETE "https://mms.mongodb.com/api/public/
↪v1.0/groups/4d1b6314e528c81a1f200e03/alertConfigs/5357ce3e3004d83bd9c7864c"

HTTP/1.1 200 OK
```

Backup Configurations

On this page

The resource modification operation `PATCH` only accepts requests from whitelisted IP addressses. You can only modify a backup configuration if the request originates from an IP address on the API user's whitelist.

Operations

- `GET /api/public/v1.0/groups/GROUP-ID/backupConfigs/CLUSTER-ID` - Get a single backup configuration by cluster ID. `CLUSTER-ID` must be the ID of either a replica set or a sharded cluster.
- `GET /api/public/v1.0/groups/GROUP-ID/backupConfigs` - Get all backup configurations for a group.
- `PATCH /api/public/v1.0/groups/GROUP-ID/backupConfigs/CLUSTER-ID` - Request a state change to an existing backup configuration. Note that these changes are generally asynchronous and will result in a response status code of 202 (Accepted). Additionally, if you issue a `GET` request for a backup configuration after a successful `PATCH`, the returned entity may not immediately reflect the update given the asynchronous nature of these state transitions.

  When modifying the `statusName` property, these are the acceptable transitions:

  - `STARTED` - Only valid if the current status is `STOPPED` or `INACTIVE`.
  - `STOPPED` - Only valid if the current status is `STARTED`.
  - `TERMINATING` - Only valid if the current status is `STOPPED`.
  - You cannot change the `statusName` to these values: `INACTIVE PROVISIONING`.

Sample Entity

```
{
  "groupId": "xxx",
  "clusterId": "yyy",
  "statusName": "STARTED",
  "authMechanismName": "MONGODB_CR",
  "username": "johnny5",
  "password": "guess!",
  "sslEnabled": false,
  "syncSource": "PRIMARY",
  "provisioned": true,
  "excludedNamespaces": [ "a", "b", "c.d" ]
}
```

Entity Fields

Name	Type	Description
groupId	string	ID of the group that owns this backup configuration.
clusterId	string	ID of the cluster that this backup configuration is for.
statusName	enum	The current (or desired) status of the backup configuration. Possible values are: • `INACTIVE` • `PROVISIONING` • `STARTED` • `STOPPED` • `TERMINATING`
authMechanismName	enum	The name of the authentication mechanism to use when connecting to the sync source database. Only present when using authentication. Possible values are: • `MONGODB_CR` • `GSSAPI`
username	string	The username to use to connect to the sync source database. Only present when backing up `mongod` instances that require clients to authenticate.
password	string	The password to use to connect to the sync source database. Only present when backup the `mongod` instances that require clients to authenticate. You may only send this field to Ops Manager when updating backup configuration. `GET` request do *not* include this field.
sslEnabled	boolean	Is SSL enabled for the sync source database?
syncSource	string	The `mongod` instance to get backup data from. Possible values are either a specific hostname or one of: `PRIMARY` and `SECONDARY`. This field is only used when updating a backup configuration. It is not returned by a `GET` request.
provisioned	boolean	Reports if Ops Manager has provisioned the resources needed to store a backup. This field is only present when the amount of data to be backed up exceeds a certain threshold.
excludedNamespaces	string array	A list of database names and/or collection names that to omit from the back up. If a string has a dot (e.g. `.`), then it is a fully qualified namespace in the form of `<database>.<collection>`, otherwise strings are database names.

Links

Relation	Description
self	Me
cluster	The cluster that this backup configuration is for.
group	The group that owns this backup configuration.
snapshotSchedule	The snapshot schedule for this backup configuration.

Examples

Get a Single Backup Configuration

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e/backupConfigs/5196e5b0e4b0fca9cc88334a"

HTTP/1.1 200 OK
```

```
{
  "groupId" : "5196d3628d022db4cbc26d9e",
  "clusterId" : "5196e5b0e4b0fca9cc88334a",
  "statusName" : "STARTED",
  "sslEnabled" : false,
  "excludedNamespaces" : [ ],
  "links" : [ ... ]
}
```

Get All Backup Configurations for a Group

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/groups/
↪5196d3628d022db4cbc26d9e/backupConfigs"

HTTP/1.1 200 OK

{
  "totalCount" : 3,
  "results" : [ {
    "groupId" : "5196d3628d022db4cbc26d9e",
    "clusterId" : "5196e5b0e4b0fca9cc88334a",
    "statusName" : "STARTED",
    "sslEnabled" : false,
    "excludedNamespaces" : [ ],
    "links" : [ ... ]
  }, {
    "groupId" : "5196d3628d022db4cbc26d9e",
    "clusterId" : "51a2ac88e4b0371c2dbf46ea",
    "statusName" : "STARTED",
    "sslEnabled" : false,
    "excludedNamespaces" : [ ],
    "links" : [ ... ]
  }, {
    "groupId" : "5196d3628d022db4cbc26d9e",
    "clusterId" : "52d33abee4b0ca49bc6acd6c",
    "statusName" : "STOPPED",
    "sslEnabled" : false,
    "excludedNamespaces" : [ ],
    "links" : [ ... ]
  } ],
  "links" : [ ... ]
}
```

Update a Backup Configuration

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X PATCH
↪"https://mms.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e/
↪backupConfigs/5196e5b0e4b0fca9cc88334a" --data '
{
  "statusName": "STOPPED"
}'

HTTP/1.1 202 Accepted
```

```
{
  "groupId" : "5196d3628d022db4cbc26d9e",
  "clusterId" : "5196e5b0e4b0fca9cc88334a",
  "statusName" : "STOPPED",
  "sslEnabled" : false,
  "excludedNamespaces" : [ ],
  "links" : [ ... ]
}
```

Snapshot Schedule

On this page

This resource allows you to view and configure various properties of snapshot creation and retention for a replica set or cluster. In order to modify this resource, the request must originate from an IP address on the API user's whitelist.

Operations

- `GET /api/public/v1.0/groups/GROUP-ID/backupConfigs/CLUSTER-ID/snapshotSchedule` - Get the snapshot schedule for a cluster. `CLUSTER-ID` must be the ID of either a replica set or a sharded cluster.
- `PATCH /api/public/v1.0/groups/GROUP-ID/backupConfigs/CLUSTER-ID/snapshotSchedule` - Change the parameters of snapshot creation and retention. Any combination of the snapshot schedule's attributes can be modified.

Sample Entity

```
{
  "groupId": "xxx",
  "clusterId": "yyy",
  "snapshotIntervalHours": 6,
  "snapshotRetentionDays": 3,
  "clusterCheckpointIntervalMin": 15,
  "dailySnapshotRetentionDays": 14,
  "weeklySnapshotRetentionWeeks": 6,
  "monthlySnapshotRetentionMonths": 12
}
```

Entity Fields

Name	Type	Description
groupId	string	ID of the group that owns the backup configuration.
clusterId	string	ID of the cluster to which this backup configuration applies.
snapshotIntervalHours	integer	Number of hours between snapshots. Supported values are 6, 8, 12, and 24.
snapshotRetentionDays	integer	Number of days to keep recent snapshots. Supported values are 1 - 5.
clusterCheckpointIntervalMin	integer	Number of minutes between successive cluster checkpoints. This only applies to sharded clusters. This number determines the granularity of point-in-time restores for sharded clusters.
dailySnapshotRetentionDays	integer	Number of days to retain daily snapshots. Supported values are 1 - 365.
weeklySnapshotRetentionWeeks	integer	Number of weeks to retain weekly snapshots. Supported values are 1 - 52.
monthlySnapshotRetentionMonths	integer	Number of months to retain monthly snapshots. Supported values are 1 - 36.

Links

Relation	Description
self	Me
cluster	The cluster that this backup configuration is for.
group	The group that owns this backup configuration.
backupConfig	The backup configuration that this schedule belongs to.

Examples

Get a Snapshot Schedule

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/groups/
↪525ec8394f5e625c80c7404a/backupConfigs/53bc556ce4b049c88baec825/snapshotSchedule"

HTTP/1.1 200 OK

{
  "groupId" : "525ec8394f5e625c80c7404a",
  "clusterId" : "53bc556ce4b049c88baec825",
  "snapshotIntervalHours" : 6,
  "snapshotRetentionDays" : 2,
  "dailySnapshotRetentionDays" : 7,
  "weeklySnapshotRetentionWeeks" : 4,
  "monthlySnapshotRetentionMonths" : 13,
  "links": [ ... ]
}
```

Update a Snapshot Schedule

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X PATCH
↪"https://mms.mongodb.com/api/public/v1.0/groups/525ec8394f5e625c80c7404a/
↪backupConfigs/53bc556ce4b049c88baec825/snapshotSchedule" --data '
{
```

```
  "snapshotIntervalHours": 8,
  "dailySnapshotRetentionDays": 14,
  "monthlySnapshotRetentionMonths": 6
}'

HTTP/1.1 200 OK

{
  "groupId" : "525ec8394f5e625c80c7404a",
  "clusterId" : "53bc556ce4b049c88baec825",
  "snapshotIntervalHours" : 8,
  "snapshotRetentionDays" : 2,
  "dailySnapshotRetentionDays" : 14,
  "weeklySnapshotRetentionWeeks" : 4,
  "monthlySnapshotRetentionMonths" : 6,
  "links": [ ... ]
}
```

Snapshots

On this page

This resource allows you to view snapshot metadata and remove existing snapshots. A snapshot is a complete copy of the data in a `mongod` instance at a point in time. In order to delete a resource, the request must originate from an IP address on the API user's whitelist.

Note that this resource is only meant to provide snapshot metadata. In order to retrieve the snapshot data (in order to perform a restore, for example), you must create a Restore Job.

Operations

- `GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/snapshots` - Get all snapshots for a cluster. `CLUSTER-ID` must be the ID of either a replica set or a sharded cluster.
- `GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/snapshots/SNAPSHOT-ID` - Get a single snapshot.
- `DELETE /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/snapshots/SNAPSHOT-ID` - Remove a single snapshot. Note that while the two above methods return metadata about the snapshot, this will actually remove the underlying backed-up data.

Sample Entity


```
{
  "id": "5875f665da588548965b",
  "groupId": "2847387cd717dabc348a",
  "clusterId": "348938fbdbca74718cba",
  "created": {
    "date": "2014-02-01T12:34:12Z",
    "increment": 54
  },
  "expires": "2014-08-01T12:34:12Z",
  "complete": true,
  "isPossiblyInconsistent": false,
  "missingShards": [
    {
      "id": "472837bcbd278abcd7",
      "groupId": "2847387cd717dabc348a",
      "typeName": "REPLICA_SET",
      "clusterName": "Cluster 1",
      "shardName": "shard002",
      "replicaSetName": "rs3",
      "lastHeartbeat": "2014-02-26T17:32:45Z"
    }
  ],
  "parts": [
    {
      "typeName": "REPLICA_SET",
      "clusterId": "2383294dbcafa82928ad",
      "replicaSetName": "rs0",
      "mongodVersion": "2.4.8",
      "dataSizeBytes": 489283492,
      "storageSizeBytes": 489746352,
      "fileSizeBytes": 518263456
    }, {
      "typeName": "REPLICA_SET",
      "clusterId": "2383294dbcafa82928b3",
      "replicaSetName": "rs1",
      "mongodVersion": "2.4.8",
      "dataSizeBytes": 489283492,
      "storageSizeBytes": 489746352,
      "fileSizeBytes": 518263456
    }, {
      "typeName": "CONFIG_SERVER",
      "mongodVersion": "2.4.6",
      "dataSizeBytes": 48928,
      "storageSizeBytes": 48974,
      "fileSizeBytes": 51826
    }
  ]
}
```

Entity Fields

Name	Type	Description
groupId	string	ID of the group that owns the snapshot.
clusterId	string	ID of the cluster represented by the snapshot.
created	BSON timestamp	The exact point-in-time at which the snapshot was taken.
expires	timestamp	The date after which this snapshot is eligible for deletion.
complete	boolean	Is this snapshot complete? This will be false if the snapshot creation job is still in progress.
isPossiblyInconsistent	boolean	Could this snapshot be inconsistent? `isPossiblyInconsistent` is only present for *sharded cluster* snapshots. In order to take a snapshot of a sharded cluster in a consistent state, the backup agent will temporarily turn off the balancer before creating the snapshot. In some cases, it will not be able to turn off the balancer in a timely manner, so the snapshot will be created with the balancer still running. If this happens, the snapshot may be in an inconsistent state (e.g., because chunk migrations may be in progress).
missingShards	array of clusters	List of shards that are missing from the snapshot. Only present for a sharded cluster snapshot. In steady state, this array will be empty. However, if the backup agent is unable to connect to a shard when a snapshot is created, it will be omitted from the snapshot. Each document in the array is a **cluster** document containing a `self` link.
parts	array of parts	The individual parts that comprise the complete snapshot. For a replica set, this array will contain a single element. For a sharded cluster, there will be one element for each shard plus one element for the config server.
parts.typeName	enum	The type of server represented by the part. Possible values are: • `REPLICA_SET` • `CONFIG_SERVER`
parts.clusterId	string	ID of the replica set. Not present for a config server.
parts.replicaSetName	string	Name of the replica set. Not present for a config server.
parts.mongodVersion	string	The version of `mongod` that was running when the snapshot was created.
parts.dataSizeBytes	integer	The total size of the data in the snapshot.
parts.storageSizeBytes	integer	The total size of space allocated for document storage.
parts.fileSizeBytes	integer	The total size of the data files.

Links

Relation	Description
self	Me
cluster	The cluster that this snapshot belongs to.
group	The group that owns this snapshot.

Examples

Get All Snapshots

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/groups/
↪525ec8394f5e625c80c7404a/clusters/53bc556ce4b049c88baec825/snapshots"

HTTP/1.1 200 OK

{
  "totalCount" : 3,
  "results" : [ {
    "id" : "53bd5fb5e4b0774946a16fad",
    "groupId" : "525ec8394f5e625c80c7404a",
    "clusterId" : "53bc556ce4b049c88baec825",
    "created" : {
      "date" : "2014-07-09T15:24:37Z",
      "increment" : 1
    },
    "expires" : "2014-07-11T15:24:37Z",
    "complete" : true,
    "parts" : [ {
      "typeName" : "REPLICA_SET",
      "clusterId" : "53bc556ce4b049c88baec825",
      "replicaSetName" : "rs0",
      "mongodVersion" : "2.6.3",
      "dataSizeBytes" : 17344,
      "storageSizeBytes" : 10502144,
      "fileSizeBytes" : 67108864
    } ],
    "links" : [ ... ]
  }, {
    ...
  } ],
  "links": [ ... ]
}
```

Get One Snapshot

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/groups/
↪525ec8394f5e625c80c7404a/clusters/53bc556ce4b049c88baec825/snapshots/
↪53bd5fb5e4b0774946a16fad"

HTTP/1.1 200 OK

{
  "id" : "53bd5fb5e4b0774946a16fad",
  "groupId" : "525ec8394f5e625c80c7404a",
  "clusterId" : "53bc556ce4b049c88baec825",
  "created" : {
    "date" : "2014-07-09T15:24:37Z",
    "increment" : 1
  },
  "expires" : "2014-07-11T15:24:37Z",
  "complete" : true,
```

```
  "parts" : [ {
    "typeName" : "REPLICA_SET",
    "clusterId" : "53bc556ce4b049c88baec825",
    "replicaSetName" : "rs0",
    "mongodVersion" : "2.6.3",
    "dataSizeBytes" : 17344,
    "storageSizeBytes" : 10502144,
    "fileSizeBytes" : 67108864
  } ],
  "links" : [ ... ]
}
```

Remove a Snapshot

```
curl -i -u "username:apiKey" --digest -X DELETE "https://mms.mongodb.com/api/public/
↪v1.0/groups/525ec8394f5e625c80c7404a/clusters/53bc556ce4b049c88baec825/snapshots/
↪53bd5fb5e4b0774946a16fad"

HTTP/1.1 200 OK
```

Restore Jobs

On this page

This resource allows you to manage restore jobs. A restore job is essentially a request to retrieve one of your existing snapshots, or a snapshot for a recent specific point-in-time, in order to restore a `mongod` to a previous state. In order to initiate a restore job, the request must originate from an IP address on the API user's whitelist.

Operations

```
GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/restoreJobs
```

- Get all restore jobs for a cluster. `CLUSTER-ID` must be the ID of either a replica set or a sharded cluster.

```
GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/restoreJobs?batchId=BATCH-ID
```

- Get all restore jobs in the specified batch. When creating a restore job for a sharded cluster, Ops Manager creates a separate job for each shard, plus another for the config server. Each of those jobs will be part of a batch. A restore job for a replica set, however, will not be part of a batch.

```
GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/restoreJobs/JOB-ID
```

- Get a single restore job.

```
POST /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/restoreJobs
```

- Create a restore job for the specified `CLUSTER-ID`. You can create a restore job for either an existing snapshot or for a specific recent point-in-time. (The recency depends on the size of your "point-in-time window.") See below for examples of each. The response body includes an array of restore jobs. When requesting a restore of a replica set, the array will contain a single element. For a sharded cluster, the array will contain one element for each shard, plus one for the config server. Each element will also include the `batchId` representing the batch to which the jobs belong.

Sample Entity

```
{
  "id" : "53bd7f13e4b0a7304e226998",
  "groupId" : "525ec8394f5e625c80c7404a",
  "clusterId" : "53bc556ce4b049c88baec825",
  "snapshotId" : "53bd439ae4b0774946a16490",
  "created" : "2014-07-09T17:42:43Z",
  "timestamp" : {
    "date" : "2014-07-09T09:24:37Z",
    "increment" : 1
  },
  "statusName" : "FINISHED",
  "pointInTime" : false,
  "delivery" : {
    "methodName" : "HTTP",
    "url" : "https://api-backup.mongodb.com/backup/restore/v2/pull/
↪ae6bc7a8bfdd5a99a0c118c73845dc75/53bd7f13e4b0a7304e226998/2292652411027442213/
↪525ec8394f5e625c80c7404a-rs0-1404897877.tar.gz",
    "expires" : "2014-07-09T18:42:43Z",
    "statusName" : "READY"
  },
  "hashes" : [
    {
      "typeName": "SHA1",
      "fileName": "filename.0",
      "hash": "5h1DziNbqx9yY2VGJUz5RFnNco0="
    }
  ],
  "links" : [ ... ]
}
```

Entity Fields

Name	Type	Description
groupId	string	ID of the group that owns the restore job.
clusterId	string	ID of the cluster represented by the restore job.
snapshotId	string	ID of the snapshot to restore.
batchId	string	ID of the batch to which this restore job belongs. Only present for a restore of a sharded cluster.
created	timestamp	When the restore job was requested.
timestamp	BSON timestamp	Timestamp of the latest oplog entry in the restored snapshot.
statusName	enum	Current status of the job. Possible values are: • `FINISHED` • `IN_PROGRESS` • `BROKEN` • `KILLED`
pointInTime	boolean	Is this job for a point-in-time restore?
delivery	object	Additional details about how the restored snapshot data will be delivered.
delivery.methodName	enum	How the data will be delivered. Possible values are: • `HTTP`
delivery.url	string	The URL from which the restored snapshot data can be downloaded. Only present if `delivery.methodName` is `HTTP`.
delivery.expires	timestamp	Date after which the URL will no longer be available. Only present if `delivery.methodName` is `HTTP`.
delivery.statusName	enum	Current status of the downloadable file. Only present if `delivery.methodName` is `HTTP`. Possible values are: • `READY` • `EXPIRED` • `MAX_DOWNLOADS_EXCEEDED`
hashes	object array	If the corresponding `delivery.url` has been downloaded, each document in this array is a mapping of a restore file to a hashed checksum. This array is present **only after** the file is downloaded.
hashes.typeName	enum	The hashing algorithm used to compute the hash value. Possible values are: • `SHA1`
hashes.fileName	string	The name of the file that has been hashed.
hashes.hash	string	The hash of the file.

Links

Relation	Description
self	Me
cluster	The cluster to restore.
snapshot	The snapshot to restore.
group	The group that owns the cluster.

Examples

Get All Restore Jobs

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/groups/
↪525ec8394f5e625c80c7404a/clusters/53bc556ce4b049c88baec825/restoreJobs"

HTTP/1.1 200 OK

{
  "totalCount" : 2,
  "results" : [ {
    "id" : "53bd7f38e4b0a7304e226b3f",
    "groupId" : "525ec8394f5e625c80c7404a",
    "clusterId" : "53bc556ce4b049c88baec825",
    "snapshotId" : "53bd4356e4b0774946a16455",
    "created" : "2014-07-09T17:43:20Z",
    "timestamp" : {
      "date" : "2014-07-08T21:24:37Z",
      "increment" : 1
    },
    "statusName" : "FINISHED",
    "pointInTime" : false,
    "delivery" : {
      "methodName" : "HTTP",
      "url" : "https://api-backup.mongodb.com/backup/restore/v2/pull/
↪ae6bc7a8bfdd5a99a0c118c73845dc75/53bd7f38e4b0a7304e226b3f/8924913772648127956/
↪525ec8394f5e625c80c7404a-rs0-1404854677.tar.gz",
      "expires" : "2014-07-09T18:43:21Z",
      "statusName" : "READY"
    },
    "links" : [ ... ]
  }, {
    "id" : "53bd7f13e4b0a7304e226998",
    "groupId" : "525ec8394f5e625c80c7404a",
    "clusterId" : "53bc556ce4b049c88baec825",
    "snapshotId" : "53bd439ae4b0774946a16490",
    "created" : "2014-07-09T17:42:43Z",
    "timestamp" : {
      "date" : "2014-07-09T09:24:37Z",
      "increment" : 1
    },
    "statusName" : "FINISHED",
    "pointInTime" : false,
    "delivery" : {
      "methodName" : "HTTP",
      "url" : "https://api-backup.mongodb.com/backup/restore/v2/pull/
↪ae6bc7a8bfdd5a99a0c118c73845dc75/53bd7f13e4b0a7304e226998/2292652411027442213/
↪525ec8394f5e625c80c7404a-rs0-1404897877.tar.gz",
      "expires" : "2014-07-09T18:42:43Z",
      "statusName" : "READY"
    },
    "links" : [ ... ]
  } ],
  "links": [ ... ]
}
```

Get a Single Restore Job

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/groups/
↪525ec8394f5e625c80c7404a/clusters/53bc556ce4b049c88baec825/restoreJobs/
↪53bd7f13e4b0a7304e226998"

HTTP/1.1 200 OK

{
  "id" : "53bd7f13e4b0a7304e226998",
  "groupId" : "525ec8394f5e625c80c7404a",
  "clusterId" : "53bc556ce4b049c88baec825",
  "snapshotId" : "53bd439ae4b0774946a16490",
  "created" : "2014-07-09T17:42:43Z",
  "timestamp" : {
    "date" : "2014-07-09T09:24:37Z",
    "increment" : 1
  },
  "statusName" : "FINISHED",
  "pointInTime" : false,
  "delivery" : {
    "methodName" : "HTTP",
    "url" : "https://api-backup.mongodb.com/backup/restore/v2/pull/
↪ae6bc7a8bfdd5a99a0c118c73845dc75/53bd7f13e4b0a7304e226998/2292652411027442213/
↪525ec8394f5e625c80c7404a-rs0-1404897877.tar.gz",
    "expires" : "2014-07-09T18:42:43Z",
    "statusName" : "READY"
  },
  "links" : [ ... ]
}
```

Create a Restore Job for an Existing Snapshot

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST
↪"https://mms.mongodb.com/api/public/v1.0/groups/525ec8394f5e625c80c7404a/clusters/
↪53bc556ce4b049c88baec825/restoreJobs" --data '
{
  "snapshotId": "53bd439ae4b0774946a16490"
}'

HTTP/1.1 200 OK

{
  "totalCount" : 1,
  "results" : [ {
    "id" : "53bd9f9be4b0a7304e23a8c6",
    "groupId" : "525ec8394f5e625c80c7404a",
    "clusterId" : "53bc556ce4b049c88baec825",
    "snapshotId" : "53bd439ae4b0774946a16490",
    "created" : "2014-07-09T20:01:31Z",
    "timestamp" : {
      "date" : "2014-07-09T09:24:37Z",
      "increment" : 1
    },
    "statusName" : "IN_PROGRESS",
    "pointInTime" : false,
```

```
    "links" : [ ... ]
  } ]
}
```

Create a Point-In-Time Restore Job

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST
↪"https://mms.mongodb.com/api/public/v1.0/groups/525ec8394f5e625c80c7404a/clusters/
↪53bc556ce4b049c88baec825/restoreJobs" --data '
{
  "timestamp": {
    "date": "2014-07-09T09:20:00Z",
    "increment": 0
  }
}'

HTTP/1.1 200 OK

{
  "totalCount" : 1,
  "results" : [ {
    "id" : "53bda0dfe4b0a7304e23b54a",
    "groupId" : "525ec8394f5e625c80c7404a",
    "clusterId" : "53bc556ce4b049c88baec825",
    "created" : "2014-07-09T20:06:55Z",
    "timestamp" : {
      "date" : "2014-07-09T09:20:00Z",
      "increment" : 0
    },
    "statusName" : "IN_PROGRESS",
    "pointInTime" : true,
    "links" : [ ... ]
  } ]
}
```

Whitelist

On this page

The resource modification operations `POST` and `DELETE` are whitelisted. For example, you can only add an IP address to a whitelist if the request originates from an IP address on the existing whitelist.

Operations

- `GET /api/public/v1.0/users/USER-ID/whitelist` - Gets the whitelist for the specified user. You can only access your own whitelist, so the `USER-ID` in the URL *must* match the ID of the user associated with the API Key.
- `GET /api/public/v1.0/users/USER-ID/whitelist/IP-ADDRESS` - Gets the whitelist entry for a single IP address.
- `POST /api/public/v1.0/users/USER-ID/whitelist` - Add one or more IP addresses to the user's whitelist.
  - The entity body must be an array of whitelist entities, even if there is only one. The only field you need to specify for each entity is the `ipAddress`.
  - If an IP address is already in the whitelist, it will be ignored.
- `DELETE /api/public/v1.0/users/USER-ID/whitelist/IP-ADDRESS` - Remove an IP address from the whitelist.
  - You cannot remove your current IP address from the whitelist.

Sample Entity

```
{
  "ipAddress": "1.2.3.4",
  "created": "2014-01-02T12:34:56Z",
  "lastUsed": "2014-03-12T02:03:04Z",
  "count": 1234
}
```

Entity Fields

Name	Type	Description
ipAddress	string	A whitelisted IP address.
created	date	The date this IP address was added to the whitelist.
lastUsed	date	The date of the most recent request that originated from this IP address. Note that this field is *only* updated when a resource that is protected by the whitelist is accessed.
count	integer	The total number of requests that originated from this IP address. Note that this field is *only* updated when a resource that is protected by the whitelist is accessed.

Links

Relation	Description
self	Me
user	The user that owns this whitelist.

Examples

Get a User's Whitelist

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/users/
↪5356823b3004dee37132bb7b/whitelist"

HTTP/1.1 200 OK

{
  "totalCount" : 1,
  "results" : [ {
    "ipAddress" : "12.34.56.78",
    "created" : "2014-04-23T16:17:44Z",
    "count" : 482,
    "links" : [ ... ]
  } ],
  "links" : [ ... ]
}
```

Get a Single Whitelist Entry

```
curl -i -u "username:apiKey" --digest "https://mms.mongodb.com/api/public/v1.0/users/
↪5356823b3004dee37132bb7b/whitelist/12.34.56.78"

HTTP/1.1 200 OK

{
  "ipAddress" : "12.34.56.78",
  "created" : "2014-04-23T16:17:44Z",
  "count" : 482,
  "links" : [ ... ]
}
```

Add Entries to a User's Whitelist

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST
↪"https://mms.mongodb.com/api/public/v1.0/users/5356823b3004dee37132bb7b/whitelist" -
↪-data '
[ {
    "ipAddress" : "76.54.32.10"
  }, {
    "ipAddress" : "2.3.4.5"
} ]'

HTTP/1.1 201 Created
Location: https://mms.mongodb.com/api/public/v1.0/users/5356823b3004dee37132bb7b/
↪whitelist

{
  "totalCount" : 3,
  "results" : [ {
    "ipAddress" : "12.34.56.78",
```

```
    "created" : "2014-04-23T16:17:44Z",
    "count" : 0,
    "links" : [ ... ]
  }, {
    "ipAddress" : "76.54.32.10",
    "created" : "2014-04-23T16:23:44Z",
    "count" : 0,
    "links" : [ ... ]
  }, {
    "ipAddress" : "2.3.4.5",
    "created" : "2014-04-23T16:23:44Z",
    "count" : 0,
    "links" : [ ... ]
  } ],
  "links" : [ ... ]
}
```

Delete an Entry from a User's Whitelist

```
curl -i -u "username:apiKey" --digest -X DELETE "https://mms.mongodb.com/api/public/
↪v1.0/users/5356823b3004dee37132bb7b/whitelist/2.3.4.5"

HTTP/1.1 200 OK
```

Automation Configuration

On this page

Overview

The Public API provides the `automationConfig` endpoint to let you manipulate a group's *automation configuration*. The processes you specify in the configuration, define the deployment that your *Automation Agents* will attempt to build. These processes can include clusters, replica sets and standalones, as well as Backup and Monitoring Agents.

Each Automation Agent resides on its own host and is responsible for the processes the configuration defines for that host. The agent runs the processes as defined by the configuration. When a running process matches its defined configuration, the process is in "goal state." When all processes on all hosts are in goal state, the deployment itself is in goal state.

Operations

- `GET /api/public/v1.0/groups/GROUP-ID/automationConfig`

  Retrieve the current automation configuration for a group.

- `PUT /api/public/v1.0/groups/GROUP-ID/automationConfig`

  Update a group's automation configuration. For steps for updating an automation configuration, see *Deploy a Cluster through the API*.

Updates

To update the configuration through the `automationConfig` endpoint, send an object that *contains a specific subset of the configuration's fields*. This is the same subset as you receive when you retrieve the configuration through the endpoint.

When you submit updates, Ops Manager makes internal modifications to the data and then saves your new configuration version. For example, Ops Manager might add a field to each specified community MongoDB version to indicate where the agents download them from.

The Automation Agents continuously poll Ops Manager for changes to the configuration and fetch configuration updates when they occur. The agents then adjust the states of their live processes to match. For a tutorial on updating a deployment using the API, see *Deploy a Cluster through the API*.

Concurrent Modifications Warning

There is **no protection** in the Public API to prevent concurrent modifications. If two administrators both start with a configuration based on the current version, make their own modifications, and then submit their modifications, the later modification wins.

Other Representations of the Automation Configuration

You can view the internal representation of the configuration through the *Raw AutomationConfig* page on the *Deployment* tab of the Ops Manager interface. The raw configuration shows the internal representation of the configuration that Ops Manager stores, including fields that should **not** be updated through the API.

The Automation Agent also stores a copy of the configuration in the `mms-cluster-config-backup.json` file. The agent stores the most recent version of configuration **with which the agent was able to reach goal state**. If an agent is not able to process configuration changes, it continues to store an older version of the configuration.

Sample Automation Configuration Entity

```
{
    "monitoringVersions": [
        {
            "hostname": "one.example.net",
            "logPath": "/var/log/mongodb-mms-automation/monitoring-agent.log",
            "logRotate": {
                "sizeThresholdMB": 1000,
                "timeThresholdHrs": 24
            }
        }
```

```
    ],
    "backupVersions": [
        {
            "hostname": "one.example.net",
            "logPath": "/var/log/mongodb-mms-automation/backup-agent.log",
            "logRotate": {
                "sizeThresholdMB": 1000,
                "timeThresholdHrs": 24
            }
        }
    ],
    "processes": [
        {
            "name": "MyCLUSTER_MySHARD_0_0",
            "processType": "mongod",
            "version": "2.6.7",
            "hostname": "testAutoAPI-0.dns.placeholder",
            "logRotate": {
                "sizeThresholdMB": 1000,
                "timeThresholdHrs": 24
            },
            "authSchemaVersion": 1,
            "args2_6": {
                "net": {
                    "port": 27017
                },
                "storage": {
                    "dbPath": "/data/MyCLUSTER_MySHARD_0_0"
                },
                "systemLog": {
                    "path": "/data/MyCLUSTER_MySHARD_0_0/mongodb.log",
                    "destination": "file"
                },
                "replication": {
                    "replSetName": "MySHARD_0"
                },
                "operationProfiling": {}
            }
        },

        ...,

        {
            "name": "MyCLUSTER_MyCONFIG_SERVER_8",
            "processType": "mongod",
            "version": "2.6.7",
            "hostname": "SERVER-8",
            "logRotate": {
                "sizeThresholdMB": 1000,
                "timeThresholdHrs": 24
            },
            "authSchemaVersion": 1,
            "args2_6": {
                "net": {
                    "port": 27019
                },
                "storage": {
                    "dbPath": "/data/MyCLUSTER_MyCONFIG_SERVER_8"
```

```
                },
                "systemLog": {
                    "path": "/data/MyCLUSTER_MyCONFIG_SERVER_8/mongodb.log",
                    "destination": "file"
                },
                "sharding": {
                    "clusterRole": "configsvr"
                },
                "operationProfiling": {}
            }
        },
        {
            "name": "MyCLUSTER_MyMONGOS_9",
            "processType": "mongos",
            "version": "2.6.7",
            "hostname": "SERVER-9",
            "cluster": "myShardedCluster",
            "logRotate": {
                "sizeThresholdMB": 1000,
                "timeThresholdHrs": 24
            },
            "authSchemaVersion": 1,
            "args2_6": {
                "net": {
                    "port": 27017
                },
                "systemLog": {
                    "path": "/data/MyCLUSTER_MyMONGOS_9/mongodb.log",
                    "destination": "file"
                },
                "operationProfiling": {}
            }
        }
    ],
    "replicaSets": [
        {
            "_id": "MySHARD_0",
            "members": [
                {
                    "_id": 0,
                    "host": "MyCLUSTER_MySHARD_0_0",
                    "priority": 1,
                    "votes": 1,
                    "slaveDelay": 0,
                    "hidden": false,
                    "arbiterOnly": false
                },
                {
                    "_id": 1,
                    "host": "MyCLUSTER_MySHARD_0_1",
                    "priority": 1,
                    "votes": 1,
                    "slaveDelay": 0,
                    "hidden": false,
                    "arbiterOnly": false
                },
                {
                    "_id": 2,
```

```
                    "host": "MyCLUSTER_MySHARD_0_2",
                    "priority": 1,
                    "votes": 1,
                    "slaveDelay": 0,
                    "hidden": false,
                    "arbiterOnly": false
                }
            ]
        },
        {
            "_id": "MySHARD_1",
            "members": [
                {
                    "_id": 0,
                    "host": "MyCLUSTER_MySHARD_1_3",
                    "priority": 1,
                    "votes": 1,
                    "slaveDelay": 0,
                    "hidden": false,
                    "arbiterOnly": false
                },
                {
                    "_id": 1,
                    "host": "MyCLUSTER_MySHARD_1_4",
                    "priority": 1,
                    "votes": 1,
                    "slaveDelay": 0,
                    "hidden": false,
                    "arbiterOnly": false
                },
                {
                    "_id": 2,
                    "host": "MyCLUSTER_MySHARD_1_5",
                    "priority": 1,
                    "votes": 1,
                    "slaveDelay": 0,
                    "hidden": false,
                    "arbiterOnly": false
                }
            ]
        }
    ],
    "sharding": [
        {
            "name": "myShardedCluster",
            "configServer": [
                "MyCLUSTER_MyCONFIG_SERVER_6",
                "MyCLUSTER_MyCONFIG_SERVER_7",
                "MyCLUSTER_MyCONFIG_SERVER_8"
            ],
            "shards": [
                {
                    "_id": "MySHARD_0",
                    "rs": "MySHARD_0"
                },
                {
                    "_id": "MySHARD_1",
                    "rs": "MySHARD_1"
```

```
                }
            ],
            "collections": [],
        }
    ],
    "mongoDbVersions": [
        {"name": "2.4.12"},
        {"name": "2.6.7"}
    ]
}
```

Entity Fields

Name	Type	Description
monitoringVersions	array	*Optional*. Objects that define version information for each Monitoring Agent.
- hostname	string	The hostname of the machine that runs the Monitoring Agent. If the Monitoring Agent is not running on the machine, Ops Manager installs it.
- logPath	string	*Optional*. The directory where the agent stores its logs. The default is to store logs in `/dev/null`.
- logRotate	object	*Optional*. Enables log rotation for the MongoDB logs for a process.
- - sizeThresholdMB	number (integer or float)	The maximum size in MB for an individual log file before rotation.
- - timeThresholdHrs	integer	The maximum time in hours for an individual log file before rotation.
backupVersions	array	*Optional*. Objects that define version information for each Backup Agent.
- hostname	string	The hostname of the machine that runs the Backup Agent. If the Backup Agent is not running on the machine, Ops Manager installs it.
- logPath	string	*Optional*. The directory where the agent stores its logs. The default is to store logs in `/dev/null`.
- logRotate	object	*Optional*. Enables log rotation for the MongoDB logs for a process.
- - sizeThresholdMB	number (integer or float)	The maximum size in MB for an individual log file before rotation.
- - timeThresholdHrs	integer	The maximum time in hours for an individual log file before rotation.
processes	array	The `processes` array contains objects that define the `mongos` and `mongod` instances that Ops Manager manages. Each object defines a different instance.
- name	string	A unique name to identify the instance.
- processType	string	Either `mongod` or `mongos`.
- version	string	The name of the `mongoDbVersions` specification used with this instance.
- hostname	string	*Optional*. The name of the host this process should run on. This defaults to `localhost`.
- cluster	string	*Optional*. Required for a `mongos`. The name of the cluster. This must correspond to the `sharding.name` field in the `sharding` array for the `mongos`.
- logRotate	object	*Optional*. Enables log rotation for the MongoDB logs for a process.

Continued on next page

Table 3 – continued from previous page

Name	Type	Description
- - sizeThresholdMB	number (integer or float)	The maximum size in MB for an individual log file before rotation.
- - timeThresholdHrs	integer	The maximum time in hours for an individual log file before rotation.
- authSchemaVersion	integer	*Required if auth is turned on. Otherwise optional*. The schema version of the user credential documents. This should match all other elements of the `processes` array that belong to the same cluster. The possible values are `1`, `3`, and `5`. The default is `3` for `2.6` clusters and `1` for `2.4` clusters.
- <args>	object	This field is named either `args2_6`, for MongoDB versions 2.6 and higher (including 3.0 and higher), or `args2_4`, for versions 2.4 and earlier. The field contains a MongoDB configuration document in the format appropriate to the version. For information on format and supported MongoDB options, see *supported configuration options*.
replicaSets	array	*Optional*. Objects that define the configuration of each replica set. The Automation Agent uses the values in this array to create valid replica set configuration documents. The agent regularly checks that replica sets are configured correctly. If a problem occurs, the agent reconfigures the replica set according to its configuration document. The array can contain the following fields from a replica set configuration document: `_id`; `version`; and `members`. The `members.host` field must specify the host's name as listed in `processes.name`. The Automation Agent expands the `host` field to create a valid replica set configuration.
sharding	array	*Optional*. Objects that define the configuration of each sharded cluster. Each object in the array contains the specifications for one cluster. The Automation Agent regularly checks each cluster's state against the specifications. If the specification and cluster don't match, the agent will change the configuration of the cluster, which might cause the balancer to migrate chunks.
- name	string	The name of the cluster. This must correspond with the value in `processes.cluster` for a `mongos`.
- configServer	array	String values that provide the names of each config server's hosts. The host names are the same names as are used in each host's `processes.name` field.
- shards	array	Objects that define the cluster's shards.
- - _id	string	The name of the shard.
- - rs	string	The name of the shard's replica set, as specified in the `replicaSets._id` field.
- collections	array	Objects that define the sharded collections and their shard keys.
mongoDbVersions	array	The `mongoDbVersions` array is required and defines versions of MongoDB used by the MongoDB instances.
- name	string	The MongoDB version.

Examples

For configuration examples, please see the following page on GitHub: https://github.com/10gen-labs/mms-api-examples/tree/master/automation/.

Automation Status

On this page

Overview

The Public API provides the `automationStatus` endpoint to let you see whether each MongoDB process is up-to-date with the current *automation configuration*. The endpoint returns the `goalVersion` field to report the current version of the automation configuration and the `lastGoalVersionAchieved` fields to report the versions of the configuration running on each server.

Operation

- `GET /api/public/v1.0/groups/GROUP-ID/automationStatus`

  Retrieve the status of each MongoDB process in the deployment.

Sample Entity

```
"goalVersion": 29,
"processes": [
    {
        "hostname": "AGENT_HOST_0",
        "name": "BLUE_0",
        "lastGoalVersionAchieved": 28,
        "plan": ["Download", "Start", "WaitRsInit"]
    },
    {
        "hostname": "AGENT_HOST_1",
        "name": "BLUE_1",
        "lastGoalVersionAchieved": 29,
        "plan": []
    }
]
```

Entity Fields

Name	Type	Description
goalVersion	integer	The version of the most recently submitted *automation configuration*. If there is a *conflict in submissions of automation configurations*, this field lists the winning configuration.
processes	array	The group's deployed MongoDB instances.
- hostname	string	The fully qualified domain name (retrieved by issuing `hostname -f`) of the server on which the MongoDB process and Automation Agent are hosted.
- name	string	The process name as specified in the automation configuration.
- lastGoalVersion-Achieved	integer	The last version of the automation configuration with which this process had deployed as configured. If the `lastGoalVersionAchieved` number is not equal to the `goalVersion` number, the process has not yet deployed according to the current configuration.
- plan	array	Describes how a process that is not yet up-to-date with the configuration will achieve the goal state.

Example

Retrieve status:

```
curl -u "username:apiKey" --digest -i "https://mms.mongodb.com/api/public/v1.0/groups/
↪533c5895b91030606f21033a/automationStatus"
```

Response:


```
{
  "processes": [
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_myShard_0_0",
      "hostname": "testDeploy-0"
    },
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_myShard_0_1",
      "hostname": "testDeploy-1"
    },
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_myShard_0_2",
      "hostname": "testDeploy-2"
    },
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_myShard_1_3",
      "hostname": "testDeploy-3"
    },
    {
      "plan": [],
```

```
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_myShard_1_4",
      "hostname": "testDeploy-4"
    },
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_myShard_1_5",
      "hostname": "testDeploy-5"
    },
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_config_6",
      "hostname": "testDeploy-6"
    },
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_config_7",
      "hostname": "testDeploy-7"
    },
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_config_8",
      "hostname": "testDeploy-8"
    },
    {
      "plan": [],
      "lastGoalVersionAchieved": 2,
      "name": "shardedCluster_mongos_9",
      "hostname": "testDeploy-9"
    }
  ],
  "goalVersion": 2
}
```


9.3 Public API Tutorials

Enable the Public API Enable the Public API for group.

Deploy a Sharded Cluster Create a cluster using the Public API.

Upgrade a Deployment's MongoDB Version Upgrade the version of MongoDB used by the deployment's instances.

Enable the Public API

On this page

Overview

You enable the Public API on a per-group basis. To enable the Public API for a group, you must have the *Owner* or *Global Owner* role.

Considerations

You can have up to ten keys associated to your account. Each key can be either enabled or disabled, but be aware that they both count towards the ten key limit.

API Keys are associated to a user and therefore have the same level of access as that user.

Procedure

Step 1: Enable the Public API for each group.

The Public API is enabled on a per-group basis, so make sure to enable it for all groups that need to use it.

To enable the Public API for a group, select the *Administration* tab and then select *Group Settings*. Click the *Enable Public API* button to toggle it *ON*.

Step 2: Generate an API key.

An API Key is like a password. Keep it secret.

To generate a key, select the *Administration* tab and then *API Keys & Whitelists*. In the *API Keys* section, use the *Generate* button to generate a new key. Follow the prompts, being sure to **copy the key once it is generated**. Ops Manager displays the full key *one time only*. Ops Manager displays only the partial key in the key list and provides no access to the full key.

Step 3: Define your whitelist.

Certain API operations require a higher level of security and are protected by a whitelist. Only client requests that originate from a whitelisted IP address will be permitted to invoke such operations.

To define your whitelist, select the *Administration* tab and then *API Keys & Whitelists*. In the *API Whitelist* section, use the *Add* button to add a permitted IP address.

To delete an address, select the gear icon for the address and select *Delete*.

CIDR notation is not supported.

Additional Information

See the *Public API Principles* for a background on the use and operation of the Public API, and *Public API Resources* for a complete reference of all resources available in the Public API.

Deploy a Cluster through the API

On this page

Overview

This tutorial manipulates the *Public API's* automation configuration to deploy a sharded cluster that is owned by another user. The tutorial first creates a new group, then a new user as owner of the group, and then a sharded cluster owned by the new user. You can create a script to automate these procedures for use in routine operations.

To perform these steps, you must have access to Ops Manager as a user with the *Global Owner* role.

The procedures install a cluster with two shards. Each shard comprises a three-member replica set. The tutorial installs one `mongos` and three config servers. Each component of the cluster resides on its own server, requiring a total of 10 servers.

The tutorial installs the *Automation Agent* on each server.

Prerequisites

Ops Manager must have an existing user with *Global Owner* role. The first user you create has this role. Global owners can perform any Ops Manager action, both through the Ops Manager interface and through the API.

You must have the URL of the Ops Manager Web Server, as set in the `mmsBaseUrl` setting of the *Monitoring Agent configuration file*.

Provision ten servers to host the components of the sharded cluster. For server requirements, see the Production Notes in the MongoDB manual.

Each server must provide its *Automation Agent* with full networking access to the hostnames and ports of the Automation Agents on all the other servers. Each agent runs the command `hostname -f` to self-identify its hostname and port and report them to Ops Manager.

Tip

To ensure agents can reach each other, *provision the servers using Automation*. This installs the Automation Agents with correct network access. Then use this tutorial to reinstall the Automation Agents on those machines.

Examples

As you work with the API, you can view examples on the following GitHub page: https://github.com/10gen-labs/mms-api-examples/tree/master/automation/.

Procedures

Retrieve API Key

This procedure displays the full API key **just once**. You must record the API key when it is displayed.

Note that this API key for the Public API is different from the API key for a group, which is always visible in Ops Manager through the *Group Settings* tab.

Step 1: Log in as a Global Owner.

Log into the Ops Manager web interface as a user with the *Global Owner* role.

Step 2: Select the *Administration* tab and then *API Keys & Whitelists*.

Step 3: Generate a new API key.

In the *API Keys* section, click *Generate*. Then enter a description, such as "API Testing," and click *Generate*.

If prompted for a two-factor verification code, enter the code and click *Verify*. Then click *Generate* again.

Step 4: Copy and record the key.

Copy the key **immediately when it is generated**. Ops Manager displays the full key one time only. You will not be able to view the full key again.

Record the key in a secure place. After you have successfully recorded the key, click *Close*.

Create the Group and the User through the API

Step 1: Use the API to create a group.

Use the Public API to send a *groups* document to create the new group. Issue the following command, replacing `<user@example.net>` with the credentials of the *global owner*, `<api_key>` with your API key, `<app-example.net>` with the Ops Manager URL, and `<group_name>` with the name of the new group:

```
curl -u "<user@example.net>:<api_key>" -H "Content-Type: application/json" "http://
↪<app-example.net>/api/public/v1.0/groups" --digest -i -X POST --data '
{
   "name": "<group_name>"
}'
```

The API returns a document that includes the group's `agentApiKey` and `id`. The API automatically sets the `publicApiEnabled` field to `true` to allow subsequent API-based configuration.

Step 2: Record the values of `agentApiKey` and `id` in the returned document.

Record these values for use in this procedure and in other procedures in this tutorial.

Step 3: Use the API to create a user in the new group.

Use the `/users` endpoint to add a user to the new group.

The body of the request should contain a *users* document with the user's information. Set the user's `roles.roleName` to `GROUP_OWNER`, and the user's `roles.groupId` set to the new group's‘ `id`.

```
curl -u "<user@example.net>:<api_key>" -H "Content-Type: application/json" "http://
↪<app-example.net>/api/public/v1.0/users" --digest -i -X POST --data '
{
   "username": "<new_user@example.com>",
   "emailAddress": "<new_user@example.com>",
   "firstName": "<First>",
   "lastName": "<Last>",
   "password": "<password>",
   "roles": [{
     "groupId": "<group_id>",
     "roleName": "GROUP_OWNER"
   }]
}'
```

Step 4: Remove global owner from the group. (Optional)

The *global owner* that you used to create the group is also automatically added to the group. You can remove the global owner from the group without losing the ability to make changes to the group in the future. As long as you have the group's `agentApiKey` and `id`, you have full access to the group when logged in as the global owner.

`GET` the global owner's ID. Issue the following command to request the group's users, replace the credentials, API key, URL, and group ID, with the relevant values:

```
curl -u "<user@example.net>:<api_key>" "http://<app-example.net>/api/public/v1.0/
↪groups/<group_id>/users" --digest -i
```

The API returns a document that lists all the group's users. Locate the user with `roles.roleName` set to `GLOBAL_OWNER`. Copy the user's `id` value, and issue the following to remove the user from the group, replacing `<user_id>` with the user's `id` value:

```
curl -u "<user@example.net>:<api_key>" "http://<app-example.net>/api/public/v1.0/
↪groups/<group_id>/users/<user_id>" --digest -i -X DELETE
```

Upon successful removal of the user, the API returns the HTTP `200 OK` status code to indicate the request has succeeded.

Install the Automation Agent on each Provisioned Server

Your servers must have the networking access described in the *Prerequisites*.

Step 1: Create the Automation Agent configuration file to be used on the servers.

Create the following configuration file and enter values as shown below. The file uses your `agentApiKey`, group `id`, and the Ops Manager URL.

Save this file as `automation-agent.config`. You will distribute this file to each of your provisioned servers.

```
# REQUIRED
# Enter your Group ID - It can be found at /settings
#
mmsGroupId=<Enter_the_value_you_retrieved_for_group_``id``>

# REQUIRED
# Enter your API key - It can be found at /settings
#
mmsApiKey=<Enter_the_value_you_retrieved_for_``agentApiKey``>

# Base url of the MMS web server.
#
mmsBaseUrl=<Enter_the_URL_of_the_|application|>

# Path to log file
#
logFile=/var/log/mongodb-mms-automation/automation-agent.log

# Path to backup automation configuration
#
mmsConfigBackup=/var/lib/mongodb-mms-automation/mms-cluster-config-backup.json

# Lowest log level to log.  Can be (in order): DEBUG, ROUTINE, INFO, WARN, ERROR, DOOM
#
logLevel=INFO

# Maximum number of rotated log files
#
maxLogFiles=10

# Maximum size in bytes of a log file (before rotating)
#
maxLogFileSize=268435456

# URL to proxy all HTTP requests through
#
#httpProxy=
```

Step 2: Retrieve the command strings used to download and install the Automation Agent.

In the Ops Manager web interface, select the *Administration* tab and then select the *Agents* page. Under *Automation* at the bottom of the page, select your operating system to display the install instructions. Copy and save the following strings from these instructions:

- The `curl` string used to download the agent.
- The `rpm` or `dpkg` string to install the agent. For operating systems that use `tar` to unpackage the agent, no install string is listed.
- The `nohup` string used run the agent.

Step 3: Download, configure, and run the Automation Agent on each server.

Do the following on each of the provisioned servers. You can create a script to use as a turn-key operation for these steps:

Use the `curl` string to download the Automation Agent.

Use `rpm`, `dpkg`, or `tar` to install the agent. Make the agent controllable by the new user you added to the group in the previous procedure.

Replace the *contents* of the config file with the file you created in the first step. The config file is one of the following, depending on the operating system:

- `/etc/mongodb-mms/automation-agent.config`
- `<install_directory>/local.config`

Check that the following directories exist and are accessible to the Automation Agent. If they do not, create them. The first two are created automatically on RHEL, CentOS, SUSE, Amazon Linux, and Ubuntu:

- `/var/lib/mongodb-mms-automation`
- `/var/log/mongodb-mms-automation`
- `/data`

Use the `nohup` string to run the Automation Agent.

Step 4: Confirm the initial state of the automation configuration.

When the Automation Agent first runs, it downloads the `mms-cluster-config-backup.json` file, which describes the desired state of the *automation configuration*.

On one of the servers, navigate to `/var/lib/mongodb-mms-automation/` and open `mms-cluster-config-backup.json`. Confirm that the file's `version` field is set to `1`. Ops Manager automatically increments this field as changes occur.

Deploy the New Cluster

To add or update a deployment, retrieve the *configuration document*, make changes as needed, and send the updated configuration though the API to Ops Manager.

Tip

You can learn more about the configuration file by viewing it in Ops Manager. Select the *Deployment* tab and then the *Raw AutomationConfig* page. Note that the raw configuration contains fields you should not update with the *configuration document*.

The following procedure deploys an updated automation configuration through the Public API:

Step 1: Retrieve the automation configuration from Ops Manager.

Use the *automationConfig* resource to retrieve the configuration. Issue the following command, replacing `<user@example.net>` with the credentials of the *global owner*, `<api_key>` with the previously retrieved API key, `<app-example.net>` with the URL of Ops Manager, and `<group_id>` with the previously retrieved group ID:

```
curl -u "<user@example.net>:<api_key>" "http://<app-example.net>/api/public/v1.0/
↪groups/<group_id>/automationConfig" --digest -i
```

Confirm that the `version` field of the retrieved *automation configuration* matches the `version` field in the `mms-cluster-config-backup.json` file.

Step 2: Create the top level of the new configuration document.

Create a document with the following fields. As you build the configuration document, refer the *description of an automation configuration* for detailed explanations of the settings. For examples, refer to the following page on GitHub: https://github.com/10gen-labs/mms-api-examples/tree/master/automation/.

```
{
    "options": {
        "downloadBase": "/var/lib/mongodb-mms-automation"
    },
    "mongoDbVersions": [],
    "monitoringVersions": [],
    "backupVersions": [],
    "processes": [],
    "replicaSets": [],
    "sharding": []
}
```

Step 3: Add MongoDB versions to the configuration document.

In the `mongoDbVersions` array, add the versions of MongoDB to have available to the deployment. Add only those versions you will use. For this tutorial, the following array includes just one version, `2.4.12`, but you can specify multiple versions. Using `2.4.12` allows this deployment to later upgrade to `2.6`, as described in *Update the MongoDB Version of a Deployment*.

```
"mongoDbVersions": [
    { "name": "2.4.12" }
]
```

Step 4: Add the Monitoring Agent to the configuration document.

In the `monitoringVersions.hostname` field, enter the hostname of the server where Ops Manager should install the Monitoring Agent. Use the fully qualified domain name that running `hostname -f` on the server returns, as in the following:

```
"monitoringVersions": [
    {
        "hostname": "<server_x.example.net>",
        "logPath": "/var/log/mongodb-mms-automation/monitoring-agent.log",
        "logRotate": {
            "sizeThresholdMB": 1000,
            "timeThresholdHrs": 24
        }
    }
]
```

This configuration example also includes the `logPath` field, which specifies the log location, and `logRotate`, which specifies the log thresholds.

Step 5: Add the servers to the configuration document.

This sharded cluster has 10 MongoDB instances, as described in the *Overview*, each running on its own server. Thus, the automation configuration's `processes` array will have 10 documents, one for each MongoDB instance.

The following example adds the first document to the `processes` array. Replace `<process_name_1>` with any name you choose, and replace `<server_1.example.net>` with the FQDN of the server. You will need to add 9 documents: one for each MongoDB instance in your sharded cluster.

The example uses the `args2_4` syntax for the `processes.<args>` field. For MongoDB versions 2.6 and later, use the `args2_6` syntax. See *Supported MongoDB Options for Automation* for more information.

```
"processes": [
    {
        "version": "2.4.12",
        "name": "<process_name_1>",
        "hostname": "<server_1.example.net>",
        "logRotate": {
            "sizeThresholdMB": 1000,
            "timeThresholdHrs": 24
        },
        "authSchemaVersion": 1,
        "processType": "mongod",
        "args2_4": {
            "port": 27017,
            "replSet": "rs1",
            "dbpath": "/data/",
            "logpath": "/data/mongodb.log"
        }
    },
    ...
]
```

Step 6: Add the sharded cluster topology to the configuration document.

Add two replica set documents to the `replicaSets` array. Add three members to each document. The following example shows one replica set member added in the first replica set document:

```
"replicaSets": [
    {
        "_id": "rs1",
        "members": [
            {
                "_id": 0,
                "host": "<process_name_1>",
                "priority": 1,
                "votes": 1,
                "slaveDelay": 0,
                "hidden": false,
                "arbiterOnly": false
            },
            ...
        ]
    },
    ...
```

In the `sharding` array, add the replica sets to the shards, and add the three config servers, as in the following:

```
"sharding": [
    {
        "shards": [
            {
                "tags": [],
                "_id": "shard1",
                "rs": "rs1"
            },
            {
                "tags": [],
                "_id": "shard2",
                "rs": "rs2"
            }
        ],
        "name": "sharded_cluster_via_api",
        "configServer": [
            "<process_name_7>",
            "<process_name_8>",
            "<process_name_9>"
        ],
        "collections": []
    }
]
```

Step 7: Send the configuration document.

Use the `groups/<group_id>/automationConfig` endpoint to to send the automation configuration document to Ops Manager, as in the following. Replace `<user@example.net>` with the credentials of the *global owner*, `<api_key>` with previously retrieved API key, `<app-example.net>` with the Ops Manager URL, and `<group_id>` with the previously retrieved group `id`.

Replace `<configuration_document>` with the configuration document you have created in the previous steps.

```
curl -u "<user@example.net>:<api_key>" -H "Content-Type: application/json" "http://
↪<app-example.net>/api/public/v1.0/groups/<group_id>/automationConfig" --digest -i -
↪X PUT --data '
<configuration_document>
'
```

Upon successful update of the configuration, the API returns the `HTTP/1.1 200 OK` status code to indicate the request has succeeded.

Step 8: Confirm successful update of the automation configuration.

Retrieve the automation configuration to compare it against the document you sent. In particular, confirm that the `version` field equals `2`.

Issue a command similar to the following. Replace the credentials, API key, `<app-example.net>` URL, and group `id` as in previous steps.

```
curl -u "<user@example.net>:<api_key>" "http://<app-example.net>/api/public/v1.0/
↪groups/<group_id>/automationConfig" --digest -i
```

Step 9: Check the deployment status to ensure goal state is reached.

Use the *automationStatus* resource to retrieve the deployment status. Issue the following command, replacing the credentials, API key, URL, and group `id` as in previous steps.

```
curl -u "<user@example.net>:<api_key>" "http://<app-example.net>/api/public/v1.0/
↪groups/<group_id>/automationStatus" --digest -i
```

The command returns the `processes` array and the `goalVersion` field. The `processes` array contains a document for each server that is to run a MongoDB instance, similar to the following:

```
{
  "plan": [],
  "lastGoalVersionAchieved": 2,
  "name": "<process_name_1>",
  "hostname": "<server_1.example.net>",
}
```

If any document has a `lastGoalVersionAchieved` field equal to `1`, the configuration is in the process of deploying. The document's `plan` field displays the remaining work. Wait several seconds and issue the `curl` command again.

When all `lastGoalVersionAchieved` fields equal the value specified in the `goalVersion` field, the new configuration has successfully deployed.

To view the new configuration in the Ops Manager web interface, select the *Deployment* tab and then the *Deployment* page.

Next Steps

To make an additional version of MongoDB available in the cluster, follow the steps in *Update the MongoDB Version of a Deployment*.

Update the MongoDB Version of a Deployment

On this page

Overview

This tutorial describes how to use the *API* to migrate a MongoDB deployment to a new version of MongoDB. These steps assume you have an existing deployment that uses a `2.4.x` version of MongoDB, as would be the case if you used the tutorial to *Deploy a Cluster through the API*, that you will migrate to a `2.6` or later deployment.

Beginning with version `2.6`, MongoDB implemented a new format for MongoDB configuration options. These steps describe how to modify an existing `2.4` sharded cluster to conform to the `2.6` configuration format.

Consideration

The API supports the MongoDB options listed on the *Supported MongoDB Options for Automation* page.

Prerequisite

You must have credentials to access Ops Manager as a user with the *Global Owner* role.

Procedure

Step 1: Retrieve the automation configuration from Ops Manager.

Use the *automationConfig* resource to retrieve the configuration. Issue the following command, replacing `<user@example.net>` with the credentials of the *global owner*, `<api_key>` with the previously retrieved API key, `<app-example.net>` with the URL of Ops Manager, and `<group_id>` with the previously retrieved group ID:

```
curl -u "<user@example.net>:<api_key>" "http://<app-example.net>/api/public/v1.0/
↪groups/<group_id>/automationConfig" --digest -i
```

Confirm that the `version` field of the retrieved *automation configuration* matches the `version` field in the `mms-cluster-config-backup.json` file.

Step 2: Open the configuration document for editing.

As you edit the configuration document in the next steps, reference the *description of an automation configuration* for detailed descriptions of settings.

Step 3: Add the new MongoDB version number to the configuration document.

Update the `mongoDbVersions` array to include `{"name": "2.6.7"}`:

```
"mongoDbVersions": [
    {"name": "2.4.12"},
    {"name": "2.6.7"}
]
```

Step 4: Update the MongoDB configuration options to the `2.6` syntax.

Update **each document** in the `processes` array as follows.

Update the `processes.version` field to specify `2.6.7`:

```
"version": "2.6.7"
```

Change the `processes.args2_4` field to `processes.args2_6`:

```
"args2_6": {
```

Edit the options in the `processes.args2_6` field to conform with the MongoDB 2.6 structure described in Configuration File Options in the MongoDB manual. For example:

```
"args2_6": {
    "net": {
        "port": 27017
    },
    "storage": {
        "dbPath": "/data/"
    },
    "systemLog": {
        "path": "/data/mongodb.log",
        "destination": "file"
    },
    "replication": {
        "replSetName": "rs1"
    },
    "operationProfiling": {}
}
```

Step 5: Send the configuration document.

Use the *automationConfig* resource to send the updated automation configuration.

Issue the following command, replacing `<user@example.net>` with the credentials of the *global owner*, `<api_key>` with the API key, `<app-example.net>` with the Ops Manager URL, and `<group_id>` with the group `id`. Replace `<configuration_document>` with the **entire** updated configuration document.

```
curl -u "<user@example.net>:<api_key>" -H "Content-Type: application/json" "http://
↪<app-example.net>/api/public/v1.0/groups/<group_id>/automationConfig" --digest -i -
↪X PUT --data '
<configuration_document>
'
```

Upon successful update of the configuration, the API returns the HTTP `200 OK` status code to indicate the request has succeeded.

Step 6: Confirm successful update of the automation configuration.

Retrieve the automation configuration from Ops Manager to compare it against the document you sent. Issue a command similar to the following. Replace the credentials, API key, URL, and group `id` as in the previous step.

```
curl -u "<user@example.net>:<api_key>" "http://<app-example.net>/api/public/v1.0/
↪groups/<group_id>/automationConfig" --digest -i
```

Step 7: Check the deployment status to ensure goal state is reached.

Use the *automationStatus* resource to retrieve the deployment status. Issue the following command, replacing the credentials, API key, URL, and group `id` as in previous steps.

```
curl -u "<user@example.net>:<api_key>" "http://<app-example.net>/api/public/v1.0/
↪groups/<group_id>/automationStatus" --digest -i
```

Confirm that the values of all the `lastGoalVersionAchieved` fields in the `processes` array match the `goalVersion` field. For more information on deployment status, see *Automation Status*.

To view the new configuration in the Ops Manager web interface, select the *Deployment* tab and then the *Deployment* page.

10 Troubleshooting

On this page

This document provides advice for troubleshooting problems with Ops Manager.

10.1 Getting Started Checklist

To begin troubleshooting, complete these tasks to check for common, easily fixed problems:

1. *Authentication Errors*
2. *Check Agent Output or Log*
3. *Confirm Only One Agent is Actively Monitoring*
4. *Ensure Connectivity Between Agent and Monitored Hosts*
5. *Ensure Connectivity Between Agent and Ops Manager Server*
6. *Allow Agent to Discover Hosts and Collect Initial Data*

Authentication Errors

If your MongoDB instances run with authentication enabled, ensure Ops Manager has the MongoDB credentials. See *Configure MongoDB Authentication and Authorization*.

Check Agent Output or Log

If you continue to encounter problems, check the agent's output for errors. See *Agent Logs* for more information.

Confirm Only One Agent is Actively Monitoring

If you *run multiple Monitoring Agents*, make sure they are all the same version and that only one if actively monitoring. When you upgrade a Monitoring Agent, do not forget to delete any old standby agents.

When you run multiple agents, one runs as the primary agent and the others as standby agents. Standby agents poll Ops Manager periodically to get the configuration but do not send data.

To determine which agent is the primary agent, look at the *Status* value on the *Administration* tab's *Agents* page. If there is no last ping value for a listed agent, the agent is a standby agent.

See *Monitoring FAQs* and *Add Existing MongoDB Processes to Monitoring* for more information.

Ensure Connectivity Between Agent and Monitored Hosts

Ensure the system running the agent can resolve and connect to the MongoDB instances. To confirm, log into the system where the agent is running and issue a command in the following form:

```
mongo [hostname]:[port]
```

Replace `[hostname]` with the hostname and `[port]` with the port that the database is listening on.

Ensure Connectivity Between Agent and Ops Manager Server

Verify that the Monitoring Agent can connect on TCP port 443 (outbound) to the Ops Manager server (i.e. "`mms.mongodb.com`".)

Allow Agent to Discover Hosts and Collect Initial Data

Allow the agent to run for 5-10 minutes to allow host discovery and initial data collection.

10.2 Installation

Why doesn't the monitoring server startup successfully?

Confirm the URI or IP address for the Ops Manager service is stored correctly in the *`mongo.mongoUri`* property in the `<install_dir>/conf/conf-mms.properties` file:

```
mongo.mongoUri=<SetToValidUri>
```

If you don't set this property, Ops Manager will fail while trying to connect to the default 127.0.0.1:27017 URL.

If the URI or IP address of your service changes, you must update the property with the new address. For example, update the address if you deploy on a system without a static IP address, or if you deploy on EC2 without a fixed IP and then restart the EC2 instance.

If the URI or IP address changes, then each user who access the service must also update the address in the URL used to connect and in the client-side `monitoring-agent.config` files.

If you use the Ops Manager `<install_dir>/bin/credentialstool` to encrypt the password used in the `mongo.mongoUri` value, also add the *`mongo.encryptedCredentials`* key to the `<install_dir>/conf/conf-mms.properties` file and set the value for this property to true:

```
mongo.encryptedCredentials=true
```

10.3 Monitoring

Alerts

For information on creating and managing alerts, see *Create an Alert Configuration* and *Manage Alerts*.

Cannot Turn Off Email Notifications

There are at least two ways to turn off alert notifications:

- Remove the deployment from your Ops Manager account. See *Remove Processes from Monitoring*.
- Disable or delete the alert in Ops Manager. Click the *Activity* tab then click *Alert Settings*. To the right of an alert, select the *gear icon* and select *Disable* or *Delete*.

Receive Duplicate Alerts

If the notification email list contains multiple email-groups, one or more people may receive multiple notifications of the same alert.

Receive "Host has low open file limits" or "Too many open files" error messages

These error messages appear on the *Deployment* page, under a host's name. They appear if the number of available connections does not meet an Ops Manager-defined minimum value. These errors are not generated by the `mongos` instance and, therefore, will not appear in `mongos` log files.

On a host by host basis, the Monitoring Agent compares the number of open file descriptors and connections to the maximum connections limit. The max open file descriptors ulimit parameter directly affects the number of available server connections. The agent calculates whether or not enough connections exist to meet an Ops Manager-defined minimum value.

In ping documents, for each node and its `serverStatus.connections` values, if the sum of the `current` value plus the `available` value is less than the `maxConns` configuration value set for a monitored host, the Monitoring Agent will send a *Host has low open file limits* or *Too many open files* message to Ops Manager.

Ping documents are data sent by Monitoring Agents to Ops Manager. To view ping documents, click the *Deployment* page, then click the host's name, and then click *Last Ping*.

To prevent this error, we recommend you set `ulimit` open files to 64000. We also recommend setting the `maxConns` command in the mongo shell to at least the recommended settings.

See the MongoDB ulimit reference page and the the MongoDB maxConns reference page for details.

Deployments

Deployment Hangs in `In Progress`

If you have added or restarted a deployment and the deployment remains in the `In Progress` state for several minutes, click *View Agent Logs* and look for any errors.

If you diagnose an error and need to correct the deployment configuration:

1. Click *Edit Configuration* and then click *Edit Configuration* again.
2. Reconfigure the deployment.
3. When you complete your changes, click *Review & Deploy* and then *Confirm & Deploy*.

If you shut down the deployment and still cannot find a solution, *unmanage the deployment*.

Monitoring Agent Fails to Collect Data

Possible causes for this state:

- If the Monitoring Agent can't connect to the server because of networking restrictions or issues (i.e. firewalls, proxies, routing.)
- If your database is running with SSL. You must enable SSL either globally or on a per-host basis. See *Configure Monitoring Agent for SSL* and *Configure SSL for MongoDB* for more information.
- If your database is running with authentication. You must supply Monitoring with the authentication credentials either when you're adding a host *or* by clicking the gear icon and then *Edit Host* to the right of the entry on the *Deployment* page.

Hosts

Hosts are not Visible

Problems with the Monitoring Agent detecting hosts can be caused by a few factors.

Host not added to |mms|: In Ops Manager, select the *Deployment* tab, then the *Deployment* page, and then click the *Add Host* button. In the *New Host* window, specify the host type, internal hostname, and port. If appropriate, add the database username and password and whether or not Ops Manager should use SSL to connect with your Monitoring Agent. Note it is not necessary to restart your Monitoring Agent when adding (or removing) a host.

Accidental duplicate mongods If you add the host after a crash and restart the Monitoring Agent, you might not see the hostname on the Ops Manager *Deployment* page. Ops Manager detects the host as a duplicate and suppresses its data. To reset, select the *Administration* tab, then *Group Settings*, and then the *Reset Duplicates* button.

Too many Monitoring Agents installed: Only one Monitoring Agent is needed to monitor all hosts within a single network. You can use a single Monitoring Agent if your hosts exist across multiple data centers and can be discovered by a single agent. Check you have only one Monitoring Agent and remove old agents after upgrading the Monitoring Agent.

A second Monitoring Agent can be set up for redundancy. However, the Ops Manager Monitoring Agent is robust. Ops Manager sends an *Agent Down* alert only when there are no available Monitoring Agents available. See *Monitoring FAQ* and *Monitoring Architecture* for more information.

Cannot Delete a Host

In rare cases, the `mongod` is brought down and the replica set is reconfigured. The down host cannot be deleted and returns an error message, "This host cannot be deleted because it is enabled for backup." Contact |mms| Support for help in deleting these hosts.

For more information on deleted hosts, see *Remove Hosts*.

Groups

Cannot Delete a Group

Please contact your Ops Manager administrator to remove a company or group from your Ops Manager account.

Additional Information on Groups

Create a group to monitor additional segregated systems or environments for servers, agents, users, and other resources. For example, your deployment might have two or more environments separated by firewalls. In this case, you would need two or more separate Ops Manager groups.

API and shared secret keys are unique to each group. Each group requires its own agent with the appropriate API and shared secret keys. Within each group, the agent needs to be able to connect to all hosts it monitors in the group.

For information on creating and managing groups, see *Manage Groups*.

Munin

Install and configure the `munin-node` daemon on the monitored MongoDB server(s) before starting Ops Manager monitoring. The Ops Manager agent README file provides guidelines to install `munin-node`. However, new versions of Linux, specifically Red Hat Linux (RHEL) 6, can generate error messages. See *Configure Hardware Monitoring with munin-node* for details about monitoring hardware with `munin-node`.

Restart `munin-node` after creating links for changes to take effect.

"No package munin-node is available" Error

To correct this error, install the most current version of the Linux repos. Type these commands:

```
sudo yum install http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.
↪noarch.rpm
```

Then type this command to install `munin-node` and all dependencies:

```
sudo yum install munin-node
```

Non-localhost IP Addresses are Blocked

By default, munin blocks incoming connections from non-localhost IP addresses. The `/var/log/munin/munin-node.log` file will display a "Denying connection" error for your non-localhost IP address.

To fix this error, open the `munin-node.conf` configuration file and comment out these two lines:

```
allow ^127\.0\.0\.1$
allow ^::1$
```

Then add this line to the `munin-node.conf` configuration file with a pattern that matches your subnet:

```
cidr_allow 0.0.0.0/0
```

Restart `munin-node` after editing the configuration file for changes to take effect.

Verifying iostat and Other Plugins/Services Returns "# Unknown service" Error

The first step is to confirm there is a problem. Open a telnet session and connect to `iostat`, `iostat_ios`, and `cpu`:

```
telnet HOSTNAME 4949 <default/required munin port>
fetch iostat
fetch iostat_ios
fetch cpu
```

The `iostat_ios` plugin creates the `iotime` chart, and the `cpu` plugin creates the `cputime` chart.

If any of these telnet `fetch` commands returns an "# Unknown Service" error, create a link to the plugin or service in /etc/munin/plugins/ by typing these commands:

```
cd /etc/munin/plugins/
sudo ln -s /usr/share/munin/plugins/<service> <service>
```

Replace `<service>` with the name of the service that generates the error.

Disk names are not listed by Munin

In some cases, Munin will omit disk names with a dash between the name and a numerical prefix, for example, `dm-0` or `dm-1`. There is a documented fix for Munin's iostat plugin.

10.4 Authentication

Two-Factor Authentication

Missed SMS Authentication Tokens

Unfortunately SMS is not a 100% reliable delivery mechanism for messages, especially across international borders. The Google authentication option is 100% reliable. Unless you must use SMS for authentication, use the Google Authenticator application for two-factor authentication.

If you do not receive the SMS authentication tokens:

1. Refer to the *Administration* page for more details about using two-factor authentication. This page includes any limitations which may affect SMS delivery times.
2. Enter the SMS phone number with country code first followed by the area code and the phone number. Also try 011 first followed by the country code, then area code, and then the phone number.

If you do not receive the authentication token in a reasonable amount of time contact |mms| Support to rule out SMS message delivery delays.

How to Delete or Reset Two-Factor Authentication

Contact your system administrator to remove or reset two-factor authentication on your account.

For administrative information on two-factor authentication, see *Manage Two-Factor Authentication for Ops Manager*.

LDAP

Cannot Enable LDAP

You must enable LDAP **before** you sign up the first user through the Ops Manager interface, as required in *Authentication Requirements*. If you cannot enable LDAP because you have created a user through Ops Manager, you

must reinstall Ops Manager, being sure to enable LDAP before creating users and hosts. Please see *Authentication Requirements*.

Forgot to Change MONGODB-CR Error

If your MongoDB deployment uses LDAP for authentication, and you find the following error message:

```
You forget to change "MONGODB-CR" to "LDAP (PLAIN)" since they both
take username/password.
```

Then make sure that you specified the `LDAP (PLAIN)` as is the authentication mechanism for both the Monitoring Agent and the Backup Agent. See *Configure Backup Agent for LDAP Authentication* and *Configure Monitoring Agent for LDAP*.

All Deployments

Networking

All hosts must be able to allow communication between MongoDB ports. The default is `27017`, but you can configure alternate port ranges in the Ops Manager interface.

The Automation Agent *must* be able to connect to `mms.mongodb.com` on port `443` (i.e. `https`). For more information on access to ports and IP addresses, see *Security Overview*.

Directory and File Permissions

The Automation Agent directories and files require the permissions describe here. Paths and filenames vary depending on the operating system.

- **Automation Agent Directory**

  The agent directory and the agent configuration file require *Read* and *Execute* permissions for the user that runs the Automation Agent.

  On RHEL, CentOS, Amazon Linux, & Ubuntu, the agent directory is `/etc/mongodb-mms`. The agent stores its configuration in the `automation-agent.config` file in that directory:

  On other Linux systems and on OS X, you will define the agent directory during installation. The agent stores its configuration in the `local.config` file in that directory.

- **Supporting Files**

  Supporting files include the Monitoring and Backup Agents, the MongoDB binaries, and a backup copy of the JSON-based automation configuration file. The directory that stores these requires `Read`, `Write`, and `Execute` permissions for the user that runs the Automation Agent. The agent requires write permissions are so that the agent can write to the automation configuration file.

  On RHEL, CentOS, Amazon Linux, & Ubuntu, Automation stores these in the same directory as the Automation Agent.

  On other Linux operating systems and on OS X, Automation stores these in the `/var/lib/mongodb-mms-automation` directory.

- **Log File**

  The log file requires `Write` permission for the user that runs the Automation Agent. By default, the agent logs events in the following log file:

```
/var/log/mongodb-mms-automation/automation-agent.log
```

The Automation Agent's configuration file specifies the location of the log file, as well as the log level and log-rotation settings.

Automation Configuration

After completing the automation configuration, always ensure that the deployment plan satisfies the needs of your deployment. Always double check hostnames and ports before confirming the deployment.

Sizing

- Ensure that you provision machines with enough space to run MongoDB and support the requirements of your data set.
- Ensure that you provision sufficient machines to run your deployment. Each `mongod` should run on its own host.

Operating System

The Automation Agent only supports Linux and OS X hosts. The Automation Agent does not support Windows.

10.5 Backup

Logs Display MongodVersionException

The `MongodVersionException` can occur if the *Backup Daemon's* host cannot access the internet to download the version or versions of MongoDB required for the backed-up databases. Each database requires a version of MongoDB that matches the database's version. Specifically, for each instance you must run the latest stable release of that release series. For versions earlier than 2.4, the database requires the latest stable release of 2.4.

If the Daemon runs without access to the internet, you must manually download the required MongoDB versions, as described here:

1. Go to the MongoDB downloads page and download the appropriate versions for your environment.
2. Copy the download to the Daemon's host.
3. Decompress the download into the directory specified in the *`mongodb.release.directory`* setting in the Daemon's `conf-daemon.properties` file. For the file's location, see *Ops Manager Configuration Files*.

   Within the directory specified in the *`mongodb.release.directory`* setting, the folder structure for MongoDB should look like the following:

```
<path-to-mongodb-release-directory>/
|-- mongodb-<platform>
|   |-- THIRD-PARTY-NOTICES
|   |-- README
|   |-- GNU-AGPL-3.0
|   |-- bin
|   |   |-- bsondump
|   |   |-- mongo
|   |   |-- mongod
```

```
|   |   |-- mongodump
|   |   |-- mongoexport
|   |   |-- mongofiles
|   |   |-- mongoimport
|   |   |-- mongooplog
|   |   |-- mongoperf
|   |   |-- mongorestore
|   |   |-- mongos
|   |   |-- mongostat
|   |   |-- mongotop
```

10.6 System

Logs Display OutOfMemoryError

If your logs display `OutOfMemoryError`, ensure you are running with sufficient ulimits and RAM.

Increase Ulimits

For the recommended ulimit setting, see the FAQ on *Receive "Host has low open file limits" or "Too many open files" error messages*.

Ops Manager infers the host's `ulimit` setting using the total number of available and current connections. For more information about ulimits in MongoDB, see the UNIX ulimit Settings reference page.

Ensure Sufficient RAM for All Components

- Ensure that each server has enough RAM for the components it runs. If a server runs multiple components, its RAM must be at least the sum of the required amount of RAM for **each** component.

  For the individual RAM requirements for the Ops Manager Application server, Ops Manager Application database, Backup Daemon server, and Backup Blockstore database, see *Ops Manager Hardware and Software Requirements*.

10.7 Automation Checklist

Ops Manager Automation allows you to deploy, configure, and manage MongoDB deployments with the Ops Manager UI. Ops Manager Automation relies on an Automation Agent, which must be installed on every server in the deployment. The Automation Agents periodically poll the Ops Manager service to determine the current goal, and continually report their status to Ops Manager.

To use Automation, you must install the Automation Agent on each server that you want Ops Manager to manage.

11 Frequently Asked Questions

On this page

- *Backup FAQs*
- *Administration FAQs*

This document addresses common questions about Ops Manager and its use.

11.1 Monitoring FAQs

Host Configuration

How do I add a new host or server?

See *Add Existing MongoDB Processes to Monitoring*.

Can Ops Manager monitor itself?

Yes. You can use Ops Manager to monitor the backing MongoDB replica sets that hold data for the Ops Manager Application and Backup Daemon.

You **cannot**, however, use Ops Manager to backup or automate the backing replica sets.

Can I monitor Kerberos-enabled instances?

Yes. Monitoring does support monitoring for Kerberos-enabled MongoDB instances. See *Configure the Monitoring Agent for Kerberos* for more information.

How does Ops Manager gather database statistics?

In most instances, Monitoring will scale its request cycle to limit more expensive statistics gathering. The DB Stats information updates every 10 minutes, and the agent will throttle the frequency to reduce the impact on the database.[1] Even so, the "DB stats" operation impacts the performance of your database, as is possible when installations have a large number of databases and collections.

If the collection of database statistics impacts database performance, disable database stats collection before starting your agent. Select the *Administration* tab, then the *Group Settings* page, and then set *Collect Database Specific Statistics* to `NO`.

Monitoring Agent

Do I need a Monitoring Agent for every MongoDB Instance?

No. A single Monitoring Agent can connect to all MongoDB databases in your Ops Manager group. Unless you have multiple groups, complete your initial Monitoring Agent setup with a single agent.

For redundancy, you may wish to run a second Monitoring Agent. See the *Monitoring Agent Redundancy* for more information.

[1] DB Stats will not appear until 30 minutes after you add the host to Monitoring

Can I use two Monitoring Agents to connect MongoDBs in different data centers?

No, not within the same group. The group's Monitoring Agent must connect to every server in the MongoDB deployment. Configure firewalls to allow the Monitoring Agent to connect across data centers and servers.

Use multiple Monitoring Agents within a single Ops Manager group *only to provide redundancy*. For each Ops Manager group, the agent must be able to connect to every monitored MongoDB. Unless you have multiple groups, complete your initial Monitoring Agent setup with a single agent.

What happens if a Monitoring Agent becomes unavailable? How can I ensure my MongoDBs are always monitored?

You can run multiple Monitoring Agents. If one Monitoring Agent fails, another starts monitoring. As long as at least one Monitoring Agent is available, Ops Manager will not trigger a *Monitoring Agent Down* alert. To run multiple Monitoring Agents, see *Monitoring Agent Redundancy*.

You also can create an alert to notify you when an agent is down. In Ops Manager, click the *Activity* tab and then *Alert Settings*. Click the *Add Alert* button then set the alert through the fields in the *Create a New Alert* window.

Where should I run the Monitoring Agent?

The amount of resources the Monitoring Agent requires varies depending on infrastructure size, the number of servers and the databases it's monitoring. Run the agent on an existing machine with additional capacity that *does not* run a `mongod` instance. You may also run the Monitoring Agent on a smaller dedicated instance.

The Monitoring Agent load scales with the number of monitored `mongod` plus `mongos` processes and the number of databases in your MongoDB environment.

Never install the Monitoring Agent on the same server as a data bearing `mongod` instance. This will allow you to perform maintenance on a the `mongod` and its host without affecting the monitoring for your deployment. Additionally a Monitoring Agent may contend for resources with the `mongod`

You can install the Monitoring Agent on the same system as an arbiter, a `mongos`, or an application server depending on the requirements of these services and available resources.

Can I run the Monitoring Agent on an AWS micro server?

If you monitor five or fewer `mongod` instances, you can use a AWS micro server.

Why can't the Monitoring Agent connect to my host?

The most common problem is that the agent is unable to resolve the hostname of the server. Check DNS and the `/etc/hosts` file.

The second most common problem is that there are firewall rules in place that prohibit access to the server from the agent.

To test the connection, login to the server running the agent and run: `mongo hostname:port/test` If you are unable to connect, the agent will not be able to connect.

In addition, Monitoring supports monitoring for Kerberos-enabled instances.

Why does the Monitoring Agent connect with hostnames instead of IP addresses?

By default, the Monitoring Agent tries to connect by resolving hostnames. If the agent cannot connect by resolving a hostname, you can force the Monitoring Agent to prefer an IP address over its corresponding hostname for a specific IP address.

To create a preferred hostname, select the *Administration* tab, then the *Group Settings* page, and then click the *Add* button for the *Preferred Hostnames* setting. If your IP addresses have a common prefix, create a preferred hostname with the *ends-with* button or click the *regexp* button to use a regular expression.

Preferred hostnames also allow you to specify the hostname to use for servers with multiple aliases. This prevents servers from appearing multiple times under different names in the Ops Manager interface.

How do I download the Monitoring Agent?

You can update the Monitoring Agent from the *Agents* page on the Ops Manager Administration tab.

How do I setup and configure the agent?

See the `README` file included in the agent download.

How do I delete a Monitoring Agent from Ops Manager?

Monitoring Agents report their status to the Ops Manager. When an agent does not report for more than 24 hours, the agent no longer appears in Ops Manager.

For more details, see *Remove Monitoring Agents from Ops Manager*.

Can I run the Ops Manager Monitoring Agent with Ops Manager Backup?

Yes. Both the Ops Manager Monitoring Service and Ops Manager Backup can operate in the same environment. You will need to install and configure two separate Monitoring Agents: configure one agent for the Ops Manager environment and the other for the Ops Manager Service.

Data Presentation

What are all those vertical bars in my charts?

A *red bar* indicates a server restart.

A *orange bar* indicates the server is now a primary.

A *brown bar* indicates the server is now a secondary.

Why is my Monitoring Agent highlighted in red?

Your agent is out of date.

You can update the Monitoring Agent from the *Agents* page on the Ops Manager Administration tab.

Data Retention

What is the data retention policy for Ops Manager?

Ops Manager retains two distinct types of data: metrics, which describe usage; and snapshots, which back up your data.

Data-retention policies, as defined in the Terms of Service, are always subject to change.

As of this writing, Ops Manager preserves:

- Minute-level metrics for 48 hours.
- Hourly metrics for 94 days.
- Snapshots according to their *retention policy*.

11.2 Backup FAQs

Ops Manager Backup creates backups of MongoDB replica sets and sharded clusters. After an initial sync to MongoDB's datacenters, Ops Manager Backup tails the operation log (oplog) to provide a continuous backup with point-in-time recovery of replica sets and consistent snapshots of sharded clusters. For more information, please review these frequently asked questions or create an Ops Manager Backup account.

Requirements

What version of MongoDB does Backup require?

To back up a sharded cluster, Backup requires version 2.4.3 or later.

To back up a replica set, Backup requires version 2.2 or later.

Are there any limits to the types of deployments Backup supports?

Yes. Backup does not currently support standalone deployments. Backup has full support for replica sets and sharded clusters.

Why doesn't Backup support standalone deployments?

After an initial sync of your data to Ops Manager, Backup copies data from the oplog to provide a continuous backup with point-in-time recovery. Backup does not support standalone servers, which do not have an oplog. To support backup with a single `mongod` instance, you can run a one-member replica set.

See also:

Convert a Standalone to a Replica Set.

How Does Ops Manager Measure Data Size?

Ops Manager uses the following conversions to measure snapshot size and to measure how much oplog data has been processed:

- 1 MB = 1024^2 bytes

- 1 GB = 1024^3 bytes
- 1 TB = 1024^4 bytes

Interface

How can I verify that I'm running the latest version of the Backup Agent?

If your Backup Agent is out of date, it will be highlighted in red on the *Agents* page of the *Administration* tab.

Why is my Backup Agent highlighted in red?

If your agent is highlighted in red on the *Agents* page of the *Administration* tab, your agent is out of date. For instructions on updating the agent, see *Install Backup Agent*.

Operations

How does Backup work?

You install the Backup Agent on a server in the same deployment with your MongoDB infrastructure. The agent conducts an initial sync of your data to Ops Manager. After the initial sync, the agent tails the oplog to provide a continuous backup of your deployment.

Where should I run the Backup Agent?

The Backup Agent can run anywhere in your infrastructure that has access to your `mongod` instances. To avoid contention for network and CPU resources, *do not* run the Backup Agent on the same hosts that provide your `mongod` instances.

The Backup Agent has the same performance profile impact as a secondary. For the initial backup, the load scales with the size of your data set. Once an initial backup exists, the load scales with oplog gigabytes used per hour.

Can I run the Backup and Monitoring Agents on a Single System?

There is no technical restriction that prevents the Backup Agent and the Monitoring Agent from running on a single system or host. However, both agents have resource requirements, and running both on a single system can affect the ability of these agents support your deployment in Ops Manager.

The resources required by the Backup Agent depend on rate and size of new oplog entries (i.e. total oplog gigabyte/per-hour produced.) The resources required by the Monitoring Agent depend on the number of monitored `mongod` instances and the total number of databases provided by the `mongod` instances.

Can I run multiple Backup Agents to achieve high availability?

You can run multiple Backup Agents for high availability. If you do, the Backup Agents must run on different hosts.

When you run multiple Backup Agents, only one agent per group or environment is the **primary agent**. The primary agent performs the backups. The remaining agents are completely idle, except to log their status as standbys and to periodically ask Ops Manager whether they should become the primary.

Does the Backup Agent modify my database?

The Backup Agent writes a small token called a "checkpoint" into the oplog of the source database at a regular interval. These tokens provide a heartbeat for Backup and have no effect on the source deployment. Each token is less than 100 bytes. See: *Checkpoints* for more about checkpoints.

Will the MongoDB Backup Service impact my production databases?

Backup will typically have minimal impact on production MongoDB deployments. This impact will be similar to that of adding a new secondary to a replica set.

By default, the Backup Agent will perform its initial sync, the most resource intensive operation for Backup, against a secondary member of the replica set to limit its impact. You may optionally configure the Backup Agent to perform the initial sync against the replica set's primary, although this will increase the impact of the initial sync operation.

What is the load on the database during the initial Backup sync?

The impact of the initial backup synchronization should be similar to syncing a new secondary replica set member. The Backup Agent does not throttle its activity, and attempts to perform the sync as quickly as possible.

Can I backup my standalone deployment?

No. Backup does not currently support standalone deployments. To convert to a replica set, consult MongoDB's replication documentation.

How do I perform maintenance on a Replica Set with Backup enabled?

Most operations in a replica set are replicated via the oplog and are thus captured by the backup process. Some operations, however, make changes that are *not* replicated: for these operations you *must* *have the Backup service resync from your current replica set* to include the changes.

The following operations are not replicated and therefore require resync:

- Renaming or deleting a database by deleting the data files in the data directory. As an alternative, remove databases using an operation that MongoDB will replicate, such as `db.dropDatabase()` from the `mongo` shell.
- Changing any data while the instance is running as a standalone.
- Using `compact` or `repairDatabase` to reclaim a significant amount of space. This is not strictly necessary but will ensure that the Ops Manager copy of the data is resized, which means quicker restores and lower costs.

See also:

Maintenance Operations for Replica Set Members.

How Do I Delete a Backup Snapshot?

You can delete replica set backup snapshots and snapshots for replica sets in a sharded cluster set if the snapshots are not needed for point-in-time restores. See *Delete Snapshots for Replica Sets and Sharded Clusters* for details.

Configuration

What are "excluded namespaces"?

Excluded namespaces are databases and collections that Ops Manager will not back up. This is useful for large databases or collections that contain data that you will not need to restore: caches and logs, for example.

How can I prevent Backup from backing up a collection?

Backup allows you to specify "excluded namespaces", which are collections or databases that you do not want Ops Manager to back up.

You can specify the namespaces to exclude when you initially enable backup on a replica set or sharded cluster, or can edit the list at any time by selecting the "*gear icon*" in the *Sharded Cluster Status* or *Replcia Set Status* tables in Ops Manager.

How can I change which namespaces are on the "excluded namespaces" list?

Click on the "*gear icon*" next to the name of the replica set or sharded cluster whose excluded namespaces you want to modify in the *Sharded Cluster Status* or *Replcia Set Status* tables in Ops Manager. A modal window will open, where you can add databases or collections to the list, or remove list items by clicking on the red *x* icon.

Removing a namespace from the excluded namespaces list necessitates a re-sync. Backup handles this re-sync.

How can I use backup if Backup Jobs Fail to Bind?

The most common reason that jobs fail to bind to a Backup daemon is when no daemon has space for local copy of the backed up replica set.

To increase capacity so that the backup job can bind, you can:

- add an additional backup daemon.
- increase the size of the file system that holds the `rootDirectory` directory.
- move the `rootDirectory` data to a new volume with more space, and create a symlink or configure the file system mount points so that the daemon can access the data using the original path.

How do I resolve `applyOps` Errors During Backups?

If you notice *consistent* errors in `applyOps` commands in your Backup logs, it *may* indicate that the daemon has run out of space.

To increase space on a daemon to support continued operations, you can:

- increase the size of the file system that holds the `rootDirectory` directory.
- move the `rootDirectory` data to a new volume with more space, and create a symlink or configure the file system mount points so that the daemon can access the data using the original path.

Restoration

Ops Manager Backup produces a copy of your data files that you can use to seed a new deployment.

How can Ops Manager provide point-in-time restores for any point in time?

Although it is faster to provide a restore for the time at which a snapshot was actually stored, this might not be ideal when restoring a replica set or sharded cluster. In consequence, the Backup service can build a restore to any point in time within a 24-hour period by replaying the oplog to the desired time.

For details, see the *procedures for restoring replica sets and sharded clusters*.

Can I take snapshots more frequently than every 6 hours?

No, Ops Manager does not support a snapshot schedule more frequent than every 6 hours. For more information, see *Snapshot Frequency and Retention Policy*.

Can I set my own snapshot retention policy?

Yes. Administrators can change the *snapshot frequency and retention policy* through the *snapshotSchedule resource* in the API.

For example, you may choose to capture more frequent snapshots for the most mission critical data, and capture snapshots less frequently for less critical data.

How long does it take to create a restore?

Ops Manager transmits all backups in a compressed form from the Ops Manager server to your infrastructure.

In addition, point-in-time restores that require creating a new snapshot take additional time, which depends on the size of the scheduled snapshot and the amount the oplog entries that Backup must apply to the preceding snapshot to roll forward to the requested point-in-time of the backup.

Does Backup perform any data validation?

Backup conducts basic corruption checks and provides an alert if any component (e.g., the agent) is down or broken, but does not perform explicit data validation. When it detects corruption, Backup errs on the side of caution and invalidates the current backup and sends an alert.

How do I restore? What do I get when I restore?

You can request a restore via Ops Manager, where you can then choose which snapshot to restore and how you want Backup to deliver the restore. All restores require 2-factor authentication. Ops Manager will send an authorization code via SMS code to your administrator. You must enter the authorization code into the backup interface to begin the restore process.

Note: From India, use Google Authenticator for two-factor authentication. Google Authenticator is more reliable than authentication with SMS text messages to Indian mobile phone numbers (i.e. country code 91).

Backup delivers restores as `tar.gz` archives of MongoDB data files.

Restore delivery options are:

- **SCP to your Infrastructure**: Backup will transmit the backup to your infrastructure over a secure channel. You must provide connection information for a host in your deployment.
- **Download**: Backup will make your restore data available using a custom, one-time-use URL.

How do I know an SCP restore push has completed and is correct?

When you receive restoration files through an SCP push, Ops Manager sends SHA-1 hash files, also called checksum files, along with the restore files. The hash files have the `.sha1` extension,

To ensure the restore files are complete and correct, use the Unix shasum utility:

```
shasum -c <checksum file>
```

What is the SCP public key for Ops Manager?

Ops Manager generates an SSH public key on a per user basis to use when delivering backups via SCP. To generate a key, see the steps to grant access via SSH public key in *Select Backup File Delivery Method and Format*.

How does Backup handle Rollbacks?

If your MongoDB deployment experiences a rollback, then Ops Manager Backup also rolls back.

Backup detects the rollback when a tailing cursor finds a mismatch in timestamps or hashes of write operations. Backup enters a rollback state and tests three points in the oplog of your replica set's primary to locate a common point in history. Ops Manager rollback differs from MongoDB secondary rollback in that the common point does not necessarily have to be the most *recent* common point.

When Backup finds a common point, the service invalidates oplog entries and snapshots beyond that point and rolls back to the most recent snapshot before the common point. Backup then resumes normal operations.

If Ops Manager cannot find a common point, a *resync* is required.

What conditions will require a resync?

If the Backup Agent's tailing cursor cannot keep up with your deployment's oplog, then you must *resync the Backup Service*.

This scenario might occur, for example, if:

- Your application periodically generates a lot of data, shrinking the primary's oplog window to the point that data is written to the oplog faster than Backup can consume it.
- If the Backup Agent is running on an under-provisioned or over-used machine and cannot keep up with the oplog activity.
- If the Backup Agent is down for a period of time longer than the oplog size allows. If you bring down your agents, such as for maintenance, restart them in a timely manner. For more information on oplog size, see Replica Set Oplog in the MongoDB manual.
- If all you delete all replica set data and deploy a new replica set with the same name, as might happen in a test environment where deployments are regularly torn down and rebuilt.
- If there is a rollback, and Backup cannot find a common point in the oplog.

- If an oplog event tries to update a document that does not exist in the Backup replica set, as might happen if syncing from a secondary that has inconsistent data with respect to the primary.

11.3 Administration FAQs

User and Group Management

How do I reset my password?

You can reset your password using the password reset form.

How do I change my password?

You can change your password by resetting your password.

What are the password requirements?

Passwords must be at least 8 characters long and contain at least one letter, one digit, and one special character.

How do I add a user to my company/group?

Instruct the user to create a new account through the URL for your Ops Manager installation.

After they have created an account, you can add their username to the company/group on the admin page.

How do I remove my company/group?

Please contact your Ops Manager administrator to remove a company or group from your Ops Manager account.

How can I configure multiple Google Authenticator apps to use the same account?

By selecting the *Can't scan the barcode?* option during the procedure to *Configure Two-Factor Authentication with Google Authenticator*. The option provides a common key that multiple Google Authenticator apps can use.

Activity

My alert email says my host(s) are exposed to the public Internet. What does that mean?

This alert indicates only that the Ops Manager server can connect to the `mongod` that it is monitoring. It does not diagnose whether your host is exposed to the public, despite the alert message. This alert occurs if you configured a setting called *Exposed DB Host Check*, which is a setting used with the Cloud version of Ops Manager.

See *Manage Alerts* to disable or modify the exposed host alerts.

How do I modify my alert settings?

To enable, disable, or modify alerts, select the *Activity* tab and then the *Alert Settings* page. For more information, see *Manage Alert Configuration*.

How frequently can alerts be set?

Ops Manager processes alerts on a 5-minute interval. Therefore, the minimum frequency for an alert is 5 minutes. The default frequency for new alert configurations is 60 minutes.

Operations

Do Web or Database Hosting Companies Integrate with Ops Manager?

Web hosting companies can offer the ability to use Ops Manager with their hosted MongoDB databases, for example, to set up software agents to monitor and backup databases hosted on their servers. MongoDB has confirmed compatability with MongoHQ, MongoLab, and Heroku. Implementation details depend on each hosting company.

MongoHQ offers Ops Manager upon request as part of their Database as a Service (DaaS) business.

MongoLab offers Ops Manager as part of their Database as a Service (DaaS) business. MongoLab offers the service on their dedicated plans and shared replica set plan. They also provide instructions to tune MongoDB performance with Ops Manager on their servers.

MongoHQ and MongoLab are MongoDB Advanced Partners.

Heroku offers web hosting with a MongoHQ add-on and MongoLab add-on to use MongoDB databases from these database hosting companies. Heroku also offers Ops Manager monitoring of those databases with detailed setup instructions.

About Ops Manager

What open source projects does Ops Manager use?

- Database: MongoDB
- App framework: Google Guice
- Http server: Jetty
- Web framework: Jersey
- Misc server libs: Apache Commons
- UI lib: jQuery , Bootstrap
- Charts: dygraphs
- Graphics: Font-Awesome

12 Reference

Ops Manager Configuration Files Describes configuration options in the Ops Manager configuration files.

Automation Agent Install and configure the automation agent.

Monitoring Agent Install and configure the Monitoring Agent.

Backup Agent Install and configure the Backup Agent.

Audit Events An inventory of all audit events reported in the activity feed.

Monitoring Reference A reference sheet for the monitoring service.

Supported Browsers Browsers supported by Ops Manager.

Advanced Options for MongoDB Deployments Describes the advanced deployment options for replica sets and sharded clusters.

Automation Configuration Describes the settings available in the cluster configuration file used to determine the desired state of the MongoDB deployment.

Supported MongoDB Options for Automation Supported options for a MongoDB process in an automation configuration.

12.1 Ops Manager Configuration Files

On this page

Overview

The Ops Manager Application and the Backup Daemon use the *conf-mms.properties* and *conf-daemon.properties* configuration files respectively. This document describes all available settings between the two files. Each configuration file uses a subset of the settings described here.

conf-mms.properties

The Ops Manager Application uses the `conf-mms.properties` configuration file. If you installed Ops Manager using an `rpm` or `deb` package, the file's location is:

```
/opt/mongodb/mms/conf/
```

If you installed using a `tar.gz` file, the configuration file's location is:

```
<install-directory>/conf/
```

If you installed on Windows, the file's location is:

```
<install-folder>\Server\Config
```

By default, this is `C:\MMSData\Server\Config`.

conf-daemon.properties

The Backup Daemon uses the `conf-daemon.properties` configuration file. If you installed Ops Manager using an `rpm` or `deb` package, the file's location is:

```
/opt/mongodb/mms-backup-daemon/conf/
```

If you installed using a `tar.gz` file, the configuration file's location is:

```
<install-directory>/conf/
```

If you installed on Windows, the configuration file's location is:

```
<install-folder>\BackupDaemon\Config
```

Mandatory Settings

To start the Ops Manager Application, you must configure the *Ops Manager Application URL Settings* and *Email Address Settings* in the `conf-mms.properties` file.

Security

For configuration settings that store credentials, you can either store the credentials in plain text or use the Ops Manager `credentialstool` to encrypt the credentials, as described in *Encrypt MongoDB User Credentials*.

If you choose to store credentials in plain text, reduce the permissions on the configuration file. For example:

```
sudo chmod 600 <install_dir>/conf/conf-mms.properties
```

Settings

Ops Manager Application URL Settings

The following two settings are mandatory for the Ops Manager Application.

`mms.`**`centralUrl`**
Type: string

Required. Fully qualified URL, including the port number, of the Ops Manager Application. For example,

```
mms.centralUrl=http://mms.example.com:8080
```

If you wish to use a port other than `8080`, *Change the Ops Manager Ports* describes how to change the ports that Ops Manager uses.

`mms.`**`backupCentralUrl`**
Type: string

Required. The hostname and port of the Backup HTTP Service. For example,

```
mms.backupCentralUrl=http://mms.example.com:8081
```

You must set `mms.backupCentralUrl`, even if you are only using Ops Manager Monitoring and not Ops Manager Backup.

If you wish to use a port other than `8081`, *Change the Ops Manager Ports* describes how to change the ports that Ops Manager uses.

Load Balancer

Set the following when using a load balancer with the Ops Manager Application.

`mms.remoteIp.`**`header`**
Type: string

Specify the name of the header that the load balancer will use to specify the original client's IP address to the application server.

See *Configure a Highly Available Ops Manager Application* for more information.

When you specify `mms.remoteIp.header`, do not allow clients to connect directly to any application server.

Ops Manager Application HTTPS Settings

You can configure the Ops Manager Application's application servers to use HTTPS to encrypt connections between the Ops Manager Application, the agents, and the web interface.

The default port for HTTPS access to the Ops Manager Application is `8443`, as set in `<install_dir>/conf/mms.conf` file. If you change this default, change the ports specified in the `mms.centralUrl` and `mms.backupCentralUrl` settings.

`mms.https.`**`PEMKeyFile`**
Type: string

Specify the PEM file that contains the application's valid certificate and private key.

`mms.https.`**`PEMKeyFilePassword`**
Type: string

Required if the PEM file contains an encrypted private key. Specify the password for PEM file. You can encrypt the specified password using the Ops Manager `credentialstool`. See *Encrypt MongoDB User Credentials*.

Email Settings

Email Address Settings

The following email address settings are mandatory. You **must** define them before the Monitoring instance will start.

`mms.`**`fromEmailAddr`**
Type: string

Required. The email address used for sending the general emails, such as Ops Manager alerts. You can include an alias with the email address. For example:

```
mms.fromEmailAddr=|mms| Alerts <mms-alerts@example.com>
```

`mms.`**`replyToEmailAddr`**
Type: string

Required. The email address to send replies to general emails. For example:

```
mms.replyToEmailAddr=mms-no-reply@example.com
```

mms.**adminFromEmailAddr**
Type: string

Required. The email address to send messages from the Ops Manager admin. You can include an alias with the email address. For example:

```
mms.adminFromEmailAddr=|mms| Admin <mms-admin@example.com>
```

mms.**adminEmailAddr**
Type: string

Required. The email address to send messages or replies to the Ops Manager admin. You can include an alias with the email address. For example:

```
mms.adminEmailAddr=mms-admin@example.com
```

mms.**bounceEmailAddr**
Type: string

Required. The email address to send bounce messages, i.e. messages of non-delivery of alerts or messages from Ops Manager admin. For example:

```
mms.bounceEmailAddr=bounce@example.com
```

Email Service Settings

mms.**emailDaoClass**
Type: string

The email interface to use. For AWS Simple Email Service, specify `com.xgen.svc.core.dao.email.AwsEmailDao`, as in:

```
mms.emailDaoClass=com.xgen.svc.core.dao.email.AwsEmailDao
```

For AWS Simple Email Service, see also `aws.accesskey` and `aws.secretkey`.

For JavaEmailDao, specify `com.xgen.svc.core.dao.email.JavaEmailDao`, as in:

```
mms.emailDaoClass=com.xgen.svc.core.dao.email.JavaEmailDao
```

mms.mail.**transport**
Type: string

Default: smtp

Transfer protocol `smtp` or `smtps` as specified by your email provider. For example:

```
mms.mail.transport=smtp
```

mms.mail.**hostname**
Type: string

Default: localhost

Email hostname as specified by your email provider. For example:

```
mms.mail.hostname=mail.example.com
```

`mms.mail.`**`port`**
Type: number

Default: 25

Port number for the transfer protocol as specified by your email provider. For example:

```
mms.mail.port=25
```

`mms.mail.`**`tls`**
Type: boolean

Default: false

Indicator of whether the transfer protocol runs on top of TLS. For example:

```
mms.mail.tls=false
```

`mms.mail.`**`username`**
Type: string

User name of the email account. If unset, defaults to disabled SMTP authentication.

```
mms.mail.username=
```

`mms.mail.`**`password`**
Type: string

Password for the email account. If unset, defaults to disabled SMTP authentication.

```
mms.mail.password=emailPassword
```

`aws.`**`accesskey`**
Type: string

Required if using AWS Simple Email Service. The access key ID for AWS.

```
aws.accesskey=EXAMPLEAccessKeyID
```

`aws.`**`secretkey`**
Type: string

Required if using AWS Simple Email Service. The secret access key for AWS.

```
aws.secretkey=eXampLe/aCcESs/KEY
```

Twilio SMS Alert Settings

To receive alert notifications via SMS, you must have a Twilio account and specify your Twilio account information in the configuration file.

`twilio.account.`**`sid`**
Type: string

Twilio account ID.

`twilio.auth.token`
Type: string

Twilio API token.

`twilio.from.num`
Type: string

Twilio phone number.

MongoDB Settings

The following settings configure the Ops Manager connections to the *Ops Manager Application Database*.

Connection String

`mongo.mongoUri`
Type: string

Required. The connection string used to access the *backing MongoDB instance*. The `conf-mms.properties` file can contain multiple `mongo.mongoUri` settings. The following example specifies connection to a replica set:

```
mongo.mongoUri=mongodb://db1.example.net:40000,db2.example.net:40000,db3.example.
↪net:40000
```

If you omit the port number, Ops Manager uses the default `27017` port for all hosts.

For a backing MongoDB instance with **access control**, the connection string must include authentication credentials. The connecting user must possess the `readWriteAnyDatabase`, `dbAdminAnyDatabase`, and `clusterMonitor` user roles. If the database is a sharded cluster, the `clusterAdmin` role is required instead of `clusterMonitor`.

The following examples show the formats to use for the different `authentication mechanisms`.

For a MongoDB instance using the default `MONGODB-CR` / `SCRAM-SHA-1` challenge-response mechanism, prefix the hostname with the MongoDB username and password in the form `<username>:<password>@`

```
mongo.mongoUri=mongodb://mongodbuser1:password@mydb1.example.net:40000
```

For a MongoDB instance using `MONGODB-X509` authentication, you must first add the value of the **subject** from the client certificate as a MongoDB user, as described in Use x.509 Certificates to Authenticate Clients in the MongoDB manual. The client certificate is contained in the PEM file you specify in the `mongodb.ssl.PEMKeyFile` setting. Once you have created the user, prefix the host specified in `mongo.mongoUri` with the name of the new user and append `authMechanism=MONGODB-X509` after the specified port:

```
mongo.mongoUri=mongodb://<new_mongodb_user>@mydb1.example.net:40000/?
↪authMechanism=MONGODB-X509
```

For a MongoDB instance using **LDAP**, prefix the hostname with the MongoDB username and password in the form `<username>:<password>@`, and append the `authMechanism=PLAIN&authSource=$external` options after the port:

```
mongo.mongoUri=mongodb://mongodbuser1:password@mydb1.example.net:40000/?
↪authMechanism=PLAIN&authSource=$external
```

For a MongoDB instance using Kerberos, prefix the hostname with the Kerberos user principal and specify the authentication mechanism, `authMechanism=GSSAPI`, after the port.

Kerberos user principal names have the form `<username>@<KERBEROS REALM>`. You must escape the user principal, replacing symbols with the URL encoded representation. A Kerberos user principal of `username@REALM.EXAMPLE.COM` would therefore become `username%40REALM.EXAMPLE.COM`.

The following is an example of Kerberos authentication:

```
mongo.mongoUri=mongodb://username%40REALM.EXAMPLE.COM@mydb1.example.net:40000/?
↪authMechanism=GSSAPI
```

To enable Kerberos authentication between the Ops Manager Application and the backup-database, see *Kerberos Settings*. See also `authMechanism` and `authSource` in the MongoDB manual.

MongoDB SSL Settings

The following settings in `conf-mms.properties` and `conf-daemon.propeties` configure Ops Manager to use SSL to encrypt connections to the *backing MongoDB instances* that host the *Ops Manager Application Database* and *Backup Blockstore Database*.

`mongo.`**`ssl`**
Type: boolean

Enables SSL connection to the *Ops Manager Application Database* when set to `true`.

`mongodb.ssl.`**`CAFile`**
Type: string

The name of the PEM file that contains the root certificate chain from the Certificate Authority that signed the MongoDB server certificate.

`mongodb.ssl.`**`PEMKeyFile`**
Type: string

The name of the PEM file that contains the X509 certificate and private key. Required if the MongoDB instance is running with the `--sslCAFile` option. For more information on the option, see `net.ssl.CAFile` in the MongoDB manual.

If you authenticate using the `MONGODB-X509` authentication mechanism, you also enter this as the name of the user in the `mongoUri` connection string.

`mongodb.ssl.`**`PEMKeyFilePassword`**
Type: string

Required if the PEM file contains an encrypted private key. Specify the password for PEM file. You can encrypt the specified password using the Ops Manager `credentialstool`. See *Encrypt MongoDB User Credentials*.

Encrypted Credentials

`mongo.`**`encryptedCredentials`**
Type: boolean

Add this property and set it to `true` if `mongo.mongoUri` contains the encrypted username and password:

```
mongo.encryptedCredentials=true
```

You must encrypt the username and password in `mongo.mongoUri` using the Monitoring `credentialstool`. See *Encrypt MongoDB User Credentials*.

Important: The `conf-mms.properties` file can contain multiple `mongo.mongoUri` settings. If `mongo.encryptedCredentials` is `true`, you must encrypt all user credentials found in the various `mongo.mongoUri` settings.

Automation Versions Settings

The following settings in the `conf-mms.properties` file determine how Ops Manager knows what MongoDB releases exist and from what servers the Automation Agent downloads the binaries for a MongoDB release.

`automation.versions.`**`source`**

Type: string

Default: mongodb

Selects whether the Automation Agents retrieve MongoDB binaries over the internet from MongoDB Inc. or locally from the Ops Manager Application server. Set this to `mongodb` if the Automation Agents have internet access to retrieve the binaries from MongoDB Inc. Set this to `local` if your Automation Agents cannot reach the internet. If you specify `local`, you must set the `automation.versions.directory` and place `.tgz` archive files for the MongoDB binaries in the specified directory. You must also provide the MongoDB version manifest. For details, see *Configure Local Mode if Ops Manager has No Internet Access*.

`automation.versions.`**`directory`**

Type: string

Default:

When `automation.versions.source` is set to `local`, this specifies the directory on the Ops Manager Application server from which the Automation Agents accesses MongoDB binaries when installing a new deployment or changing the MongoDB version of an existing deployment. You must download the desired MongoDB binaries as `.tgz` archive files and place the `.tgz` files in the specified directory.

Ops Manager Backup Daemon Settings

These settings are found only in the `conf-daemon.properties` file and are necessary only if you are using Ops Manager Backup.

`rootDirectory`

Type: string

The disk partition used by the Backup Daemon to dynamically create and maintain the replica set HEAD directories. For more information on HEADs, see the *Backup functional overview*.

This directory must be writable by the mongodb-mms user and must end in a trailing slash. It is critical that this partition is sized appropriately.

Important: Data in this directory is dynamically created, maintained and destroyed by the Backup Daemon. This partition should not be used for any other purpose. This partition should *not* overlap with the partition used for the Backup Blockstore database.

`numWorkers`

Type: number

The number of replica sets that should be processed at a time.

`mongodb.release.`**`directory`**

Type: string

Specifies the full path to the directory that contains every MongoDB release needed by the Backup Daemon. When backing up a replica set, The Backup Daemon must use a `mongod` that matches the version of the replica set being backed up.

> **Warning:** If you use MongoDB Enterprise, you must **pre-install** the *MongoDB Enterprise dependencies* to all servers that run MongoDB Enterprise.

If you set `mongodb.release.autoDownload` to `false`, you must download the MongoDB releases manually. For each version needed, you must download the archive for that version and extract it into this directory. The extracted archive creates a subdirectory that uses the following naming convention: `mongodb-<platform>-<architecture>-<version>`.

Adhere to the following rules for populating the release directory:

- The release directory can contain versions from either the MongoDB Community edition or the MongoDB Enterprise but not from both.
- For the MongoDB 3.0 Community edition, do not use platform-specific archives. Instead, for all platforms that run MongoDB 3.0 Community, use the `Linux 64-bit legacy` archive from http://www.mongodb.org/downloads.

Important: If you are backing up MongoDB custom builds, you must manually place a matching binary distribution for each custom build in this directory.

Beginning in Ops Manager version 1.5, the Backup Daemon uses the following rules to match the MongoDB version of the replica set being backed up:

- If the MongoDB version uses the MongoDB standard `A.B.C` version format, the daemon looks for a folder named `mongodb-<platform>-<architecture>-A.B.x`, where `x` is greater than or equal to `C`.
- If the MongoDB version does not use the standard format, the daemon looks for a folder named `mongodb-<platform>-<architecture>-<version>`, where `<version>` ends with the MongoDB instance's version. For example, if the source version is `2.4.10-abc`, the daemon would match on `mongodb-linux-x86_64-production-2.4.10-abc`.

`mongodb.release.`**`autoDownload`**

Type: boolean

If you set this to `true`, Backup automatically downloads the latest release of MongoDB from mongodb.org/downloads and stores it in the directory specified by the `mongodb.release.directory` setting. Backup's `mongodb-fetch` utility, located in the `/opt/mongodb/backup-daemon/bin` directory, runs once an hour to perform the downloads.

If you set this to `false`, you must manually download and install the needed MongoDB releases to the directory specified in the `mongodb.release.directory` setting. Downloads must adhere to the rules described in the `mongodb.release.directory` entry above.

Advanced Backup Restore Settings

These settings affect Ops Manager Backup restore behaviors. They are are found only in the `conf-daemon.properties` file.

`mms.backup.restore.`**`linkExpirationHours`**
Type: number

Default: 1

The amount of time in hours that a restore link is available.

`mms.backup.restore.`**`linkUnlimitedUses`**
Type: boolean

Default: false

Sets whether the link to a restored point-in-time snapshot can be used more than once. By default, when you create a point-in-time snapshot, the link to download the snapshot can be used just once. To allow multiple downloads of the snapshot, set this value to `true`.

`mms.backup.restore.`**`snapshotPITExpirationHours`**
Type: number

Default: 24

The length of time in hours that a link to a restored point-in-time snapshot is available. By default, the link is available for 24 hours after creation of the point-in-time snapshot.

Session Management Setting

`mms.session.`**`maxHours`**
Type: number

The number of hours before a session on the Ops Manager website expires.

`mms.monitoring.agent.session.`**`timeoutMillis`**
Type: number

Default: 300000

Minimum: 90000

The Monitoring Agent failover time, in milliseconds. If Ops Manager does not receive a deployment status from the primary Monitoring Agent in the time specified, Ops Manager will make a standby Monitoring Agent the new primary. Configuring the timeout below *90000* (90 seconds) will cause Ops Manager to fail at startup with a configuration error.

Password Policy Settings

You can configure the password policy for Ops Manager user accounts with the following settings:

`mms.password.`**`minChangesBeforeReuse`**
Type: number

The number of previous passwords to remember. You cannot reuse a remembered password as a new password.

`mms.password.`**`maxFailedAttemptsBeforeAccountLock`**
Type: number

The number of failed login attempts before an account becomes locked. Only an an Ops Manager Administrator can unlock a locked account.

`mms.password.`**`maxDaysInactiveBeforeAccountLock`**
Type: number

The maximum number of days with no visits to the Ops Manager website before Ops Manager locks an account.

`mms.password.`**`maxDaysBeforeChangeRequired`**
Type: number

The number of days a password is valid before the password expires.

`mms.multiFactorAuth.`**`require`**
Type: boolean

Default: false

When `true`, Ops Manager will require two-factor authentication for users to log in or to perform certain destructive operations within the application.

If you configure *Twilio integration*, users may obtain their second factor tokens via Google Authenticator, SMS, or voice calls. Otherwise, the only mechanism to provide two-factor authentication is Google Authenticator.

`mms.multiFactorAuth.`**`allowReset`**
Type: boolean

Default: false

When `true`, Ops Manager will allow users to reset their two-factor authentication settings via email in an analogous fashion to resetting their passwords.

To reset two-factor authentication, a user must:

- be able to receive email at the address associated with the user account
- know the user account's password
- know the Agent API key for any Ops Manager Group of which the user is a member

`mms.multiFactorAuth.`**`issuer`**
Type: string

If Google Authenticator provides two-factor authentication, this string is the `issuer` in the Google Authenticator app. If left blank, the `issuer` is the domain name of the Ops Manager installation.

Public API

You can modify certain default behaviors of the *Public API*.

`mms.publicApi.`**`ignoreEnabledForGlobalRoles`**
Type: boolean

By default, a user with a *global role* can access any Ops Manager group through the *Public API*, whether or not the Public API is enabled for that group.

To prevent access when a group's Public API is disabled, add `mms.publicApi.ignoreEnabledForGlobalRoles` to `conf-mms.properties` and set its value to `false`:

```
mms.publicApi.ignoreEnabledForGlobalRoles=false
```

`mms.publicApi.`**`whitelistEnabled`**
Type: boolean

Certain API calls require that requests originate from a whitelisted IP address. To turn off this requirement, add `mms.publicApi.whitelistEnabled` to `conf-mms.properties` and set its value to `false`:

```
mms.publicApi.whitelistEnabled=false
```

SNMP Heartbeat Settings

Ops Manager uses SNMP v2c. You can configure the Ops Manager Application to send a periodic heartbeat trap notification (v2c) that contains an internal health assessment of the Ops Manager Application. The Ops Manager Application can send traps to one or more endpoints on the standard SNMP UDP port 162.

To configure the Ops Manager Application to send trap notifications, download the Management Information Base (MIB) file at http://downloads.mongodb.com/on-prem-monitoring/MMS-MONGODB-MIB.txt and configure the following settings:

`snmp.default.`**`hosts`**
Type: string

Default: blank

Comma-separated list of hosts where 'heartbeat' traps will be sent on the standard UDP port 162. You must set `snmp.default.hosts` to enable the SNMP heartbeat functionality; otherwise, leaving the setting blank disables the SNMP heartbeat functionality.

`snmp.listen.`**`port`**
Type: number

Default: 11611

Listening UDP port for SNMP. Setting to a number less than 1024 will require running the Ops Manager Application with root privileges.

`snmp.default.heartbeat.`**`interval`**
Type: number

Default: 300

Number of seconds between heartbeat notifications.

reCaptcha Settings

To enable reCaptcha anti-spam test on new user registration, you must have a reCaptcha account and specify the API information in the configuration file.

`reCaptcha.`**`enabled`**
Type: boolean

Set to `true` to require reCaptcha validation when a new user registers.

`reCaptcha.public.`**`key`**
Type: string

The reCaptcha public key associated with your account.

`reCaptcha.private.`**`key`**
Type: string

The reCaptcha private key associated with your account.

LDAP Settings

To configure Ops Manager for LDAP, you must start with a new installation or reset your installation to a clean state. Your Ops Manager installation cannot have existing users, groups, or hosts. For assistance, contact your MongoDB account manager.

LDAP Server Setting

`mms.userSvcClass`
Type: string

The LDAP service class `com.xgen.svc.mms.svc.user.UserSvcLdap`; i.e.

```
mms.userSvcClass=com.xgen.svc.mms.svc.user.UserSvcLdap
```

LDAP User Settings

These settings configure Ops Manager to use an LDAP server for authentication. If you use LDAP authentication, users must belong to an LDAP group to log into Ops Manager. You must create LDAP groups for each Ops Manager *user role*.

`mms.ldap.url`
Type: string

The URI for the LDAP server. For example:

```
mms.ldap.url=ldap://acme-dc1.acme.example.net:3890
```

`mms.ldap.bindDn`
Type: string

The LDAP user used to execute searches for other users. For example:

```
mms.ldap.bindDn=authUser@acme.example.net
```

`mms.ldap.bindPassword`
Type: string

The credentials for the search user. For example:

```
mms.ldap.bindPassword=<user-password>
```

`mms.ldap.user.baseDn`
Type: string

The base Distinguished Name (DN) that Ops Manager uses to search for users. Escape the = sign with \. For example:

```
mms.ldap.user.baseDn=DC\=acme,DC\=example,DC\=net
```

`mms.ldap.user.searchAttribute`
Type: string

The LDAP field used for the LDAP search. This is typically a username or email address.

The value of this field is also used as the Ops Manager username.

Example:

```
mms.ldap.user.searchAttribute=mail
```

`mms.ldap.user.`**`group`**
Type: string

The LDAP user attribute that contains the list of LDAP groups the user belongs to. The LDAP attribute can use any format to list the groups, including Common Name (`cn`) or Distinguished Name (`dn`). All Ops Manager settings in this configuration file that specify groups must match the chosen format.

Example:

```
mms.ldap.user.group=memberOf
```

`mms.ldap.global.role.`**`owner`**
Type: string

The LDAP group that has full privileges for the Ops Manager deployment, including full access to all Ops Manager *groups* and all administrative permissions. Users in the specified LDAP group receive the *global owner* role in Ops Manager. Specify the group using the format that is used by the LDAP attribute specified in the `mms.ldap.user.group` setting.

Example:

```
mms.ldap.global.role.owner=CN\=MMSGlobalOwner,OU\=MMS,OU\=acme Groups,DC\=acme,
↪DC\=example,DC\=net
```

`mms.ldap.user.`**`firstName`**
Type: string

The LDAP user attribute that contains the user's first name. After successful LDAP authentication, Ops Manager synchronizes the specified LDAP attribute with the first name from the Ops Manager user record.

For example:

```
mms.ldap.user.firstName=givenName
```

`mms.ldap.user.`**`lastName`**
Type: string

The LDAP user attribute that contains the user's last name. After successful LDAP authentication, Ops Manager synchronizes the specified LDAP attribute with the last name from the Ops Manager user record.

For example:

```
mms.ldap.user.lastName=sn
```

`mms.ldap.user.`**`email`**
Type: string

The LDAP user attribute that contains the user's email address. After successful LDAP authentication, Ops Manager synchronizes the specified LDAP attribute with the email address from the Ops Manager user record.

For example:

```
mms.ldap.user.email=mail
```

LDAP Global Role Settings

These settings assign Ops Manager *global roles* to the members of the specified LDAP groups. Specify groups using the format used by the LDAP attribute specified in the `mms.ldap.user.group` setting. You can specify multiple groups using the `;;` delimiter. To change the default delimiter, use the `mms.ldap.group.separator` setting.

Each Ops Manager global role provides its level of access to all the Ops Manager *groups* in the deployment. To provide access to specific groups, use *group-level roles*.

`mms.ldap.global.role.`**`automationAdmin`**
Type: string

The LDAP group whose members have the *global automation admin role* in Ops Manager. For example:

```
mms.ldap.global.role.automationAdmin=CN\=MMS-AutomationAdmin,OU\=MMS,OU\=acme␣
↪Groups,DC\=acme,DC\=example,DC\=net
```

`mms.ldap.global.role.`**`backupAdmin`**
Type: string

The LDAP group whose members have the *global backup admin role* in Ops Manager. For example:

```
mms.ldap.global.role.backupAdmin=CN\=MMS-BackupAdmin,OU\=MMS,OU\=acme Groups,
↪DC\=acme,DC\=example,DC\=net
```

`mms.ldap.global.role.`**`monitoringAdmin`**
Type: string

The LDAP group whose members have the *global monitoring admin role* in Ops Manager. For example:

```
mms.ldap.global.role.monitoringAdmin=CN\=MMS-MonitoringAdmin,OU\=MMS,OU\=acme␣
↪Groups,DC\=acme,DC\=example,DC\=net
```

`mms.ldap.global.role.`**`userAdmin`**
Type: string

The LDAP group whose members have the *global user admin role* in Ops Manager. For example:

```
mms.ldap.global.role.userAdmin=CN\=MMS-UserAdmin,OU\=MMS,OU\=acme Groups,DC\=acme,
↪DC\=example,DC\=net
```

`mms.ldap.global.role.`**`readOnly`**
Type: string

The LDAP group whose members have the *global read-only role* in Ops Manager. For example:

```
mms.ldap.global.role.readOnly=CN\=MMS-ReadOnly,OU\=MMS,OU\=acme Groups,DC\=acme,
↪DC\=example,DC\=net
```

`mms.ldap.group.`**`separator`**
Type: string

Each of the global role values can take a delimited list of groups: for example `"dbas,sysadmins"`.

If a group value contains the delimiter, the delimiter must be set to another value. For example, if you have the group value `"CN\=foo,DN\=bar"` and the delimiter is `,` then Ops Manager parses `"CN\=foo,DN\=bar"` as two elements rather than as the description for a single group. Change the delimiter by adding the `mms.ldap.group.separator` setting to the configuration file and specifying a different delimiter.

Starting with Ops Manager 1.5, the default delimiter is `;;`.

Kerberos Settings

To enable Kerberos authentication between the Ops Manager Application and the *Ops Manager Application Database*, configure the following settings. You must configure all required Kerberos settings to enable Kerberos authentication.

`jvm.java.security.krb5.`**`kdc`**
Type: string

Required if using Kerberos. The IP/FQDN (Fully Qualified Domain Name) of the KDC server. The value will be set to JVM's java.security.krb5.kdc.

```
jvm.java.security.krb5.kdc=kdc.example.com
```

`jvm.java.security.krb5.`**`realm`**
Type: string

Required if using Kerberos. This is the default REALM for Kerberos. It is being used for JVM's java.security.krb5.realm.

```
jvm.java.security.krb5.realm=EXAMPLE.COM
```

`mms.kerberos.`**`principal`**
Type: string

Required if using Kerberos. The principal used to authenticate with MongoDB. This should be the exact same user on the `mongo.mongoUri` above.

```
mms.kerberos.principal=mms/mmsweb.example.com@EXAMPLE.COM
```

`mms.kerberos.`**`keyTab`**
Type: string

Required if using Kerberos. The absolute path to the keytab file for the principal.

```
mms.kerberos.keyTab=/path/to/mms.keytab
```

`mms.kerberos.`**`debug`**
Type: boolean

The debug flag to output more information on Kerberos authentication process.

```
mms.kerberos.debug=false
```

Encrypt MongoDB User Credentials

If you do not want to store credentials in plain text, Monitoring provides a tool to encrypt the MongoDB credentials. To encrypt authentication credentials:

1. Issue the following command to create an encrypted credential pair, replacing `<username>` with your username:

   ```
   sudo <install_dir>/bin/credentialstool --username <username> --password
   ```

   This will prompt you to enter the password and will output the encrypted credential pair.

   `credentialstool` requires root privileges, (i.e. `sudo`) when installed with `rpm` or `deb` packages, because it modifies the `/etc/mongodb-mms/gen.key` file.

2. Use the encrypted credential pair in the `mongo.mongoUri` settings where needed, and add the `mongo.encryptedCredentials = true` setting. For example:

```
mongo.mongoUri=mongodb://da83ex3s:a4fbcf3a1@mydb1.example.net:40000/admin
mongo.encryptedCredentials=true
```

Important: The `conf-mms.properties` file can contain multiple `mongo.mongoUri` settings. If `mongo.encryptedCredentials` is `true`, you must encrypt all user credentials found in the various `mongo.mongoUri` settings.

MongoDB User Access

The MongoDB user must have the following roles: `readWriteAnyDatabase`, `clusterAdmin`, and `dbAdminAnyDatabase`. For an overview of user roles used with Ops Manager, see: *Ops Manager Roles*.

12.2 Automation Agent

Install the Agent Install the automation agent on existing hardware.

Automation Agent Configuration Documentation of the settings available in the Automation Agent configuration file.

Install the Automation Agent

The Automation Agent runs on every host that runs a monitored MongoDB deployment. The agent provides the interface for starting and managing deployments. If you run MongoDB on hardware that you provision, you must install the Automation Agent manually on each server. If you run MongoDB through the Ops Manager integration with Amazon Web Services (AWS), Ops Manager automatically deploys the Automation Agents every time you provision an new EC2 instance.

Install with RPM Packages Install and start the Automation Agent using an `rpm` package.

Install on Ubuntu Install and start the automation Agent on Ubuntu using a `deb` package.

Install on Other Linux Systems Install and start the Automation Agent on other Linux systems using the `tar.gz` archive packages.

Install on OS X Install and start the Automation Agent on OS X.

Install the Automation Agent with `rpm` Packages

On this page

Overview

The Automation Agent runs on every host that runs a monitored MongoDB deployment. The agent provides the interface for starting and managing deployments. If you run MongoDB on hardware that you provision, you must install the Automation Agent manually on each server. If you run MongoDB through the Ops Manager integration with Amazon Web Services (AWS), Ops Manager automatically deploys the Automation Agents every time you provision an new EC2 instance.

Automation Agents can run only on 64-bit architectures.

Use this procedure to install the agent on RHEL, CentOS, SUSE, Amazon Linux, and other systems that use `rpm` packages.

Prerequisites

Before installing the agent, review the *Automation Checklist* for considerations and prerequisites specific to the agent.

Procedures

This section includes procedures for both installing and updating the Automation Agent.

Install the Automation Agent with an `rpm` Package

Step 1: Download the latest version of the Automation Agent archive.

On a system shell, issue a command that resembles the following. Replace `amd64` with your platform, as needed:

```
curl -OL <OpsManagerCentralURL>/download/agent/automation/mongodb-mms-automation-
↪agent-manager-latest.x86_64.rpm
```

Step 2: Install the Automation Agent Package

```
sudo rpm -U mongodb-mms-automation-agent-manager-latest.x86_64.rpm
```

Step 3: Edit the `automation-agent.config` file.

Edit the `automation-agent.config` file.

```
sudo vi /etc/mongodb-mms/automation-agent.config
```

For `mmsGroupId`, enter your GroupID as the value. For `mmsApiKey`, enter your API key.

```
mmsGroupId=<Group ID>
mmsApiKey=<API Key>
```

For SUSE 11+ deployments only, configure the `sslTrustedMMSServerCertificate` setting. All other users should omit this step.

```
sslTrustedMMSServerCertificate=/etc/ssl/certs/UTN_USERFirst_Hardware_Root_CA.pem
```

Step 4: Prepare a directory in which to store your MongoDB data.

The directory must be owned by the `mongod` user. For example, use a set of commands similar to the following:

```
sudo mkdir /data
sudo chown mongod:mongod /data
```

Step 5: Start the Automation Agent.

Issue the following command:

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

Update the Automation Agent with an `rpm` Package

You do **not** need to stop the agent to install. The update package automatically stops, unpacks, and then restarts the agent.

Step 1: Download the latest version of the Automation Agent archive.

On a system shell, issue a command that resembles the following. Replace `amd64` with your platform, as needed:

```
curl -OL <OpsManagerCentralURL>/download/agent/automation/mongodb-mms-automation-
↪agent-manager-latest.x86_64.rpm
```

Step 2: Install the Automation Agent Package

```
sudo rpm -U mongodb-mms-automation-agent-manager-latest.x86_64.rpm
```

Step 3: Prepare a directory in which to store your MongoDB data.

The directory must be owned by the `mongod` user. For example, use a set of commands similar to the following:

```
sudo mkdir /data
sudo chown mongod:mongod /data
```

Install the Automation Agent with `deb` Packages

On this page

Overview

The Automation Agent runs on every host that runs a monitored MongoDB deployment. The agent provides the interface for starting and managing deployments. If you run MongoDB on hardware that you provision, you must install the Automation Agent manually on each server. If you run MongoDB through the Ops Manager integration with Amazon Web Services (AWS), Ops Manager automatically deploys the Automation Agents every time you provision an new EC2 instance.

Automation Agents can run only on 64-bit architectures.

Use this procedures to install the Automation Agent on Ubuntu with `deb` packages. For Debian systems, use the *Install the Automation Agent from an Archive* procedure.

Prerequisites

Before installing the agent, review the *Automation Checklist* for considerations and prerequisites specific to the agent.

Procedures

This section includes procedures for both installing and updating the Automation Agent.

Install the Automation Agent with a `deb` Package

Step 1: Download the latest version of the Automation Agent archive.

On a system shell, issue a command that resembles the following. Replace `amd64` with your platform, as needed:

```
curl -OL <OpsManagerCentralURL>/download/agent/automation/mongodb-mms-automation-
↪agent-manager_latest_amd64.deb
```

Step 2: Install the Automation Agent Package.

```
sudo dpkg -i mongodb-mms-automation-agent-manager_latest_amd64.deb
```

Step 3: Edit the `automation-agent.config` file.

Edit the `automation-agent.config` file.

```
sudo vi /etc/mongodb-mms/automation-agent.config
```

For `mmsGroupId`, enter your GroupID as the value. For `mmsApiKey`, enter your API key.

```
mmsGroupId=<Group ID>
mmsApiKey=<API Key>
```

Step 4: Prepare a directory in which to store your MongoDB data.

The directory must be owned by the `mongodb` user. For example, use a set of commands similar to the following:

```
sudo mkdir /data
sudo chown mongodb:mongodb /data
```

Step 5: Start the Automation Agent.

Issue the following command:

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

Update the Automation Agent with a `deb` Package

You do **not** need to stop the agent to install. The update package automatically stops, unpacks, and then restarts the agent.

Step 1: Download the latest version of the Automation Agent archive.

On a system shell, issue a command that resembles the following. Replace `amd64` with your platform, as needed:

```
curl -OL <OpsManagerCentralURL>/download/agent/automation/mongodb-mms-automation-
↪agent-manager_latest_amd64.deb
```

Step 2: Install the Automation Agent Package.

```
sudo dpkg -i mongodb-mms-automation-agent-manager_latest_amd64.deb
```

Step 3: Prepare a directory in which to store your MongoDB data.

The directory must be owned by the `mongodb` user. For example, use a set of commands similar to the following:

```
sudo mkdir /data
sudo chown mongodb:mongodb /data
```

Install the Automation Agent from an Archive

On this page

Overview

The Automation Agent runs on every host that runs a monitored MongoDB deployment. The agent provides the interface for starting and managing deployments. If you run MongoDB on hardware that you provision, you must install the Automation Agent manually on each server. If you run MongoDB through the Ops Manager integration with Amazon Web Services (AWS), Ops Manager automatically deploys the Automation Agents every time you provision an new EC2 instance.

Use this procedure to install the Automation Agent on a Linux system not listed in the *Agent Downloads* list on the *Agents* page in the *Administration* tab.

Automation Agents can run only on 64-bit architectures.

Prerequisites

Before installing the agent, review the *Automation Checklist* for considerations and prerequisites specific to the agent.

Procedure

This section includes procedures for both installing and updating the Automation Agent.

Install the Automation Agent from an Archive

Step 1: Download the latest version of the Automation Agent archive.

On a system shell, issue a command that resembles the following. Replace `linux_x86_64` with your platform, as needed:

```
curl -OL <OpsManagerCentralURL>/download/agent/automation/mongodb-mms-automation-
↪agent-latest.linux_x86_64.tar.gz
```

Step 2: Install the Automation Agent.

To install the agent, extract the archive using a command that resembles the following. Replace `linux_x86_64` with your platform, as needed:

```
tar -xf mongodb-mms-automation-agent-latest.linux_x86_64.tar.gz
```

The Automation Agent is installed.

Step 3: Edit the `local.config` file to include your Group ID and Ops Manager API key.

In the directory where you installed the Automation Agent, edit the `local.config` file.

- For `mmsGroupId`, enter your GroupID as the value.
- For `mmsApiKey`, enter your API key.

```
mmsGroupId=<Group ID>
mmsApiKey=<API Key>
```

Step 4: Create Automation Directory

Create the directories `/var/lib/mongodb-mms-automation`, `/var/log/mongodb-mms-automation` and `/data` and ensure that the user running the agent owns the directories.

Step 5: Start the Automation Agent.

Issue the following command:

```
nohup ./mongodb-mms-automation-agent >> automation-agent.log 2>&1 &
```

Update the Automation Agent from an Archive

Step 1: Stop any currently running Automation Agents.

Issue the following command:

```
pkill -f mongodb-mms-automation-agent
```

Step 2: Download the latest version of the Automation Agent archive.

On a system shell, issue a command that resembles the following. Replace `linux_x86_64` with your platform, as needed:

```
curl -OL <OpsManagerCentralURL>/download/agent/automation/mongodb-mms-automation-
↪agent-latest.linux_x86_64.tar.gz
```

Step 3: Install the Automation Agent.

To install the agent, extract the archive using a command that resembles the following. Replace `linux_x86_64` with your platform, as needed:

```
tar -xf mongodb-mms-automation-agent-latest.linux_x86_64.tar.gz
```

The Automation Agent is installed.

Step 4: Edit the `local.config` file to include your Group ID and Ops Manager API key.

In the directory where you installed the Automation Agent, edit the `local.config` file.

- For `mmsGroupId`, enter your GroupID as the value.
- For `mmsApiKey`, enter your API key.

```
mmsGroupId=<Group ID>
mmsApiKey=<API Key>
```

Step 5: Create Automation Directory

Create the directories `/var/lib/mongodb-mms-automation`, `/var/log/mongodb-mms-automation` and `/data` and ensure that the user running the agent owns the directories.

Step 6: Start the Automation Agent.

Issue the following command:

```
nohup ./mongodb-mms-automation-agent >> automation-agent.log 2>&1 &
```

Install the Automation Agent on OS X

On this page

Overview

The Automation Agent runs on every host that runs a monitored MongoDB deployment. The agent provides the interface for starting and managing deployments. If you run MongoDB on hardware that you provision, you must install the Automation Agent manually on each server. If you run MongoDB through the Ops Manager integration with Amazon Web Services (AWS), Ops Manager automatically deploys the Automation Agents every time you provision an new EC2 instance.

Automation Agents can run only on 64-bit architectures.

Prerequisites

Before installing the agent, review the *Automation Checklist* for considerations and prerequisites specific to the agent.

Procedure

This section includes procedures for both installing and updating the Automation Agent.

Install the Automation Agent on OS X

Step 1: Download the latest version of the Automation Agent archive.

On a system shell, issue a command that resembles the following. Replace `osx_x86_64` with your platform, as needed:

```
curl -OL <OpsManagerCentralURL>/download/agent/automation/mongodb-mms-automation-
↪agent-latest.osx_x86_64.tar.gz
```

Step 2: Install the Automation Agent.

To install the agent, extract the archive. For example:

```
tar -xf mongodb-mms-automation-agent-latest.osx_x86_64.tar.gz
```

The Automation Agent is installed.

Step 3: Edit the `local.config` file to include your Group ID and Ops Manager API key.

In the directory where you installed the Automation Agent, edit the `local.config` file.

- For `mmsGroupId`, enter your GroupID as the value.
- For `mmsApiKey`, enter your API key.

```
mmsGroupId=<Group ID>
mmsApiKey=<API Key>
```

Step 4: Create Automation Directory

Create the directories `/var/lib/mongodb-mms-automation`, `/var/log/mongodb-mms-automation` and `/data` and ensure that the user running the agent owns the directories.

Step 5: Start the Automation Agent.

Issue the following command:

```
nohup ./mongodb-mms-automation-agent --config=local.config >> /var/log/mongodb-mms-
↪automation/automation-agent.log 2>&1 &
```

Update the Automation Agent on OS X

Step 1: Stop any currently running Automation Agents.

Issue the following command:

```
pkill -f mongodb-mms-automation-agent
```

Step 2: Download the latest version of the Automation Agent archive.

On a system shell, issue a command that resembles the following. Replace `osx_x86_64` with your platform, as needed:

```
curl -OL <OpsManagerCentralURL>/download/agent/automation/mongodb-mms-automation-
↪agent-latest.osx_x86_64.tar.gz
```

Step 3: Install the Automation Agent.

To install the agent, extract the archive. For example:

```
tar -xf mongodb-mms-automation-agent-latest.osx_x86_64.tar.gz
```

The Automation Agent is installed.

Step 3: Edit the `local.config` file to include your Group ID and Ops Manager API key.

In the directory where you installed the Automation Agent, edit the `local.config` file.

- For `mmsGroupId`, enter your GroupID as the value.
- For `mmsApiKey`, enter your API key.

```
mmsGroupId=<Group ID>
mmsApiKey=<API Key>
```

Step 4: Create Automation Directory

Create the directories `/var/lib/mongodb-mms-automation`, `/var/log/mongodb-mms-automation` and `/data` and ensure that the user running the agent owns the directories.

Step 6: Start the Automation Agent.

Issue the following command:

```
nohup ./mongodb-mms-automation-agent --config=local.config >> /var/log/mongodb-mms-
↪automation/automation-agent.log 2>&1 &
```

Automation Agent Configuration

On this page

Configuration File

The location of the Automation Agent configuration file depends on your operating system:

- RHEL, CentOS, Amazon Linux, and Ubuntu all use a package manager to install the agent. The package manager creates the following agent configuration file:

```
/etc/mongodb-mms/automation-agent.config
```

- OS X, Windows, and other Linux systems use either a `tar` or `msi` file for the installation. The Automation Agent stores its configuration in the following file:

```
<installation directory>/local.config
```

Settings

Ops Manager provides default values for many of the Automation Agent Configuration settings. However, you **must** set the `mmsGroupId` and `mmsApiKey` values.

Connection Settings

`mmsGroupId`
Type: string

Required. The ID of your Ops Manager group. You can find it in Ops Manager under the *Group Settings* page in the *Administration* tab.

For example:

```
mmsGroupId=8zvbo2s2asigxvmpnkq5yexf
```

`mmsApiKey`
Type: string

Required. The Ops Manager agent API key for the group. To retrieve the key from the Ops Manager interface, click the *Administration* tab, then the *Group Settings* page.

For example:

```
mmsApiKey=rgdte4w7wwbnds9nceuodx9mcte2zqem
```

`mmsBaseUrl`
Type: string

The URL of the Ops Manager Web Server.

Set this to the URL of your Ops Manager HTTP Service. For example:

```
mmsBaseUrl=http://example.com:8080
```

`logFile`
Type: string

The path to which Ops Manager should write the automation agent's log. By default, the path is `/var/log/mongodb-mms-automation/automation-agent.log`, but you can choose an alternate location if desired.

For example:

```
logFile=/var/log/mongodb-mms-automation/automation-agent.log
```

mmsConfigBackup

Type:

The path to the file where the Automation Agent stores a backup copy of the Ops Manager *automation configuration*, which describes the desired state of the deployment.

For example:

```
mmsConfigBackup=/var/lib/mongodb-mms-automation/mms-cluster-config-backup.json
```

Logging Settings

logLevel

Type: string

The level of logging granularity. You can choose from the following severity levels, from most verbose to least. By default, `logLevel` is `INFO`.

- `DEBUG`
- `ROUTINE`
- `INFO`
- `WARN`
- `ERROR`
- `DOOM`

For example:

```
logLevel=ROUTINE
```

Each level includes the log items covered by the following levels. For instance, if you choose `DEBUG`, the Automation Agent logs all messages, including `ROUTINE`, `INFO`, `WARN`, `ERROR`, and `DOOM`. By contrast, if you choose `DOOM`, the Automation Agent only logs `DOOM` messages.

maxLogFiles

Type: integer

The maximum number of rotate log files to retain. By default, `maxLogFiles` is `10`. You can change the value to retain a different number of rotated log files. For example:

```
maxLogFiles: 15
```

maxLogFileSize

Type: integer

Specifies the maximum size, in bytes, that a log file can be before triggering log rotation. For example:

```
maxLogFileSize=536870912
```

HTTP Proxy Settings

httpProxy

Type: string

To configure the Automation Agent to use an HTTP proxy, specify the URL of the proxy. For example:

```
httpProxy=http://example-proxy.com:8080
```

12.3 Monitoring Agent

Install or Update the Agent Procedures for installing and updating the Monitoring Agent.

Monitoring Agent Configuration Documentation of the settings available in the Monitoring Agent configuration file.

Required Access for Monitoring Agent Details the permissions required for Monitoring Agent to use with MongoDB instances that enforce access control.

Configure the Agent for Access Control If MongoDB uses Access Control, create a MongoDB user for the Monitoring Agent to use to authenticate and to determine the agent's access.

Configure the Agent for SSL Configure the Monitoring Agent for SSL.

Configure Hardware Monitoring Install and configure support for a `munin-node` plugin, for hardware monitoring.

Start or Stop the Agent Procedures to start and stop the Monitoring Agent.

Remove the Agent Remove the Monitoring Agent.

Install Monitoring Agent

The Ops Manager Monitoring Agent is a lightweight component that runs within your infrastructure, connects to your MongoDB processes, collects data about the state of your deployment, and then sends the data to the Monitoring service which processes and renders this data.

Install or Update with RPM Packages Install or update the Monitoring Agent using an `rpm` package.

Install or Update on Ubuntu Install or update the Monitoring Agent on Ubuntu using a `deb` package.

Install or Update on OS X Install or update the Monitoring Agent on OS X systems.

Install or Update on Other Linux Systems Install or update the Monitoring Agent on Linux systems other than RHEL or Ubuntu.

Install or Update on Windows Install or update the Monitoring Agent on windows

Install or Update the Monitoring Agent with `rpm` Packages

On this page

Overview

The Ops Manager Monitoring Agent is a lightweight component that runs within your infrastructure, connects to your MongoDB processes, collects data about the state of your deployment, and then sends the data to the Monitoring

service which processes and renders this data. The agent initiates all connections to the Monitoring service, and communications between the agent and the Monitoring service are encrypted. A single agent can collect data from multiple MongoDB processes.

This tutorial will guide you through the steps necessary to install or update Monitoring on your system. You must install the Ops Manager Monitoring server itself before installing the Monitoring Agent.

See *Monitoring FAQs* for additional information.

Considerations

Connectivity

You must configure the networking rules of your deployment so that:

- the Monitoring Agent can connect to all `mongod` and `mongos` instances that you want to monitor.
- the Monitoring Agent can connect to Monitoring server on port 443 (i.e. `https`.)

The Monitoring server does not make *any* outbound connections to the agents or to MongoDB instances. If *Exposed DB Host Check is enabled,* the Monitoring server will attempt to connect to your servers occasionally as part of a vulnerability check.

Ensure all `mongod` and `mongos` instances are not accessible to hosts outside your deployment.

Monitoring Agent Redundancy

A single Monitoring Agent is sufficient and strongly recommended. However, you can run additional instances of the agent as hot standbys to provide redundancy. If the primary agent fails, a standby agent starts monitoring.

When you run multiple agents, only one Monitoring Agent per group or environment is the **primary agent**. The primary agent reports the cluster's status to Ops Manager. The remaining agents are completely idle, except to log their status as standby agents and to periodically ask Ops Manager whether they should become the primary.

To install additional agents, simply repeat the installation process.

Collection Interval

If you are updating the agent, keep in mind that when the Monitoring Agent restarts, there is a five-minute delay before that agent begins collecting data and sending pings to Monitoring. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Monitoring to determine which agent will be primary.

During this interval, the restarted Monitoring Agent will not collect data.

Prerequisites

If your MongoDB deployment enforces access control, you must create a user in MongoDB with the appropriate access. See *Configure Monitoring Agent for Access Control*.

Procedures

This section includes procedures for both installing and updating the Monitoring Agent.

Install the Monitoring Agent with an `rpm` Package

Use this procedure to install the agent on RHEL, CentOS, SUSE, Amazon Linux, and other systems that use `rpm` packages.

Step 1: Download the latest version of the Monitoring Agent package.

In a system shell, issue the following command:

```
curl -OL <mmsUri>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.x86_
↪64.rpm
```

Step 2: Install the Monitoring Agent package.

Issue the following command:

```
sudo rpm -U mongodb-mms-monitoring-agent-latest.x86_64.rpm
```

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, click the link for your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `monitoring-agent.config` file to include your Ops Manager API key.

In the `<install-directory>/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

Step 5: Optional: For SUSE deployments only, configure the `sslTrustedMMSServerCertificate` property.

If you're deploying on SUSE, you must configure the `sslTrustedMMSServerCertificate` setting. All other users should omit this step.

Enter the following property and value in the `/etc/mongodb-mms/monitoring-agent.config` file:

```
sslTrustedMMSServerCertificate=/etc/ssl/certs/UTN_USERFirst_Hardware_Root_CA.pem
```

Save and close the file.

Step 6: Start the Monitoring Agent.

Issue the following command:

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

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Update the Monitoring Agent with an `rpm` Package

Step 1: Download the latest version of the Monitoring Agent package.

In a system shell, issue the following command:

```
curl -OL <mmsUri>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.x86_
↪64.rpm
```

Step 2: Install the Monitoring Agent package.

Issue the following command:

```
sudo rpm -U mongodb-mms-monitoring-agent-latest.x86_64.rpm
```

Install or Update the Monitoring Agent with `deb` Packages

On this page

Overview

The Ops Manager Monitoring Agent is a lightweight component that runs within your infrastructure, connects to your MongoDB processes, collects data about the state of your deployment, and then sends the data to the Monitoring service which processes and renders this data. The agent initiates all connections to the Monitoring service, and communications between the agent and the Monitoring service are encrypted. A single agent can collect data from multiple MongoDB processes.

This tutorial will guide you through the steps necessary to install or update Monitoring on your system. You must install the Ops Manager Monitoring server itself before installing the Monitoring Agent.

See *Monitoring FAQs* for additional information.

Considerations

Connectivity

You must configure the networking rules of your deployment so that:

- the Monitoring Agent can connect to all `mongod` and `mongos` instances that you want to monitor.
- the Monitoring Agent can connect to Monitoring server on port 443 (i.e. `https`.)

The Monitoring server does not make *any* outbound connections to the agents or to MongoDB instances. If *Exposed DB Host Check is enabled,* the Monitoring server will attempt to connect to your servers occasionally as part of a vulnerability check.

Ensure all `mongod` and `mongos` instances are not accessible to hosts outside your deployment.

Monitoring Agent Redundancy

A single Monitoring Agent is sufficient and strongly recommended. However, you can run additional instances of the agent as hot standbys to provide redundancy. If the primary agent fails, a standby agent starts monitoring.

When you run multiple agents, only one Monitoring Agent per group or environment is the **primary agent**. The primary agent reports the cluster's status to Ops Manager. The remaining agents are completely idle, except to log their status as standby agents and to periodically ask Ops Manager whether they should become the primary.

To install additional agents, simply repeat the installation process.

Collection Interval

If you are updating the agent, keep in mind that when the Monitoring Agent restarts, there is a five-minute delay before that agent begins collecting data and sending pings to Monitoring. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Monitoring to determine which agent will be primary.

During this interval, the restarted Monitoring Agent will not collect data.

Prerequisites

If your MongoDB deployment enforces access control, you must create a user in MongoDB with the appropriate access. See *Configure Monitoring Agent for Access Control*.

Procedures

This section includes procedures for installing and updating the Monitoring Agent on Ubuntu with `deb` packages.

For Debian systems, instead use the *Install or Update the Monitoring Agent from Archive* procedure.

Install the Monitoring Agent with a `deb` Package

Step 1: Download the latest version of the Monitoring Agent package.

Issue the following command using the system shell:

```
curl -OL <mmsUri>/download/agent/monitoring/mongodb-mms-monitoring-agent_latest_amd64.
↪deb
```

Step 2: Install the Monitoring Agent package.

Issue the following command using the system shell:

```
sudo dpkg -i mongodb-mms-monitoring-agent_latest_amd64.deb
```

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, click the link for your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `monitoring-agent.config` file to include your Ops Manager API key.

In the `<install-directory>/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

Step 5: Start the Monitoring Agent.

Issue the following command:

```
sudo start mongodb-mms-monitoring-agent
```

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Update the Monitoring Agent with a `deb` Package

Step 1: Download the latest version of the Monitoring Agent package.

Issue the following command using the system shell:

```
curl -OL <mmsUri>/download/agent/monitoring/mongodb-mms-monitoring-agent_latest_amd64.
↪deb
```

Step 2: Install the Monitoring Agent package.

Issue the following command using the system shell:

```
sudo dpkg -i mongodb-mms-monitoring-agent_latest_amd64.deb
```

Step 3: Start the Monitoring Agent.

Issue the following command:

```
sudo start mongodb-mms-monitoring-agent
```

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Install or Update the Monitoring Agent on OS X

On this page

Overview

The Ops Manager Monitoring Agent is a lightweight component that runs within your infrastructure, connects to your MongoDB processes, collects data about the state of your deployment, and then sends the data to the Monitoring service which processes and renders this data. The agent initiates all connections to the Monitoring service, and communications between the agent and the Monitoring service are encrypted. A single agent can collect data from multiple MongoDB processes.

This tutorial will guide you through the steps necessary to install or update Monitoring on your system. You must install the Ops Manager Monitoring server itself before installing the Monitoring Agent.

See *Monitoring FAQs* for additional information.

Considerations

Connectivity

You must configure the networking rules of your deployment so that:

- the Monitoring Agent can connect to all `mongod` and `mongos` instances that you want to monitor.
- the Monitoring Agent can connect to Monitoring server on port 443 (i.e. `https`.)

The Monitoring server does not make *any* outbound connections to the agents or to MongoDB instances. If *Exposed DB Host Check is enabled,* the Monitoring server will attempt to connect to your servers occasionally as part of a vulnerability check.

Ensure all `mongod` and `mongos` instances are not accessible to hosts outside your deployment.

Monitoring Agent Redundancy

A single Monitoring Agent is sufficient and strongly recommended. However, you can run additional instances of the agent as hot standbys to provide redundancy. If the primary agent fails, a standby agent starts monitoring.

When you run multiple agents, only one Monitoring Agent per group or environment is the **primary agent**. The primary agent reports the cluster's status to Ops Manager. The remaining agents are completely idle, except to log their status as standby agents and to periodically ask Ops Manager whether they should become the primary.

To install additional agents, simply repeat the installation process.

Collection Interval

If you are updating the agent, keep in mind that when the Monitoring Agent restarts, there is a five-minute delay before that agent begins collecting data and sending pings to Monitoring. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Monitoring to determine which agent will be primary.

During this interval, the restarted Monitoring Agent will not collect data.

Prerequisites

If your MongoDB deployment enforces access control, you must create a user in MongoDB with the appropriate access. See *Configure Monitoring Agent for Access Control*.

Procedures

This section includes procedures for both installing and updating the Monitoring Agent.

Install the Monitoring Agent on OS X

Use this procedure to install the agent OS X systems.

Step 1: Download the latest version of the Monitoring Agent archive.

In a system shell, issue the following command:

```
curl -OL <mmsUri>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.osx_
↪x86_64.tar.gz
```

Step 2: Install the Monitoring Agent.

To install the agent, extract the archive by issue the following command:

```
tar -xf mongodb-mms-monitoring-agent-latest.osx_x86_64.tar.gz
```

The Monitoring Agent is installed.

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, click the link for your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `monitoring-agent.config` file to include your Ops Manager API key.

In the `<install-directory>/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

Step 5: Optional: Configure the agent to use a proxy server.

If the agent will connect to Ops Manager via a proxy server, you must specify the server in the `http_proxy` environment variable. To specify the server, use the `export` command, as in the following example:

```
export http_proxy="http://proxy.example.com:9000"
```

To connect through a proxy, you must install the agent from a `.tar.gz` file, *not* from a `.deb` or `.rpm` file.

Step 6: Start the Monitoring Agent.

Issue the following command:

```
nohup ./mongodb-mms-monitoring-agent >> monitoring-agent.log 2>&1 &
```

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Update the Monitoring Agent from a `tar.gz` Archive

Use this procedure to update the agent on OS X systems.

Step 1: Stop any currently running Monitoring Agents.

Issue the following command:

```
pkill -f mongodb-mms-monitoring-agent
```

Step 2: Download the latest version of the Monitoring Agent archive.

In a system shell, issue the following command:

```
curl -OL <mmsUri>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.osx_
↪x86_64.tar.gz
```

Step 3: Install the Monitoring Agent.

To install the agent, extract the archive by issue the following command:

```
tar -xf mongodb-mms-monitoring-agent-latest.osx_x86_64.tar.gz
```

The Monitoring Agent is installed.

Step 4: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, click the link for your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 5: Edit the `monitoring-agent.config` file to include your Ops Manager API key.

In the `<install-directory>/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

Step 6: Start the Monitoring Agent.

Issue the following command:

```
nohup ./mongodb-mms-monitoring-agent >> monitoring-agent.log 2>&1 &
```

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Install or Update the Monitoring Agent from Archive

On this page

Overview

The Ops Manager Monitoring Agent is a lightweight component that runs within your infrastructure, connects to your MongoDB processes, collects data about the state of your deployment, and then sends the data to the Monitoring service which processes and renders this data. The agent initiates all connections to the Monitoring service, and communications between the agent and the Monitoring service are encrypted. A single agent can collect data from multiple MongoDB processes.

This tutorial will guide you through the steps necessary to install or update Monitoring on your system. You must install the Ops Manager Monitoring server itself before installing the Monitoring Agent.

See *Monitoring FAQs* for additional information.

Considerations

Connectivity

You must configure the networking rules of your deployment so that:

- the Monitoring Agent can connect to all `mongod` and `mongos` instances that you want to monitor.
- the Monitoring Agent can connect to Monitoring server on port 443 (i.e. `https`.)

The Monitoring server does not make *any* outbound connections to the agents or to MongoDB instances. If *Exposed DB Host Check is enabled,* the Monitoring server will attempt to connect to your servers occasionally as part of a vulnerability check.

Ensure all `mongod` and `mongos` instances are not accessible to hosts outside your deployment.

Monitoring Agent Redundancy

A single Monitoring Agent is sufficient and strongly recommended. However, you can run additional instances of the agent as hot standbys to provide redundancy. If the primary agent fails, a standby agent starts monitoring.

When you run multiple agents, only one Monitoring Agent per group or environment is the **primary agent**. The primary agent reports the cluster's status to Ops Manager. The remaining agents are completely idle, except to log their status as standby agents and to periodically ask Ops Manager whether they should become the primary.

To install additional agents, simply repeat the installation process.

Collection Interval

If you are updating the agent, keep in mind that when the Monitoring Agent restarts, there is a five-minute delay before that agent begins collecting data and sending pings to Monitoring. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Monitoring to determine which agent will be primary.

During this interval, the restarted Monitoring Agent will not collect data.

Prerequisites

If your MongoDB deployment enforces access control, you must create a user in MongoDB with the appropriate access. See *Configure Monitoring Agent for Access Control*.

Procedures

This section includes procedures for installing and updating the Monitoring Agent on a Linux system not listed in the *Agent Downloads* list on the *Agents* page in the *Administration* tab.

Install the Monitoring Agent from a `tar.gz` Archive

Use this procedure to install the agent on Linux systems:

Step 1: Download the latest version of the Monitoring Agent archive.

With a system shell, issue the following command:

```
curl -OL <mmsUri>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.linux_
↪x86_64.tar.gz
```

Step 2: Install the Monitoring Agent.

To install the agent, extract the archive by issue the following command:

```
tar -xf mongodb-mms-monitoring-agent-latest.linux_x86_64.tar.gz
```

The Monitoring Agent is installed.

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, click the link for your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `monitoring-agent.config` file to include your Ops Manager API key.

In the `<install-directory>/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

Step 5: Optional: Configure the agent to use a proxy server.

If the agent will connect to Ops Manager via a proxy server, you must specify the server in the `http_proxy` environment variable. To specify the server, use the `export` command, as in the following example:

```
export http_proxy="http://proxy.example.com:9000"
```

To connect through a proxy, you must install the agent from a `.tar.gz` file, *not* from a `.deb` or `.rpm` file.

Step 6: Start the Monitoring Agent.

Issue the following command:

```
nohup ./mongodb-mms-monitoring-agent >> monitoring-agent.log 2>&1 &
```

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Update the Monitoring Agent from a `tar.gz` Archive

Use this procedure to update the agent on Linux systems:

Step 1: Stop any currently running Monitoring Agents.

Issue the following command:

```
pkill -f mongodb-mms-monitoring-agent
```

Step 2: Download the latest version of the Monitoring Agent archive.

With a system shell, issue the following command:

```
curl -OL <mmsUri>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.linux_
↪x86_64.tar.gz
```

Step 3: Install the Monitoring Agent.

To install the agent, extract the archive by issue the following command:

```
tar -xf mongodb-mms-monitoring-agent-latest.linux_x86_64.tar.gz
```

The Monitoring Agent is installed.

Step 4: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, click the link for your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 5: Edit the `monitoring-agent.config` file to include your Ops Manager API key.

In the `<install-directory>/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

Step 6: Start the Monitoring Agent.

Issue the following command:

```
nohup ./mongodb-mms-monitoring-agent >> monitoring-agent.log 2>&1 &
```

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Additional Information

If you installed the Monitoring Agent from the `tar.gz` archives, see /tutorial/rotate-agent-log-files to configure log rotation.

Install or Update the Monitoring Agent on Windows

On this page

- *Prerequisites*
- *Procedures*

Overview

The Ops Manager Monitoring Agent is a lightweight component that runs within your infrastructure, connects to your MongoDB processes, collects data about the state of your deployment, and then sends the data to the Monitoring service which processes and renders this data. The agent initiates all connections to the Monitoring service, and communications between the agent and the Monitoring service are encrypted. A single agent can collect data from multiple MongoDB processes.

This tutorial will guide you through the steps necessary to install or update Monitoring on your system. You must install the Ops Manager Monitoring server itself before installing the Monitoring Agent.

See *Monitoring FAQs* for additional information.

Considerations

Connectivity

You must configure the networking rules of your deployment so that:

- the Monitoring Agent can connect to all `mongod` and `mongos` instances that you want to monitor.
- the Monitoring Agent can connect to Monitoring server on port 443 (i.e. `https`.)

The Monitoring server does not make *any* outbound connections to the agents or to MongoDB instances. If *Exposed DB Host Check is enabled,* the Monitoring server will attempt to connect to your servers occasionally as part of a vulnerability check.

Ensure all `mongod` and `mongos` instances are not accessible to hosts outside your deployment.

Monitoring Agent Redundancy

A single Monitoring Agent is sufficient and strongly recommended. However, you can run additional instances of the agent as hot standbys to provide redundancy. If the primary agent fails, a standby agent starts monitoring.

When you run multiple agents, only one Monitoring Agent per group or environment is the **primary agent**. The primary agent reports the cluster's status to Ops Manager. The remaining agents are completely idle, except to log their status as standby agents and to periodically ask Ops Manager whether they should become the primary.

To install additional agents, simply repeat the installation process.

Collection Interval

If you are updating the agent, keep in mind that when the Monitoring Agent restarts, there is a five-minute delay before that agent begins collecting data and sending pings to Monitoring. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Monitoring to determine which agent will be primary.

During this interval, the restarted Monitoring Agent will not collect data.

Prerequisites

If your MongoDB deployment enforces access control, you must create a user in MongoDB with the appropriate access. See *Configure Monitoring Agent for Access Control*.

Procedures

This section includes procedures for both installing and updating the Monitoring Agent.

Install the Monitoring Agent on Windows

Use this procedure to install the agent on Windows.

Step 1: Download and run the latest version of the Monitoring Agent MSI file.

Download and run the 32-bit or 64-bit MSI file. During installation, the installer prompts you to specify the folder for storing configuration and log files. It is strongly advised that you encrypt or restrict access to this folder.

To download the 32-bit MSI file, use the following URL, where `<mms-server>` is the hostname of the Monitoring server:

```
<mms-server>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.windows_
↪i386.msi
```

To download the 64-bit MSI file, use the following URL, where `<mms-server>` is the hostname of the Monitoring server:

```
<mms-server>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.windows_
↪x86_64.msi
```

During installation, the installer prompts you to specify the folder for storing configuration and log files. It is strongly advised that you encrypt or restrict access to this folder.

Step 2: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, click the link for your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 3: Edit the `monitoring-agent.config` file to include your Ops Manager API key.

In the `<install-directory>/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

The default location for the agent configuration file is `C:\MMSData\Monitoring\monitoring-agent.config`.

Step 4: Edit the `monitoring-agent.config` file to include the hostname of the Monitoring server.

Set the `mmsBaseUrl` property to the hostname of the Monitoring server.

Step 5: Start the Monitoring Agent.

Issue the following command:

In Windows `Control Panel`, open `Administrative Tools`, and then open `Services`.

In the list of services, select the Ops Manager Monitoring Agent service. Select the `Action` menu and select `Start`.

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Update the Monitoring Agent on Windows

To update the agent on Windows systems:

Step 1: Stop any currently running Monitoring Agents.

In Windows `Control Panel`, open `Administrative Tools` and then `Services`.

In the list of services, select the Ops Manager Monitoring Agent service. Select the `Action` menu and select `Stop`.

Step 2: Download and run the latest version of the Monitoring Agent MSI file.

Download and run the 32-bit or 64-bit MSI file. During installation, the installer prompts you to specify the folder for storing configuration and log files. It is strongly advised that you encrypt or restrict access to this folder.

To download the 32-bit MSI file, use the following URL, where `<mms-server>` is the hostname of the Monitoring server:

```
<mms-server>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.windows_
↪i386.msi
```

To download the 64-bit MSI file, use the following URL, where `<mms-server>` is the hostname of the Monitoring server:

```
<mms-server>/download/agent/monitoring/mongodb-mms-monitoring-agent-latest.windows_
↪x86_64.msi
```

During installation, the installer prompts you to specify the folder for storing configuration and log files. It is strongly advised that you encrypt or restrict access to this folder.

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, click the link for your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `monitoring-agent.config` file to include your Ops Manager API key.

In the `<install-directory>/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

Step 5: Start the Monitoring Agent.

In Windows `Control Panel`, open `Administrative Tools`, and then open `Services`.

In the list of services, select the Ops Manager Monitoring Agent service. Select the `Action` menu and select `Start`.

Remember, that you only need to run **1** Monitoring Agent for each Ops Manager group. A single Monitoring Agent can collect data from many MongoDB instances.

Monitoring Agent Configuration

On this page

Warning: Do **not** edit these settings for a Monitoring Agent that is managed by an Automation Agent. If you do, the Automation Agent will overwrite any changes you make.

Configuration File

The location of the Monitoring Agent configuration file depends on your operating system:

- RHEL, CentOS, Amazon Linux, and Ubuntu all use a package manager to install the agent. The package manager creates the following agent configuration file:

  `/etc/mongodb-mms/monitoring-agent.config`

- OS X, Windows, and other Linux systems use either a `tar` or `msi` file for the installation. The Monitoring Agent stores its configuration in the following file:

  `<installation directory>/monitoring-agent.config`

Settings

Connection Settings

For the Monitoring Agent communication with the Ops Manager servers, the following connection settings are **required**:

`mmsApiKey`
Type: string

The Ops Manager agent API key for a Ops Manager group. To retrieve the key from the Ops Manager interface, click the *Administration* tab, then the *Agents* page, and then the link for your operating system. Ops Manager will display the Ops Manager API key used by your Ops Manager group.

For example:

```
mmsApiKey=abc123
```

mmsBaseUrl

Type: string

The URL of the Ops Manager Web Server.

Set this to the URL of your Ops Manager HTTP Service. For example:

```
mmsBaseUrl=http://example.com:8080
```

HTTP Proxy Settings

httpProxy

New in version 2.3.1.

Type: string

To connect to the Ops Manager HTTP Service via a proxy, specify the URL of the proxy. For example:

```
httpProxy=http://example-proxy.com:8080
```

MongoDB SSL Settings

Specify these settings when the Monitoring Agent is connecting to MongoDB instances with SSL.

useSslForAllConnections

Type: boolean

Set to `true` to enable SSL support globally and to use SSL for all MongoDB connections. Setting this to `true` overrides any per-host SSL settings configured in the Ops Manager interface.

When `true`, use `useSslForAllConnections` with the `sslTrustedServerCertificates` setting to specify the certificates that Ops Manager should accept.

Note: If `useSslForAllConnections` is `true` **and** you set `sslRequireValidServerCertificates` to `false`, Ops Manager will accept any connection regardless of the certificate provided. This is only recommended for testing purposes as it makes connections susceptible to man-in-the-middle attacks.

---

sslClientCertificate

Type: string

The path to the private key, client certificate, and optional intermediate certificates in PEM format. The agent will use the client certificate when connecting to any configured MongoDB that uses SSL and requires a client certificate, i.e., that is running using the `--sslCAFile` option.

For example, if you would use the following command to connect through the `mongo` shell to a MongoDB process that uses both SSL and certificate validation:

```
mongo --ssl --sslPEMKeyFile /etc/ssl/client.pem --sslCAFile /etc/ssl/ca.pem␣
↪example.net:27017
```

Then set the following in your Monitoring Agent configuration file:

```
sslTrustedServerCertificates=/etc/ssl/ca.pem
sslClientCertificate=/etc/ssl/client.pem
```

sslClientCertificatePassword

Type: string

The password needed to decrypt the private key in the file specified in `sslClientCertificate`. This setting is necessary only if the client certificate PEM file is encrypted.

sslTrustedServerCertificates

Type: string

The path on disk that contains the trusted certificate authority certificates in PEM format. These certificates will verify the server certificate returned from any MongoDB instances running with SSL. For example:

```
sslTrustedServerCertificates=/etc/ssl/ca.pem
```

sslRequireValidServerCertificates

Type: boolean

Use this option to disable certificate verification by setting this value to `false`. That configuration is only recommended for testing purposes as it makes connections susceptible to man-in-the-middle attacks.

MongoDB Kerberos Settings

See *Configure the Monitoring Agent for Kerberos*

krb5Principal

Type: string

The Kerberos principal used by the agent. For example:

```
krb5Principal=mmsagent/myhost@EXAMPLE.COM
```

krb5Keytab

Type: string

The *absolute* path to Kerberos principal's keytab file. For example:

```
krb5Keytab=/etc/mongodb-mms/mms-monitoring-agent.keytab
```

gsappiServiceName

Type: string

The default service name used by MongoDB is `mongodb` can specify a custom service name with the `gssapiServiceName` option.

Ops Manager Server SSL Settings

Advanced SSL settings used by the Monitoring Agent when communicating to the Ops Manager HTTP Service.

sslTrustedMMSServerCertificate

By default the Monitoring Agent will use the trusted root CAs installed on the system. If the agent cannot find the trusted root CAs, configure these settings manually.

If the Ops Manager HTTP Service uses a self-signed SSL certificate, you *must* specify `sslTrustedMMSServerCertificate`.

The path on disk that contains the trusted certificate authority certificates in PEM format. The agent will use this certificate to verify that the agent is communicating with the designated Ops Manager HTTP Service. For example:

```
sslTrustedMMSServerCertificate=/etc/mongodb-mms/mms-certs.pem
```

sslRequireValidMMSServerCertificates
Type: boolean

You can disable certificate verification by setting this value to `false`. That configuration is only recommended for testing purposes as it makes connections susceptible to *man-in-the-middle* attacks.

Munin Settings

See *Configure Hardware Monitoring with munin-node* for information on configuring Munin-node.

enableMunin
Type: boolean

Set to `false` if you do not with the Monitoring Agent to collect hardware statistics via Munin-node. The default is `true`. If the agent detects `munin-node`, Ops Manager will collect hardware statistics.

Deprecated Settings

MongoDB Authentication Settings

If all monitored MongoDB instances use the same MONGODB-CR credentials, you may use these settings. Setting the username and password here will override any configuration in the Ops Manager UI.

See *Required Access for Monitoring Agent* for information on the privileges needed for this user.

globalAuthUsername
Type: string

The MongoDB username that the Monitoring Agent will use to connect. **This value overrides all other usernames configured for the Monitoring Agent.**

Example:

```
globalAuthUsername=mms-monitoring-agent
```

globalAuthPassword
Type: string

The password for the `globalAuthUsername` user. **This value overrides all other passwords configured for the Monitoring Agent.**

Example:

```
globalAuthPassword=somePassword
```

Required Access for Monitoring Agent

On this page

- *MongoDB 2.6*
- *MongoDB 2.4*
- *Authentication Mechanisms*

If your MongoDB deployment enforces access control, the Ops Manager Monitoring Agent must authenticate to MongoDB as a user with the proper access.

To authenticate, create a user with the appropriate roles in MongoDB. The following tutorials include instructions and examples for creating the MongoDB user:

- *Add Monitoring Agent User for MONGODB-CR*.
- *Configure Monitoring Agent for LDAP*.
- *Configure the Monitoring Agent for Kerberos*.

MongoDB user roles are separate from Ops Manager *user roles* and are described in the MongoDB manual beginning with the Authorization page.

Considerations

To authenticate to sharded clusters, create shard-local users on *each* shard and create cluster-wide users:

- Create cluster users while connected to the `mongos`; these credentials persist to the config servers.
- Create shard-local users by connecting directly to the replica set for each shard.

There are additional authentication configuration requirements for Ops Manager Monitoring when using MongoDB 2.4 with authentication.

Prerequisites

Connect to the `mongod` or `mongos` instance as a user with access to create users in the database. See db.createUser() method page for more information.

MongoDB 2.6

To monitor MongoDB 2.6 instances, including `dbStats`[2] and database profiling information[1], the monitoring agent must authenticate to the database as a user with the following access:

Required Role	
`clusterMonitor` role on the `admin` database	

For *mixed* MongoDB versions, the specified access is inadequate to monitor deployments of since the user cannot access the `local` database needed for mixed deployments. Monitoring a mixed deployment as a user with the specified access will produce an authorization error that will appear in the `mongod` logs.

The monitoring agent can recover from this error, and you may safely ignore these messages in the `mongod` log.

[2] Monitoring without `dbStats` excludes database storage, records, indexes, and other statistics.
[1] Profiling captures in-progress read and write operations, cursor operations, and database command information about the database.

MongoDB 2.4

Monitor without Database Profiling

To monitor MongoDB 2.4 instances, including `dbStats` operations, the agent must authenticate as a user with the following access:

Required Roles	
`clusterAdmin` role on the `admin` database	
`readAnyDatabase` role on the `admin` database	

However, a user with the specified access *cannot* monitor with profiling. If this user tries to monitor with profiling, the `mongod` log file may report the following message at the default logging level:

```
command denied: { profile: -1 }
```

You can ignore this message if you do not want Ops Manager to collect profile data. If you want to collect profile data, configure Ops Manager monitoring as specified in *Monitor with Database Profiling*.

Monitor with Database Profiling

To monitor MongoDB 2.4 databases with database profiling[1], the agent must authenticate as a user with the following access:

Required Roles	
`clusterAdmin` role on the `admin` database	
`readAnyDatabase` role on the `admin` database	
`dbAdminAnyDatabase` roles in the `admin` database	

Monitor without `dbStats`

To monitor MongoDB 2.4 databases *without* `dbStats`[2], the agent must authenticate as a user with the following access:

Required Role	
`clusterAdmin` role on the `admin` database	

Authentication Mechanisms

To authenticate, create the user in MongoDB with the appropriate access. The authentication method that the MongoDB deployment uses determines how to create the user as well as determine any additional agent configuration:

- For MONGODB-CR (MongoDB Challenge-Response) authentication, see *Configure Backup Agent for MONGODB-CR*.
- For LDAP authentication, see *Configure Backup Agent for LDAP Authentication*.
- For Kerberos authentication, see *Configure the Backup Agent for Kerberos*.

Configure Monitoring Agent for Access Control

If your MongoDB deployment enforces access control, the Monitoring Agent must authenticate to MongoDB as a user with the proper access.

Configure for MONGODB-CR Procedure to configure the Monitoring Agent for MongoDB deployments using MongoDB Challenge and Response authentication.

Configure for LDAP Procedure to configure the Monitoring Agent for MongoDB deployments using LDAP authentication.

Configure for Kerberos Procedure to configure the Monitoring Agent for MongoDB deployments using Kerberos authentication.

Add Monitoring Agent User for MONGODB-CR

On this page

Ops Manager Monitoring Agent can use the MongoDB Challenge-Response (`MONGODB-CR`) to authenticate to hosts that enforce access control.

To authenticate using `MONGODB-CR`, create a user in the `admin` database with the appropriate roles in MongoDB.

Prerequisites

There are additional authentication configuration requirements for Ops Manager Monitoring when using MongoDB 2.4 with authentication. See *Required Access for Monitoring Agent* for more information.

MongoDB 2.6

To monitor MongoDB 2.6 instances, create a user in the `admin` database with an operation that resembles the following:

```
use admin
db.createUser(
   {
     user: "<username>",
     pwd: "<password>",
     roles: [ { role: "clusterMonitor", db: "admin" } ]
   }
)
```

See *Access Control for MongoDB 2.6* for more information on the required access.

MongoDB 2.4

To monitor MongoDB 2.4 instances, create a user in the `admin` database with an operation that resembles the following:

```
use admin
db.addUser(
   {
     user: "<username>",
     pwd: "<password>",
     roles: [
        "clusterAdmin",
        "readAnyDatabase"
     ]
   }
)
```

See *Access Control for MongoDB 2.4 without Profiling* for more information on the required access.

Monitor with Database Profiling

To monitor MongoDB 2.4 databases with database profiling, create a user in the `admin` database with an operation that resembles the following:

```
use admin
db.addUser(
   {
     user: "<username>",
     pwd: "<password>",
     roles: [
        "clusterAdmin",
        "readAnyDatabase",
        "dbAdminAnyDatabase"
     ]
    }
)
```

See *Access Control for MongoDB 2.4 with Profiling* for more information on the required access.

Monitor without `dbStats`

To monitor MongoDB 2.4 instance *without* `dbStats`, create a user in the `admin` database with an operation that resembles the following:

```
use admin
db.addUser(
   {
     user: "<username>",
     pwd: "<password>",
     roles: [ "clusterAdmin" ]
   }
)
```

See *Access Control for MongoDB 2.4 without dbStats* for more information on the required access.

Host Settings

In addition to adding the agent as a MongoDB user, you must also specify the host's authentication settings. You can specify the host's authentication settings when *adding* the host, or you can *edit the settings* for an existing host.

Configure Monitoring Agent for LDAP

On this page

Overview

If your MongoDB deployment enforces access control, the Monitoring Agent must authenticate to MongoDB as a user with the proper access.

LDAP is a standard protocol for accessing user credential data. Starting in version 2.6, MongoDB Enterprise provides an *LDAP (plain)* authentication mechanism that allows clients to authenticate to MongoDB deployments using LDAP. Monitoring Agents support authenticating to MongoDB instances using LDAP.

If your MongoDB deployment uses LDAP to authenticate users, to authenticate the Monitoring Agent, create a user in the `$external` database with the appropriate roles in MongoDB.

Considerations

You must configure LDAP authentication separately for the Monitoring Agent and for the Backup Agent.

You can configure LDAP authentication when adding a host or later by editing the host.

Prerequisites

There are additional authentication configuration requirements for Ops Manager Monitoring when using MongoDB 2.4 with authentication. See *Required Access for Monitoring Agent* for more information.

Create User in MongoDB

To monitor MongoDB 2.6 instances that are using LDAP authentication, add a user to the `$external` database in MongoDB with the appropriate roles. The `$external` database allows `mongod` to consult an external source (e.g. LDAP) to authenticate.

```
db.getSiblingDB("$external").createUser(
    {
      user : "<username>",
      roles: [ { role: "clusterMonitor", db: "admin" } ]
    }
)
```

See *Access Control for MongoDB 2.6* for more information on the required access.

Host Settings

In addition to adding the agent as a MongoDB user, you must also specify the host's authentication settings. You can specify the host's authentication settings when *adding* the host, or you can *edit the settings* for an existing host.

Configure the Monitoring Agent for Kerberos

On this page

MongoDB Enterprise provides support for Kerberos. Kerberos is a generic authentication protocol available starting from MongoDB Enterprise version 2.6. The Monitoring Agents can authenticate to hosts using Kerberos.

Warning: You must install the *prerequisite packages* on your servers before deploying MongoDB Enterprise on the servers.

You can use Ops Manager for Monitoring and Backup with Kerberos, but you cannot currently use Automation with Kerberos. Automation will support these features in the next major Ops Manager release.

Prerequisites

You must configure the Kerberos Key Distribution Center (KDC) to grant tickets that are valid for at least four hours. The Monitoring Agent takes care of periodically renewing the ticket. The KDC service provides session tickets and temporary session keys to users and computers.

There are additional authentication configuration requirements for Ops Manager Monitoring when using MongoDB 2.4 with authentication. See *Required Access for Monitoring Agent* for more information.

Create Kerberos Principal

If you are running both the Monitoring Agent and the Backup Agent on the same server, then both agents must connect as the same Kerberos Principal.

Step 1: Create or choose a Kerberos principal.

Create or choose a Kerberos principal for the Monitoring and/or Backup agent.

Step 2: Generate a keytab for the Kerberos principal.

Generate a keytab for the Kerberos principal and copy it to the system where the agent runs. Ensure the user that will run the agent is the same user that owns the keytab file.

Create MongoDB User for the Principal

If you are running both the Monitoring Agent and the Backup Agent on the same server, then both agents must connect as the same Kerberos Principal.

Add a Kerberos principal, `<username>@<KERBEROS REALM>` or `<username>/<instance>@<KERBEROS REALM>`, to MongoDB in the `$external` database. Specify the Kerberos realm in all uppercase. The `$external` database allows `mongod` to consult an external source (e.g. Kerberos) to authenticate.

Kerberos Principal for the Monitoring Agent

For example, to add the principal for just the Monitoring Agent:

```
use $external
db.createUser(
   {
     user: "<Kerberos Principal>",
     roles: [ { role: "clusterMonitor", db: "admin" } ]
   }
)
```

See *MongoDB 2.6* for more information on the required access for the Monitoring Agent.

Kerberos Principal for the Monitoring Agent and Backup Agent

For example, to add the same principal for both the Monitoring Agent and the Backup Agent, specify required access for both agents. The following example specifies access required to connect to MongoDB 3.0 or greater.

```
use $external
db.createUser(
   {
     user: "<Kerberos Principal>",
     roles: [
        { role: "clusterMonitor", db: "admin" },
        { role: "backup", db: "admin" }
     ]
```

```
    }
)
```

See *MongoDB 2.6* and *MongoDB 3.0 and Later* for more information on the required access for the Monitoring Agent and the Backup Agent.

Edit Agent Configuration File

Edit the `/etc/mongodb-mms/monitoring-agent.config` file.

Step 1: Set the `krb5Principal`

Set the `krb5Principal` to the name of the Kerberos principal. For example:

```
krb5Principal=mmsagent/instance@EXAMPLE.COM
```

Step 2: Set the `krb5Keytab`

Set the `krb5Keytab` value to the complete absolute path of the keytab file. For example:

```
krb5Keytab=/etc/mongodb-mms/mmsagent.keytab
```

Step 3: Restart the agent.

Host Settings

In addition to adding the agent as a MongoDB user, you must also specify the host's authentication settings. You can specify the host's authentication settings when *adding* the host, or you can *edit the settings* for an existing host.

Configure Kerberos Environment

Step 1: Create or configure the `/etc/kerb5.conf` file on the system to integrate this host into your Kerberos environment.

Step 2: Ensure the `kinit` binary is available at the `/user/bin/kinit` path.

Configure Monitoring Agent for SSL

On this page

Overview

Ops Manager supports SSL for encrypting the following connections made by Monitoring Agents:

- Connections between the Monitoring Agents and MongoDB instances.
- Connections between the Monitoring Agents and Ops Manager.

Prerequisite

To configure the agent to use SSL, you must have a trusted CA certificate that signed the MongoDB instance's certificate.

Procedures

Connections between Agents and MongoDB Instances

To use SSL for the Monitoring Agent's connection to a MongoDB host, specify the host's SSL settings when *adding the host* or by *editing the host's settings*.

Then configure the agent to use SSL:

Step 1: Specify path to trusted CA certificate.

If your MongoDB deployment uses SSL, then you must configure the Monitoring Agent to use SSL. To configure the agent to use SSL, you must have a trusted CA certificate that signed the MongoDB instance's certificate.

In the agent's install directory, edit the `monitoring-agent.config` file to set *`sslTrustedServerCertificates`* field to the path of a file containing one or more certificates in PEM format. For example if you would use the following command to connect through the `mongo` shell:

```
mongo --ssl --sslCAFile /etc/ssl/ca.pem example.net:27017
```

Then you would set:

```
sslTrustedServerCertificates=/etc/ssl/ca.pem
```

By default, to connect to MongoDB instances using SSL requires a valid trusted certificate.

For testing purposes, however, you can set the *`sslRequireValidServerCertificates`* setting to `false` to bypass this check. When *`sslRequireValidServerCertificates`* is `false`, you do not need to specify the path to the trusted CA certificate in the *`sslTrustedServerCertificates`* setting, since Ops Manager will not verify the certificates. This configuration is **not** recommended for production use as it makes connections susceptible to man-in-the-middle attacks.

For additional information on these settings, including client certificate support, see *MongoDB SSL Settings*.

Step 2: Restart agent.

Note: For additional information on SSL settings, including client certificate support, see *MongoDB SSL Settings*.

Connections between Agents and Ops Manager

To ensure that the Monitoring Agents use SSL when connecting to Ops Manager, Configure Ops Manager to use SSL for all connections. The *Configure SSL Connections to Ops Manager* tutorial describes how to set up Ops Manager to run over HTTPS.

Starting with Ops Manager 1.4, the Monitoring Agent validates the SSL certificate of the Ops Manager by default.

If you are not using a certificate signed by a trusted 3rd party, you must configure the Monitoring Agent to trust Ops Manager.

To specify a self-signed certificate for Ops Manager that the Monitoring Agent should trust:

Step 1: Copy your PEM certificate to `/etc/mongodb-mms/`.

Issue the following sequence of commands:

```
sudo cp -a mms-ssl-unified.crt /etc/mongodb-mms/
sudo chown mongodb-mms-agent:mongodb-mms-agent /etc/mongodb-mms/mms-ssl-unified.crt
sudo chmod 600 /etc/mongodb-mms/mms-ssl-unified.crt
```

Step 2: Edit the following parameter in `/etc/mongodb-mms/monitoring-agent.config`.

For example:

```
sslTrustedMMSServerCertificate=/etc/mongodb-mms/mms-ssl-unified.crt
```

Step 3: Restart the Monitoring Agent for the configuration update to take effect.

For example:

```
sudo /etc/init.d/mongodb-mms-monitoring-agent restart
```

Configure Hardware Monitoring with `munin-node`

On this page

Overview

To chart the hardware statistics collected with Munin, Ops Manager supports the following munin-node plugins:

- `cpu` plugin, which creates the `cputime` chart.

- `iostat` plugin, which creates the `iostat` chart.
- `iostat_ios` plugin, which creates the `iotime` chart.

Install the `munin-node` Package

You must install the `munin-node` package on all of the host systems that you wish to monitor. Ensure that the Monitoring Agent can connect to the `munin-node` process on port 4949 of the monitored host to collect data.

Note: `munin-node` and hardware monitoring is only available for MongoDB instances running on Linux hosts.

On Debian and Ubuntu systems, issue the following command to install `munin-node`:

```
sudo apt-get install munin-node
```

On Red Hat, CentOS, and Fedora systems, you may need to first install the EPEL repository before installing `munin-node`. See https://fedoraproject.org/wiki/EPEL. To install `munin-node` issue the following command:

```
yum install munin-node
```

Configure `munin-node`

When installation is complete, ensure that `munin-node`:

- is running. Use the command, "`ps -ef | grep "munin"`" to confirm. If the process is not running, issue the command "`/etc/init.d/munin-node start`".
- will start following the next system reboot. This is the default behavior on most Debian-based systems. Red Hat and related distributions should use the "`chkconfig`" command, to configure this behavior (i.e. "`chkconfig munin-node on`")
- is accessible from the system running the agent. `munin-node` uses port 4949, which needs to be open on the monitored system, so the agent can access this data source. Use the following procedure to test access:

  ```
  telnet [HOSTNAME] 4949
  fetch iostat
  fetch iostat_ios
  fetch cpu
  ```

  Replace `[HOSTNAME]` with the hostname of the monitored system. Run these commands from the system where the Monitoring Agent is running. If these "`fetch`" commands return data, then `munin-node` is running and accessible by the Monitoring Agent.

Note: On some platforms, `munin-node` does not have all required plugins enabled.

For CentOS and Ubuntu, the `munin-node` package does not have the `iostat` and `iostat_ios` plugins enabled. Use the following operation to enable these plugins:

```
sudo ln -s /usr/share/munin/plugins/iostat /etc/munin/plugins/iostat
sudo ln -s /usr/share/munin/plugins/iostat_ios /etc/munin/plugins/iostat_ios
sudo /etc/init.d/munin-node restart
```

If `munin-node` is running but inaccessible, make sure that you have access granted for the system running the Monitoring Agent and that no firewalls block the port between `munin-node` and the Monitoring Agent. You may find the `munin-node` configuration at `/etc/munin-node/munin-node.conf`, `/etc/munin/munin-node.conf`, or `/etc/munin-node.conf`, depending on your distribution.

Additional Considerations for `munin-node`

- If you have numbered disk devices (e.g. `/dev/sda1` and `/dev/sda2`) then you will need to configure support for numbered disk in the munin iostat plugin. Find the configuration file at `/etc/munin/plugin-conf.d/munin-node` or a similar path, and add the following value:

  ```
  [iostat]
      env.SHOW_NUMBERED 1
  ```

- If you have Munin enabled and do not have `iostat ios` data in your Munin charts, your `munin-node` may not have write access to required state files in its `munin/plugin-state/` directory. See the `munin-node` plugin log (i.e. `/var/log/munin/munin-node.log` or similar depending on your distribution) for more information.

  The full path of this state directory depends on the system, but is typically `/var/lib/munin/plugin-state/`.

  Run the following command sequence to correct this issue. The last command in the sequence changes permissions for the `/var/lib/munin/plugin-state/` directory to ensure access for the `munin-node` plugins. Depending on your setup, you might have to use a different permission level:

  ```
  touch /var/lib/munin/plugin-state/iostat-ios.state
  chown -R [username]:[group] /var/lib/munin/plugin-state/
  chmod -R 766 /var/lib/munin/plugin-state/
  ```

  Replace `[username]` and `[group]` with the username and group that the `munin-node` process runs with.

- Add the host running the Monitoring Agent to the *allow* directive in the `munin-node.conf` file. The *allow* directive lists hosts allowed to query the munin-node process. Otherwise, traffic from the Ops Manager host will be allowed via firewall but will not be collected by munin.

  The `munin-node.conf` file is located in one of the following directories, depending on your distribution: `/etc/munin-node`, `/etc/munin`, or `/etc`.

If you encounter any other problems, check the log files for `munin-node` to ensure that there are no errors with Munin. `munin-node` writes logs files in the `/var/log/` directory on the monitored system.

See also:

Munin Diagnostics.

Start or Stop the Monitoring Agent

On this page

Overview

For maintenance or troubleshooting purposes, you may want to temporarily shut down or restart Ops Manager's Monitoring Agent. However, for proper operation of Ops Manager your Ops Manager group must have at least one Monitoring Agent running. The group needs only one Monitoring Agent running.

Procedures

Start the Monitoring Agent

The procedure to *Install the Monitoring Agent* includes a step to start the agent. If you must restart the agent, use the following procedure.

Start an Agent Installed with an `rpm` Package

If you installed the Monitoring Agent using an `rpm` package, such as on RHEL, CentOS, or SUSE, issue the following command to start the agent:

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

Start an Agent Installed with a `deb` Package

If you installed the Monitoring Agent using a `deb` package, as on Ubuntu, issue the following command to start the agent:

```
sudo start mongodb-mms-monitoring-agent
```

Start an Agent Installed with a `tar` File

Use this command if you installed to Linux or OSX using a `tar` file. Issue the following command from the directory to which you installed the Monitoring Agent:

```
nohup ./mongodb-mms-monitoring-agent- >> monitoring-agent.log 2>&1 &
```

Start the Monitoring Agent on Windows

In Windows `Control Panel`, open `Administrative Tools` and then `Services`. In the list of services, select `MMS Monitoring Agent`. Select the `Action` menu and select `Start`.

Stop the Monitoring Agent

You must have at least one Monitoring Agent running to monitor your deployment.

Stop an Agent Installed with an `rpm` Package

If you installed the Monitoring Agent using an `rpm` package, such as on RHEL, CentOS, or SUSE, issue the following command to stop the agent:

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

Stop an Agent Installed with a `deb` Package

If you installed the Monitoring Agent using a `deb` package, as on Ubuntu, issue the following command to stop the agent:

```
sudo stop mongodb-mms-monitoring-agent
```

Stop an Agent Installed with a `tar` File

If you installed to Linux or OSX using a `tar` file, issue the following command to stop the Monitoring Agent:

```
pkill -f mongodb-mms-monitoring-agent
```

Stop the Monitoring Agent on Windows

In Windows `Control Panel`, open `Administrative Tools` and then `Services`. In the list of services, select `MMS Monitoring Agent`. Select the `Action` menu and select `Stop`.

Remove Monitoring Agents from Ops Manager

On this page

Ops Manager displays active Monitoring Agents on the *Agents* page in the *Administration* tab. The page displays agents that have been active in the last 24 hours. If an agent fails to report to Ops Manager for more than 24 hours, Ops Manager removes the agent from the *Agents* page.

Remove from Ops Manager

To remove a Monitoring Agent from Ops Manager, *stop the agent* and then wait 24 hours.

Delete from the Server

To delete the Monitoring Agent **from a Linux or OSX server**, *stop the agent* and then remove the `mongodb-mms-monitoring-agent` file from the `/usr/bin` directory. If you installed the agent using a `tar.gz` file, the agent will be in the directory you chose during installation.

To delete the Monitoring Agent **from a Windows server**, *stop the agent* and then use the Windows program uninstaller to remove the `MMS Monitoring Agent` program.

12.4 Backup Agent

Install or Update the Agent Procedures for installing and updating the Backup Agent.

Backup Agent Configuration Documentation of the settings available in the Backup Agent configuration file.

Required Access for Backup Agent Details the permissions required for Backup Agent to use with MongoDB instances that enforce access control.

Configure the Agent for Access Control If MongoDB uses Access Control, create a MongoDB user for the Backup Agent to use to authenticate and to determine the agent's access.

Configure the Agent for SSL Configure the Backup Agent to support SSL.

Start or Stop the Agent Procedures to start and stop the Backup Agent.

Remove the Agent Remove the Backup Agent.

Install Backup Agent

The Backup Agent polls the primary MongoDB instance of every backup-enabled replica set and transmits the operations to the Ops Manager service.

The Backup Agent relies on the Ops Manager Monitoring Agent to populate the list of sharded clusters and replica sets eligible for backup. If the appropriate hosts are not added, or the Monitoring Agent is not being correctly run, the lists may be incomplete or out-of-date. If you have not already installed and configured Monitoring, please refer to the *Install Monitoring Agent* documentation.

Install with RPM Packages Install and start the Backup Agent using an `rpm` package.

Install on Ubuntu Install and start the Backup Agent on Ubuntu using a `deb` package.

Install on Other Linux Systems Install and start the Backup Agent on other Linux systems using the `tar.gz` archive packages.

Install on OS X Install and start the Backup Agent on OS X.

Install on Windows Install and start the Backup Agent on Windows.

Install or Update the Backup Agent with `rpm` Packages

On this page

Overview

The Backup Agent polls the primary MongoDB instance of every backup-enabled replica set and transmits the operations to the Ops Manager service.

The Backup Agent relies on the Ops Manager Monitoring Agent to populate the list of sharded clusters and replica sets eligible for backup. If the appropriate hosts are not added, or the Monitoring Agent is not being correctly run, the lists may be incomplete or out-of-date. If you have not already installed and configured Monitoring, please refer to the *Install Monitoring Agent* documentation.

Considerations

MongoDB Requirements

Ops Manager only supports backing up replica sets and sharded cluster, and does *not* support backing up standalone instances.

Ops Manager only supports backup for replica sets that run MongoDB 2.0 or later.

Ops Manager only supports backup for sharded clusters that run MongoDB 2.4 or later.

All backed up replica sets and config servers must be able to maintain oplog entries, by default, for at least 3 hours over the last 24 hour period. This window is configurable with the `brs.minimumReplicationOplogWindowHr` setting in the `conf-mms.properties` file for the Ops Manager Application. See *Ops Manager Configuration Files* for more information.

Agent Architecture

To avoid resource contention, run the agent on a host other than the hosts where the MongoDB instances are running. Be sure the agent can access the MongoDB hosts.

Running on Amazon EC2

If you run the Backup Agent on Amazon EC2, do not use the `t1.micro` instance type, which has a CPU scheduling policy that does not typically provide sufficient capacity to support a Backup Agent for a production deployment. Use a larger instance type instead.

Prerequisites

Monitoring Agent

Install and configure the Monitoring, as described in the *Monitoring Agent* documentation.

Firewall

If your MongoDB instances operate within a firewall, configure your network infrastructure to allow outbound connections on port `443` (SSL) to `api-backup.mongodb.com`.

Access Control

If you use Backup with a MongoDB deployment that uses authentication, before installing the Backup Agent, you must create a user in MongoDB with the appropriate access. See *Configure Backup Agent for Access Control*.

Backup Directory

After you install the Backup Agent, **do not** use the agent's directory location for anything other than the agent itself. The Backup Agent periodically deletes the contents of its root directory.

Procedures

This section includes procedures for both installing and updating the Backup Agent on RHEL, CentOS, SUSE, Amazon Linux, and other systems that use `rpm` packages.

Install the Backup Agent with an `rpm` Package

Use this procedure to install the agent on RHEL, CentOS, SUSE, Amazon Linux, and other systems that use `rpm` packages.

Step 1: Download the latest version of the Backup Agent package.

```
curl -OL <mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.x86_64.rpm
```

Step 2: Install the Backup Agent package.

Issue the following command:

```
sudo rpm -U mongodb-mms-backup-agent-latest.x86_64.rpm
```

Step 3: Retrieve the API key for your Ops Manager group.

In the *Administration* tab on the *Agent* page, click the box for your operating system. Ops Manager will then display a procedure that includes a step to set your API key. The step displays the actual API key used by your Ops Manager group. Copy the key.

Step 4: Configure the `backup-agent.config` file with the API key.

In the `/etc/mongodb-mms/backup-agent.config` file, set the `mmsApiKey` property to your API key.

Step 5: Optional: For SUSE 11+ deployments only, configure the `sslTrustedMMSServerCertificate` property.

If you are deploying on SUSE, you must configure the `sslTrustedMMSServerCertificate` setting. All other users should omit this step.

Enter the following property and value in the `/etc/mongodb-mms/backup-agent.config` file:

```
sslTrustedMMSServerCertificate=/etc/ssl/certs/UTN_USERFirst_Hardware_Root_CA.pem
```

Save and close the file.

Step 6: Start the Backup Agent.

Issue the following command:

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

Update the Backup Agent with an `rpm` Package

Use this procedure to update the agent on RHEL, CentOS, SUSE, Amazon Linux, and other systems that use `rpm` packages.

Step 1: Download the latest version of the Backup Agent package.

```
curl -OL <mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.x86_64.rpm
```

Step 2: Install the Backup Agent package.

Issue the following command:

```
sudo rpm -U mongodb-mms-backup-agent-latest.x86_64.rpm
```

Step 3: Start the Backup Agent.

Issue the following command:

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

Next Steps

After you have successfully installed the Backup Agent, see *Activate Backup* to enable backup for a replica set.

Additional Information

The `README` included with the downloaded package also provides information about the Backup Agent.

For details about backup operations, see *Backup FAQs*.

Install or Update the Backup Agent with `deb` Packages

On this page

Overview

The Backup Agent polls the primary MongoDB instance of every backup-enabled replica set and transmits the operations to the Ops Manager service.

The Backup Agent relies on the Ops Manager Monitoring Agent to populate the list of sharded clusters and replica sets eligible for backup. If the appropriate hosts are not added, or the Monitoring Agent is not being correctly run, the lists may be incomplete or out-of-date. If you have not already installed and configured Monitoring, please refer to the *Install Monitoring Agent* documentation.

Considerations

MongoDB Requirements

Ops Manager only supports backing up replica sets and sharded cluster, and does *not* support backing up standalone instances.

Ops Manager only supports backup for replica sets that run MongoDB 2.0 or later.

Ops Manager only supports backup for sharded clusters that run MongoDB 2.4 or later.

All backed up replica sets and config servers must be able to maintain oplog entries, by default, for at least 3 hours over the last 24 hour period. This window is configurable with the `brs.minimumReplicationOplogWindowHr` setting in the `conf-mms.properties` file for the Ops Manager Application. See *Ops Manager Configuration Files* for more information.

Agent Architecture

To avoid resource contention, run the agent on a host other than the hosts where the MongoDB instances are running. Be sure the agent can access the MongoDB hosts.

Running on Amazon EC2

If you run the Backup Agent on Amazon EC2, do not use the `t1.micro` instance type, which has a CPU scheduling policy that does not typically provide sufficient capacity to support a Backup Agent for a production deployment. Use a larger instance type instead.

Prerequisites

Monitoring Agent

Install and configure the Monitoring, as described in the *Monitoring Agent* documentation.

Firewall

If your MongoDB instances operate within a firewall, configure your network infrastructure to allow outbound connections on port `443` (SSL) to `api-backup.mongodb.com`.

Access Control

If you use Backup with a MongoDB deployment that uses authentication, before installing the Backup Agent, you must create a user in MongoDB with the appropriate access. See *Configure Backup Agent for Access Control*.

Backup Directory

After you install the Backup Agent, **do not** use the agent's directory location for anything other than the agent itself. The Backup Agent periodically deletes the contents of its root directory.

Procedures

This section includes procedures for installing and updating the Backup Agent on Ubuntu with `deb` packages. For Debian systems, use the *Install or Update the Backup Agent from an Archive* procedure.

Install the Backup Agent with a `deb` Package

Step 1: Download the latest version of the Backup Agent package.

From a system shell, issue the following command, where `<mmsUri>` is the URI of your Ops Manager installation (for example, `onprem.example.net`):

```
curl -OL <mmsUri>/download/agent/backup/mongodb-mms-backup-agent_latest_amd64.deb
```

Step 2: Install the Backup Agent package.

Issue the following command:

```
sudo dpkg -i mongodb-mms-backup-agent_latest_amd64.deb
```

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, select your the approriate link for the Backup agent and your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Configure the `backup-agent.config` file with the API key.

In the `/etc/mongodb-mms/backup-agent.config` file, set the `mmsApiKey` property to your API key.

Step 5: Start the Backup Agent.

Issue the following command:

```
sudo start mongodb-mms-backup-agent
```

Update the Backup Agent with a `deb` Package

Step 1: Download the latest version of the Backup Agent package.

From a system shell, issue the following command, where `<mmsUri>` is the URI of your Ops Manager installation (for example, `onprem.example.net`):

```
curl -OL <mmsUri>/download/agent/backup/mongodb-mms-backup-agent_latest_amd64.deb
```

Step 2: If your current Backup Agent has a version number earlier that 2.0, purge the existing agent.

Perform this step **only** if your current agent's version is earlier than 2.0. To purge the existing Backup Agent, issue the following command:

```
sudo dpkg -P mongodb-mms-backup-agent
```

Step 3: Install the Backup Agent package.

Issue the following command:

```
sudo dpkg -i mongodb-mms-backup-agent_latest_amd64.deb
```

Step 4: Start the Backup Agent.

Issue the following command:

```
sudo start mongodb-mms-backup-agent
```

Next Steps

After you have successfully installed the Backup Agent, see *Activate Backup* to enable backup for a replica set.

Additional Information

The `README` included with the downloaded package also provides information about the Backup Agent.

For details about backup operations, see *Backup FAQs*.

Install or Update the Backup Agent from an Archive

On this page

Overview

The Backup Agent polls the primary MongoDB instance of every backup-enabled replica set and transmits the operations to the Ops Manager service.

The Backup Agent relies on the Ops Manager Monitoring Agent to populate the list of sharded clusters and replica sets eligible for backup. If the appropriate hosts are not added, or the Monitoring Agent is not being correctly run, the lists may be incomplete or out-of-date. If you have not already installed and configured Monitoring, please refer to the *Install Monitoring Agent* documentation.

Considerations

MongoDB Requirements

Ops Manager only supports backing up replica sets and sharded cluster, and does *not* support backing up standalone instances.

Ops Manager only supports backup for replica sets that run MongoDB 2.0 or later.

Ops Manager only supports backup for sharded clusters that run MongoDB 2.4 or later.

All backed up replica sets and config servers must be able to maintain oplog entries, by default, for at least 3 hours over the last 24 hour period. This window is configurable with the `brs.minimumReplicationOplogWindowHr`

setting in the `conf-mms.properties` file for the Ops Manager Application. See *Ops Manager Configuration Files* for more information.

Agent Architecture

To avoid resource contention, run the agent on a host other than the hosts where the MongoDB instances are running. Be sure the agent can access the MongoDB hosts.

Running on Amazon EC2

If you run the Backup Agent on Amazon EC2, do not use the `t1.micro` instance type, which has a CPU scheduling policy that does not typically provide sufficient capacity to support a Backup Agent for a production deployment. Use a larger instance type instead.

Prerequisites

Monitoring Agent

Install and configure the Monitoring, as described in the *Monitoring Agent* documentation.

Firewall

If your MongoDB instances operate within a firewall, configure your network infrastructure to allow outbound connections on port `443` (SSL) to `api-backup.mongodb.com`.

Access Control

If you use Backup with a MongoDB deployment that uses authentication, before installing the Backup Agent, you must create a user in MongoDB with the appropriate access. See *Configure Backup Agent for Access Control*.

Backup Directory

After you install the Backup Agent, **do not** use the agent's directory location for anything other than the agent itself. The Backup Agent periodically deletes the contents of its root directory.

Procedures

This section includes procedures for installing and updating the Backup Agent on a Linux system not listed in the *Agent Downloads* list on the *Agents* page in the *Administration* tab.

Install the Backup Agent from a `tar.gz` Archive

Step 1: Download the latest version of the Backup Agent archive.

On a system shell, issue a command that resembles the following. Replace `linux_x86_64` with your platform, as needed: depending on your operating system:

```
curl -OL <mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.linux_x86_64.
↪tar.gz
```

Step 2: Install the Backup Agent.

To install the agent, extract the archive using a command that resembles the following. Replace `linux_x86_64` with your platform, as needed:

```
tar -xf mongodb-mms-backup-agent-latest.linux_x86_64.tar.gz
```

The Backup Agent is installed.

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, select your the approriate link for the Backup agent and your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `local.config` file to include your Ops Manager API key.

In the directory where you installed the Backup Agent, locate and open the `local.config` file. Enter your API key as the value for the `mmsApiKey` setting.

Step 5: Optional: Configure the agent to use a proxy server.

If the agent will connect to Ops Manager via a proxy server, you must specify the server in the `http_proxy` environment variable. To specify the server, use the `export` command, as in the following example:

```
export http_proxy="http://proxy.example.com:9000"
```

To connect through a proxy, you must install the agent from a `.tar.gz` file, *not* from a `.deb` or `.rpm` file.

Step 6: Start the Backup Agent.

Issue the following command:

```
nohup ./mongodb-mms-backup-agent >> backup-agent.log 2>&1 &
```

Update the Backup Agent from a `tar.gz` Archive

Step 1: Stop any currently running Backup Agents.

Issue the following command with the system shell:

```
pkill -f mongodb-mms-backup-agent
```

Step 2: Download the latest version of the Backup Agent archive.

On a system shell, issue a command that resembles the following. Replace `linux_x86_64` with your platform, as needed: depending on your operating system:

```
curl -OL <mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.linux_x86_64.
↪tar.gz
```

Step 3: Install the Backup Agent.

To install the agent, extract the archive using a command that resembles the following. Replace `linux_x86_64` with your platform, as needed:

```
tar -xf mongodb-mms-backup-agent-latest.linux_x86_64.tar.gz
```

The Backup Agent is installed.

Step 4: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, select your the approriate link for the Backup agent and your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 5: Edit the `local.config` file to include your Ops Manager API key.

In the directory where you installed the Backup Agent, locate and open the `local.config` file. Enter your API key as the value for the `mmsApiKey` setting.

Step 6: Start the Backup Agent.

Issue the following command:

```
nohup ./mongodb-mms-backup-agent >> backup-agent.log 2>&1 &
```

Next Steps

After you have successfully installed the Backup Agent, see *Activate Backup* to enable backup for a replica set.

Additional Information

If you installed the Backup Agent from the `tar.gz` archives, see /tutorial/rotate-agent-log-files to configure log rotation.

The `README` included with the downloaded package also provides information about the Backup Agent.

For details about backup operations, see *Backup FAQs*.

Install or Update the Backup Agent on OS X

On this page

Overview

The Backup Agent polls the primary MongoDB instance of every backup-enabled replica set and transmits the operations to the Ops Manager service.

The Backup Agent relies on the Ops Manager Monitoring Agent to populate the list of sharded clusters and replica sets eligible for backup. If the appropriate hosts are not added, or the Monitoring Agent is not being correctly run, the lists may be incomplete or out-of-date. If you have not already installed and configured Monitoring, please refer to the *Install Monitoring Agent* documentation.

Considerations

MongoDB Requirements

Ops Manager only supports backing up replica sets and sharded cluster, and does *not* support backing up standalone instances.

Ops Manager only supports backup for replica sets that run MongoDB 2.0 or later.

Ops Manager only supports backup for sharded clusters that run MongoDB 2.4 or later.

All backed up replica sets and config servers must be able to maintain oplog entries, by default, for at least 3 hours over the last 24 hour period. This window is configurable with the `brs.minimumReplicationOplogWindowHr` setting in the `conf-mms.properties` file for the Ops Manager Application. See *Ops Manager Configuration Files* for more information.

Agent Architecture

To avoid resource contention, run the agent on a host other than the hosts where the MongoDB instances are running. Be sure the agent can access the MongoDB hosts.

Running on Amazon EC2

If you run the Backup Agent on Amazon EC2, do not use the `t1.micro` instance type, which has a CPU scheduling policy that does not typically provide sufficient capacity to support a Backup Agent for a production deployment. Use a larger instance type instead.

Prerequisites

Monitoring Agent

Install and configure the Monitoring, as described in the *Monitoring Agent* documentation.

Firewall

If your MongoDB instances operate within a firewall, configure your network infrastructure to allow outbound connections on port `443` (SSL) to `api-backup.mongodb.com`.

Access Control

If you use Backup with a MongoDB deployment that uses authentication, before installing the Backup Agent, you must create a user in MongoDB with the appropriate access. See *Configure Backup Agent for Access Control*.

Backup Directory

After you install the Backup Agent, **do not** use the agent's directory location for anything other than the agent itself. The Backup Agent periodically deletes the contents of its root directory.

Procedures

Install the Backup Agent On OS X

Use the following procedure to install the agent on OS X:

Step 1: Download the latest version of the Backup Agent archive.

On a system shell, issue a command that resembles the following. Replace `linux_x86_64` with your platform, as needed: depending on your operating system:

```
curl -OL <mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.osx_x86_64.
↪tar.gz
```

Step 2: Install the Backup Agent.

To install the agent, extract the archive using a command that resembles the following. Replace `linux_x86_64` with your platform, as needed:

```
tar -xf mongodb-mms-backup-agent-latest.osx_x86_64.tar.gz
```

The Backup Agent is installed.

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, select your the approriate link for the Backup agent and your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `local.config` file to include your Ops Manager API key.

In the directory where you installed the Backup Agent, locate and open the `local.config` file. Enter your API key as the value for the `mmsApiKey` setting.

Step 5: Optional: Configure the agent to use a proxy server.

If the agent will connect to Ops Manager via a proxy server, you must specify the server in the `http_proxy` environment variable. To specify the server, use the `export` command, as in the following example:

```
export http_proxy="http://proxy.example.com:9000"
```

To connect through a proxy, you must install the agent from a `.tar.gz` file, *not* from a `.deb` or `.rpm` file.

Step 6: Start the Backup Agent.

Issue the following command:

```
nohup ./mongodb-mms-backup-agent >> backup-agent.log 2>&1 &
```

Update the Backup Agent

Use the following procedure to update the agent on OS X:

Step 1: Download the latest version of the Backup Agent archive.

On a system shell, issue a command that resembles the following. Replace `linux_x86_64` with your platform, as needed: depending on your operating system:

```
curl -OL <mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.osx_x86_64.
↪tar.gz
```

Step 2: Install the Backup Agent.

To install the agent, extract the archive using a command that resembles the following. Replace `linux_x86_64` with your platform, as needed:

```
tar -xf mongodb-mms-backup-agent-latest.osx_x86_64.tar.gz
```

The Backup Agent is installed.

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, select your the approriate link for the Backup agent and your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `local.config` file to include your Ops Manager API key.

In the directory where you installed the Backup Agent, locate and open the `local.config` file. Enter your API key as the value for the `mmsApiKey` setting.

Step 5: Optional: Configure the agent to use a proxy server.

If the agent will connect to Ops Manager via a proxy server, you must specify the server in the `http_proxy` environment variable. To specify the server, use the `export` command, as in the following example:

```
export http_proxy="http://proxy.example.com:9000"
```

To connect through a proxy, you must install the agent from a `.tar.gz` file, *not* from a `.deb` or `.rpm` file.

Step 6: Start the Backup Agent.

Issue the following command:

```
nohup ./mongodb-mms-backup-agent >> backup-agent.log 2>&1 &
```

Next Steps

After you have successfully installed the Backup Agent, see *Activate Backup* to enable backup for a replica set.

Additional Information

The `README` included with the downloaded package also provides information about the Backup Agent.

For details about backup operations, see *Backup FAQs*.

Install or Update the Backup Agent on Windows

On this page

Overview

The Backup Agent polls the primary MongoDB instance of every backup-enabled replica set and transmits the operations to the Ops Manager service.

The Backup Agent relies on the Ops Manager Monitoring Agent to populate the list of sharded clusters and replica sets eligible for backup. If the appropriate hosts are not added, or the Monitoring Agent is not being correctly run, the lists may be incomplete or out-of-date. If you have not already installed and configured Monitoring, please refer to the *Install Monitoring Agent* documentation.

Considerations

MongoDB Requirements

Ops Manager only supports backing up replica sets and sharded cluster, and does *not* support backing up standalone instances.

Ops Manager only supports backup for replica sets that run MongoDB 2.0 or later.

Ops Manager only supports backup for sharded clusters that run MongoDB 2.4 or later.

All backed up replica sets and config servers must be able to maintain oplog entries, by default, for at least 3 hours over the last 24 hour period. This window is configurable with the `brs.minimumReplicationOplogWindowHr` setting in the `conf-mms.properties` file for the Ops Manager Application. See *Ops Manager Configuration Files* for more information.

Agent Architecture

To avoid resource contention, run the agent on a host other than the hosts where the MongoDB instances are running. Be sure the agent can access the MongoDB hosts.

Running on Amazon EC2

If you run the Backup Agent on Amazon EC2, do not use the `t1.micro` instance type, which has a CPU scheduling policy that does not typically provide sufficient capacity to support a Backup Agent for a production deployment. Use a larger instance type instead.

Prerequisites

Monitoring Agent

Install and configure the Monitoring, as described in the *Monitoring Agent* documentation.

Firewall

If your MongoDB instances operate within a firewall, configure your network infrastructure to allow outbound connections on port `443` (SSL) to `api-backup.mongodb.com`.

Access Control

If you use Backup with a MongoDB deployment that uses authentication, before installing the Backup Agent, you must create a user in MongoDB with the appropriate access. See *Configure Backup Agent for Access Control*.

Backup Directory

After you install the Backup Agent, **do not** use the agent's directory location for anything other than the agent itself. The Backup Agent periodically deletes the contents of its root directory.

Procedures

Install the Backup Agent On Windows

Step 1: Download and run the latest version of the Backup Agent MSI file.

To download the 64-bit MSI file, use the following URL, where `<mmsUri>` is the hostname of the Backup server:

```
<mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.windows_x86_64.msi
```

To download the 32-bit MSI file, use the following URL, where `<mmsUri>` is the hostname of the Backup server:

```
<mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.windows_i386.msi
```

During installation, the installer prompts you to specify the folder for storing configuration and log files. It is strongly advised that you encrypt or restrict access to this folder.

Step 2: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, select your the approriate link for the Backup agent and your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 3: Edit the `local.config` file to include your Ops Manager API key.

In the directory where you installed the Backup Agent, locate and open the `local.config` file. Enter your API key as the value for the `mmsApiKey` setting.

Step 4: Edit the `local.config` file to include the hostname of the Backup server.

In the Backup Agent installation directory, open the `local.config` file and set the `mothership` property to hostname of the Backup server.

Step 5: Start the Backup Agent.

In Windows `Control Panel`, open `Administrative Tools`, and then open `Services`.

In the list of services, select the `MMS Backup Agent` service. Select the `Action` menu and select `Start`.

Update the Backup Agent on Windows

Step 1: Stop all currently running Backup Agents.

In Windows `Control Panel`, open `Administrative Tools` and then `Services`. In the list of services, select `MMS Backup Agent`. Select the `Action` menu and select `Stop`.

If you receive a message that your Backup Agent is out of date, make sure you are running an upgradeable version of the Backup Agent. If you are running the version of the Backup Agent named `MongoDBBackup`, you must remove it before upgrading. To check if you are running `MongoDBBackup`, issue the following command in an Administrative command prompt:

```
sc query MongoDBBackup
```

If the command returns a result, you must remove the `MongoDBBackup` agent. To remove it, issue the following:

```
sc delete MongoDBBackup
```

Step 2: Download and run the latest version of the Backup Agent MSI file.

To download the 64-bit MSI file, use the following URL, where `<mmsUri>` is the hostname of the Backup server:

```
<mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.windows_x86_64.msi
```

To download the 32-bit MSI file, use the following URL, where `<mmsUri>` is the hostname of the Backup server:

```
<mmsUri>/download/agent/backup/mongodb-mms-backup-agent-latest.windows_i386.msi
```

During installation, the installer prompts you to specify the folder for storing configuration and log files. It is strongly advised that you encrypt or restrict access to this folder.

Step 3: Retrieve the Ops Manager API key for your Ops Manager group.

In the *Administration* tab on the *Agents* page, select your the approriate link for the Backup agent and your operating system. Ops Manager will then display a procedure that includes a step to set your Ops Manager API key. The step displays the actual Ops Manager API key used by your Ops Manager group. Copy the key.

Step 4: Edit the `local.config` file to include your Ops Manager API key.

In the directory where you installed the Backup Agent, locate and open the `local.config` file. Enter your API key as the value for the `mmsApiKey` setting.

Step 5: Start the Backup Agent.

In Windows `Control Panel`, open `Administrative Tools`, and then open `Services`.

In the list of services, select the `MMS Backup Agent` service. Select the `Action` menu and select `Start`.

Next Steps

After you have successfully installed the Backup Agent, see *Activate Backup* to enable backup for a replica set.

Additional Information

The `README` included with the downloaded package also provides information about the Backup Agent.

For details about Backup operations, see *Backup FAQs*.

Backup Agent Configuration

On this page

Warning: Do **not** edit these settings for a Backup Agent that is managed by an Automation Agent. If you do, the Automation Agent will overwrite any changes you make.

Configuration File

The name and location of the Backup Agent configuration file depend on the operating system:

- RHEL, CentOS, Amazon Linux, and Ubuntu all use a package manager to install the agent. The package manager creates the following agent configuration file:

  ```
  /etc/mongodb-mms/backup-agent.config
  ```

- OS X, Windows, and other Linux systems use either a `tar` or `msi` file for the installation. The Backup Agent stores its configuration in the following file:

  ```
  <installation directory>/local.config
  ```

Settings

Connection Settings

For the Backup Agent communication with the Ops Manager servers, the following connection settings are **required**:

mmsApiKey
Type: string

The Ops Manager agent API key for a Ops Manager group. To retrieve the key from the Ops Manager interface, click the *Administration* tab, then the *Agents* page, and then the link for your operating system. Ops Manager will display the Ops Manager API key used by your Ops Manager group.

For example:

```
mmsApiKey=abc123
```

mothership
Type: string

The hostname of the Ops Manager Backup Web Server.

https
Type: boolean

Toggles communication with the Ops Manager Backup web server over HTTPS.

HTTP Proxy Settings

httpProxy
New in version 1.4.4.34-1.

Type: string

To connect to the Ops Manager HTTP Service via a proxy, specify the URL of the proxy. For example:

```
httpProxy=http://example-proxy.com:8080
```

MongoDB SSL Settings

Specify these settings when the Backup Agent is connecting to MongoDB instances with SSL.

sslClientCertificate
Type: string

The path to the private key, client certificate, and optional intermediate certificates in PEM format. The agent will use the client certificate when connecting to a MongoDB instance that uses SSL and requires client certificates, i.e. that is running using the `--sslCAFile` option.

sslClientCertificatePassword
Type: string

The password needed to decrypt the private key in the sslClientCertificate file. This setting is only necessary if the client certificate PEM file is encrypted.

sslTrustedServerCertificates
Type: string

The path on disk that contains the trusted certificate authority certificates in PEM format. These certificates will verify the server certificate returned from any MongoDBs running with SSL. For example:

```
sslTrustedServerCertificates=/etc/mongodb-mms/mongodb-certs.pem
```

sslRequireValidServerCertificates

Type: boolean

Use this option to disable certificate verification by setting this value to `false`. That configuration is only recommended for testing purposes as it makes connections susceptible to man-in-the-middle attacks.

MongoDB Kerberos Settings

Specify these settings if the Backup Agent authenticates to hosts using Kerberos. For more information, see *Configure the Backup Agent for Kerberos*.

krb5Principal

Type: string

The Kerberos principal used by the agent. For example:

```
krb5Principal=mmsagent/myhost@EXAMPLE.COM
```

krb5Keytab

Type: string

The *absolute* path to Kerberos principal's keytab file. For example:

```
krb5Keytab=/etc/mongodb-mms/backup-agent.keytab
```

gsappiServiceName

Type: string

The default service name used by MongoDB is `mongodb` can specify a custom service name with the `gssapiServiceName` option.

Ops Manager Server SSL Settings

Advanced SSL settings used by the Backup Agent when communicating to the Ops Manager Backup Web Server.

sslTrustedMMSBackupServerCertificate

By default the Backup Agent will use the trusted root CAs installed on the system. If the agent cannot find the trusted root CAs, configure these settings manually.

If the Ops Manager Backup Server is using a self-signed SSL certificate this setting is required.

The path on disk that contains the trusted certificate authority certificates in PEM format. The agent will use this certificate to verify that the agent is communicating with the designated Ops Manager Backup Server. For example:

```
sslTrustedMMSBackupServerCertificate=/etc/mongodb-mms/mms-certs.pem
```

sslRequireValidMMSBackupServerCertificate

Type: boolean

You can disable certificate verification by setting this value to `false`. That configuration is only recommended for testing purposes as it makes connections susceptible to *man-in-the-middle* attacks.

Required Access for Backup Agent

On this page

If your MongoDB deployment enforces access control, the Ops Manager Backup Agent must authenticate to MongoDB as a user with the proper access. To authenticate, create a user with the appropriate roles in MongoDB. The following tutorials include instructions and examples for creating the MongoDB user:

- *Configure Backup Agent for MONGODB-CR*.
- *Configure Backup Agent for LDAP Authentication*.
- *Configure the Backup Agent for Kerberos*.

MongoDB user roles are separate from Ops Manager *user roles*.

Considerations

To authenticate to sharded clusters, create shard-local users on *each* shard and create cluster-wide users:

- Create cluster users while connected to the `mongos`; these credentials persist to the config servers.
- Create shard-local users by connecting directly to the replica set for each shard.

MongoDB 3.0 and Later

To backup MongoDB instances running 3.0 and later, the Backup Agent must authenticate as a user with the following role:

Required Role	
`backup` role on the `admin` database	

MongoDB 2.6

To backup MongoDB 2.6 release series instances, the Backup Agent must be able to authenticate to with the following roles:

Required Role	
`clusterAdmin` role on the `admin` database	
`readAnyDatabase` role on the `admin` database	
`userAdminAnyDatabase` role on the `admin` database	
`readWrite` role on the `admin` database	
`readWrite` role on the `local` database	

MongoDB 2.4

To backup MongoDB 2.4 release series instances, the Backup Agent must be able to authenticate to the database with a user that has specified `roles` and `otherDBRoles`. Specifically, the user must have the following roles:

Required Role	
`clusterAdmin` role on the `admin` database	
`readAnyDatabase` role on the `admin` database	
`userAdminAnyDatabase` role on the `admin` database	

And the following `otherDBRoles`:

Required Role	
`readWrite` role on the `local` database	
`readWrite` role on the `admin` database	
`readWrite` role on the `config` database	

Authentication Mechanisms

To authenticate, create the user in MongoDB with the appropriate access. The authentication method that the MongoDB deployment uses determines how to create the user as well as determine any additional agent configuration:

- For MONGODB-CR (MongoDB Challenge-Response) authentication, see *Configure Backup Agent for MONGODB-CR*.
- For LDAP authentication, see *Configure Backup Agent for LDAP Authentication*.
- For Kerberos authentication, see *Configure the Backup Agent for Kerberos*.

Configure Backup Agent for Access Control

If your MongoDB deployment enforces access control, the Backup Agent must authenticate to MongoDB as a user with the proper access.

Configure for MONGODB-CR Procedure to configure the Backup Agent for MongoDB deployments using MONGODB-CR authentication.

Configure for LDAP Procedure to configure the Backup Agent for MongoDB deployments using LDAP authentication.

Configure for Kerberos Procedure to configure the Backup Agent for MongoDB deployments using Kerberos authentication.

Configure Backup Agent for MONGODB-CR

On this page

- *MongoDB 2.6*
- *MongoDB 2.4*
- *Host Settings*

If your MongoDB deployment enforces access control, the Ops Manager Backup Agent must authenticate to MongoDB as a user with the proper access.

To authenticate using `MONGODB-CR`, create a user in the `admin` database with the appropriate roles in MongoDB.

MongoDB user roles are separate from Ops Manager *user roles* and are described in the MongoDB manual beginning with the Authorization page.

Considerations

To authenticate to sharded clusters, create shard-local users on *each* shard and create cluster-wide users:

- Create cluster users while connected to the `mongos`; these credentials persist to the config servers.
- Create shard-local users by connecting directly to the replica set for each shard.

Prerequisites

Connect to the `mongod` or `mongos` instance as a user with access to create users in the database. See db.createUser() method page for more information.

3.0 and Later

To backup MongoDB instances running 3.0 and later, create a user in the `admin` database with an operation that resembles the following:

```
use admin
db.createUser(
   {
     user: "<username>",
     pwd: "<password>",
     roles: [ { role: "backup", db: "admin" } ]
   }
)
```

See *Access Control for MongoDB 3.0* for more information on the required access.

MongoDB 2.6

To backup MongoDB 2.6 release series instances, create a user in the `admin` database with an operation that resembles the following:

```
use admin
db.createUser(
   {
     user: "<username>",
     pwd: "<password>",
     roles: [
```

```
        "clusterAdmin",
        "readAnyDatabase",
        "userAdminAnyDatabase",
        { role: "readWrite", db: "admin" },
        { role: "readWrite", db: "local" },
     ]
   }
)
```

See *Access Control for MongoDB 2.6* for more information on the required access.

MongoDB 2.4

To backup MongoDB 2.4 release series instances, create a user in the `admin` database with an operation that resembles the following:

```
use admin
db.addUser(
   {
     user: "<username>",
     pwd: "<password>",
     roles: [
        "clusterAdmin",
        "readAnyDatabase",
        "userAdminAnyDatabase"
     ],
     otherDBRoles: {
        local: ['readWrite'],
        admin: ['readWrite'],
        config: ['readWrite']
     }
   }
)
```

See *Access Control for MongoDB 2.4* for more information on the required access.

Host Settings

In addition to adding the agent as a MongoDB user, you must also specify the host's authentication settings. You can specify the host's authentication settings when *adding* the host, or you can *edit the settings* for an existing host.

Configure Backup Agent for LDAP Authentication

On this page

- *Host Settings*

Overview

If your MongoDB deployment enforces access control, the Backup Agent must authenticate to MongoDB as a user with the proper access.

Starting in version 2.6, MongoDB Enterprise provides an *LDAP (plain)* authentication mechanism that allows clients to authenticate to MongoDB deployments using LDAP. Backup Agents support authenticating to MongoDB instances using LDAP.

If your MongoDB deployment uses LDAP to authenticate users, to authenticate the Backup Agent, create a user in the `$external` database with the appropriate roles in MongoDB.

Considerations

You must configure LDAP authentication separately for the Backup Agent and for the Monitoring Agent.

You can configure LDAP authentication when activating backup or later by editing the backup configuration.

Prerequisites

There are additional authentication configuration requirements for Ops Manager Backup when using MongoDB 2.4 with authentication. See *Required Access for Backup Agent* for more information.

Create User in MongoDB

To monitor MongoDB 2.6 instances that are using LDAP authentication, add a user to the `$external` database in MongoDB with the appropriate roles. The `$external` database allows `mongod` to consult an external source (e.g. LDAP) to authenticate.

MongoDB 3.0 or later

```
db.getSiblingDB("$external").createUser(
    {
      user : "<username>",
      roles: [ { role: "backup", db: "admin" } ]
    }
)
```

MongoDB 2.6

```
db.getSiblingDB("$external").createUser(
   {
     user: "<username>",
     roles: [
        "clusterAdmin",
        "readAnyDatabase",
```

```
        "userAdminAnyDatabase",
        { role: "readWrite", db: "admin" },
        { role: "readWrite", db: "local" },
      ]
    }
)
```

See *Access Control for MongoDB 2.6* for more information on the required access.

Host Settings

In addition to adding the agent as a MongoDB user, you must also specify the host's authentication settings. You can specify the host's authentication settings when *adding* the host, or you can *edit the settings* for an existing host.

Configure the Backup Agent for Kerberos

On this page

MongoDB Enterprise provides support for Kerberos. Kerberos is a generic authentication protocol available starting from MongoDB Enterprise version 2.6. The Backup Agent can authenticate to hosts using Kerberos.

Warning: You must install the *prerequisite packages* on your servers before deploying MongoDB Enterprise on the servers.

You can use Ops Manager for Monitoring and Backup with Kerberos, but you cannot currently use Automation with Kerberos. Automation will support these features in the next major Ops Manager release.

Prerequisites

You must configure the Kerberos Key Distribution Center (KDC) to grant tickets that are valid for at least four hours. The Backup Agent takes care of periodically renewing the ticket. The KDC service provides session tickets and temporary session keys to users and computers.

There are additional authentication configuration requirements for Ops Manager Backup when using MongoDB 2.4 with authentication. See *Required Access for Backup Agent* for more information.

Create Kerberos Principal

If you are running both the Monitoring Agent and the Backup Agent on the same server, then both agents must connect as the same Kerberos Principal.

Step 1: Create or choose a Kerberos principal.

Create or choose a Kerberos principal for the Monitoring and/or Backup agent.

Step 2: Generate a keytab for the Kerberos principal.

Generate a keytab for the Kerberos principal and copy it to the system where the agent runs. Ensure the user that will run the agent is the same user that owns the keytab file.

Create MongoDB User for the Principal

If you are running both the Monitoring Agent and the Backup Agent on the same server, then both agents must connect as the same Kerberos Principal.

Kerberos Principal for the Backup Agent

Add a Kerberos principal, `<username>@<KERBEROS REALM>` or `<username>/<instance>@<KERBEROS REALM>`, to MongoDB in the `$external` database. Specify the Kerberos realm in all uppercase. The `$external` database allows `mongod` to consult an external source (e.g. Kerberos) to authenticate.

MongoDB 3.0 or Later

For MongoDB 3.0 or later, to add the principal for just the Backup Agent, use an operation that resembles the following:

```
use $external
db.createUser(
   {
     user: "<Kerberos Principal>",
     roles: [
        { role: "backup", db: "admin" }
     ]
   }
)
```

See *MongoDB 3.0 and Later* for more information on the required access for the Backup Agent.

MongoDB 2.6

For the MongoDB 2.6 release series, to add the principal for just the Backup Agent, use an operation that resembles the following:

```
use $external
db.createUser(
   {
     user: "<Kerberos Principal>",
     roles: [
        "clusterAdmin",
        "readAnyDatabase",
        "userAdminAnyDatabase",
        { role: "readWrite", db: "admin" },
        { role: "readWrite", db: "local" },
     ]
   }
)
```

See *MongoDB 2.6* for more information on the required access for the Backup Agent.

Kerberos Principal for the Monitoring Agent and Backup Agent

Add a Kerberos principal, `<username>@<KERBEROS REALM>` or `<username>/<instance>@<KERBEROS REALM>`, to MongoDB in the `$external` database. Specify the Kerberos realm in all uppercase. The `$external` database allows `mongod` to consult an external source (e.g. Kerberos) to authenticate.

For example, to add the same principal for both the Monitoring Agent and the Backup Agent, specify required access for both agents. The following example specifies access required to connect to MongoDB 3.0 or greater.

```
use $external
db.createUser(
   {
     user: "<Kerberos Principal>",
     roles: [
        { role: "clusterMonitor", db: "admin" },
        { role: "backup", db: "admin" }
     ]
   }
)
```

See *MongoDB 2.6* and *MongoDB 3.0 and Later* for more information on the required access for the Monitoring Agent and the Backup Agent.

Edit Agent Configuration File

Edit the `/etc/mongodb-mms/backup-agent.config` file.

Step 1: Set the `krb5Principal`

Set the `krb5Principal` to the name of the Kerberos principal. For example:

```
krb5Principal=mmsagent/instance@EXAMPLE.COM
```

Step 2: Set the `krb5Keytab`

Set the `krb5Keytab` value to the complete absolute path of the keytab file. For example:

```
krb5Keytab=/etc/mongodb-mms/mmsagent.keytab
```

Step 3: Restart the agent.

Host Settings

In addition to adding the agent as a MongoDB user, you must also specify the host's authentication settings. You can specify the host's authentication settings when *adding* the host, or you can *edit the settings* for an existing host.

Configure Kerberos Environment

Step 1: Create or configure the `/etc/kerb5.conf` file on the system to integrate this host into your Kerberos environment.

Step 2: Ensure the `kinit` binary is available at the `/user/bin/kinit` path.

Configure Backup Agent for SSL

On this page

Overview

If your MongoDB deployment uses SSL, then you must configure the Backup Agent to use SSL to connect to your deployment's `mongod` and `mongos` instances.

Configuring the agent to use SSL involves specifying which certificate to use to sign MongoDB certificates and turning on the SSL option for the MongoDB instances in Ops Manager.

Prerequisite

To configure the agent to use SSL, you must have a trusted CA certificate that signed the MongoDB instance's certificate.

Procedures

Connections between Agents and MongoDB Instances

To use SSL for the Backup Agent's connection to a MongoDB host, specify the host's SSL settings when *adding the host* or by *editing the host's settings*.

Then configure the agent to use SSL:

Step 1: Specify path to trusted CA certificate.

Edit the *Backup Agent configuration file* to set the `sslTrustedServerCertificates` field to the path of a file containing one or more certificates in PEM format. For example:

```
sslTrustedServerCertificates=/path/to/mongodb-certs.pem
```

The agent configuration file is located in either the agent install directory or the `/etc/mongodb-mms/` directory, depending on your operating system.

By default, to connect to MongoDB instances using SSL requires a valid trusted certificate. For testing purposes, however, you can set the *`sslRequireValidServerCertificates`* setting to `False` to bypass this check. This configuration is **not** recommended for production use as it makes the connection insecure.

For additional information on these settings, see *MongoDB SSL Settings*.

Step 2: Restart agent.

Connections between Agents and Ops Manager

To ensure that the Backup Agents use SSL when connecting to Ops Manager, Configure Ops Manager to use SSL for all connections. The *Configure SSL Connections to Ops Manager* tutorial describes how to set up Ops Manager to run over HTTPS.

Starting with Ops Manager 1.4, the Backup Agent validates the SSL certificate of the Ops Manager server by default.

If you are not using a certificate signed by a trusted 3rd party, you must configure the Backup Agent to trust the Ops Manager server.

To specify a self-signed certificate of the Ops Manager server that the Backup Agent should trust:

Step 1: Copy your PEM certificate to `/etc/mongodb-mms/`.

Issue the following sequence of commands:

```
sudo cp -a mms-ssl-unified.crt /etc/mongodb-mms/
sudo chown mongodb-mms-backup-agent:mongodb-mms-backup-agent /etc/mongodb-mms/mms-ssl-
↪unified.crt
sudo chmod 600 /etc/mongodb-mms/mms-ssl-unified.crt
```

Step 2: Edit the following parameter in the Backup Agent configuration file.

For example:

```
sslTrustedMMSBackupServerCertificate=/etc/mongodb-mms/mms-ssl-unified.crt
```

Step 3: Restart the Backup Agent for the configuration update to take effect.

Start or Stop the Backup Agent

On this page

Overview

For maintenance or troubleshooting purposes, you may want to temporarily shut down or restart Ops Manager Backup's Backup Agent. However, for proper operation of Ops Manager Backup your Ops Manager group must have at least one Backup Agent running. The group needs only one Backup Agent.

Procedures

Start the Backup Agent

The procedure to *Install the Backup Agent* includes a step to start the agent. If you must restart the agent, use the following procedure.

Start an Agent Installed with an `rpm` Package

If you installed the Backup Agent using an `rpm` package, such as on RHEL, CentOS, or SUSE, issue the following command to start the agent:

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

Start an Agent Installed with a `deb` Package

If you installed the Backup Agent using a `deb` package, as on Ubuntu, issue the following command to start the agent:

```
sudo start mongodb-mms-backup-agent
```

Start an Agent Installed with a `tar` File

Use this command if you installed to Linux or OSX using a `tar` file. Issue the following command from the directory to which you installed the Backup Agent:

```
nohup ./mongodb-mms-backup-agent >> backup-agent.log 2>&1 &
```

Start the Backup Agent on Windows

In Windows `Control Panel`, open `Administrative Tools` and then `Services`. In the list of services, select `MMS Backup Agent`. Select the `Action` menu and select `Start`.

Stop the Backup Agent

If you use Ops Manager Backup, you must have a Backup Agent running to ensure up-to-date backup data.

Stop an Agent Installed with an `rpm` Package

If you installed the Backup Agent using an `rpm` package, such as on RHEL, CentOS, or SUSE, issue the following command to stop the agent:

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

Stop an Agent Installed with a `deb` Package

If you installed the Backup Agent using a `deb` package, as on Ubuntu, issue the following command to stop the agent:

```
sudo stop mongodb-mms-backup-agent
```

Stop an Agent Installed with a `tar` File

If you installed to a Linux system or OSX using a `tar` file, issue the following command to stop the Backup Agent:

```
pkill -f mongodb-mms-backup-agent
```

Stop the Backup Agent on Windows

In Windows `Control Panel`, open `Administrative Tools` and then `Services`. In the list of services, select `MMS Backup Agent`. Select the `Action` menu and select `Stop`.

If you receive a message that your Backup Agent is out of date, make sure you are running an upgradeable version of the Backup Agent. If you are running the version of the Backup Agent named `MongoDBBackup`, you must remove it before upgrading. To check if you are running `MongoDBBackup`, issue the following command in an Administrative command prompt:

```
sc query MongoDBBackup
```

If the command returns a result, you must remove the `MongoDBBackup` agent. To remove it, issue the following:

```
sc delete MongoDBBackup
```

Remove the Backup Agent from Ops Manager

Ops Manager displays active Backup Agents on the *Agents* page in in the *Administration* tab. The page displays agents that have been active in the last 24 hours. If an agent fails to report to Ops Manager for more than 24 hours, Ops Manager removes the agent from the *Agents* page.

To remove a Backup Agent **from |mms|**, *stop the agent* and then wait 24 hours.

To delete the Backup Agent **from a Linux or OSX server**, *stop the agent* and then remove the `mongodb-mms-backup-agent` file from the `/usr/bin` directory. If you installed the agent using a `tar.gz` file, the agent will be in the directory you chose during installation.

To delete the Backup Agent **from a Windows server**, *stop the agent* and then use the Windows program uninstaller to remove the `MMS Backup Agent` program.

12.5 Audit Events

On this page

Ops Manager maintains an audit log of key operations that users and administrators perform in the system. Unless otherwise stated, the *Events* page aggregates and displays these events.

User Audits

`JOINED_GROUP`
A user joined a Group.

This audit is not supported if using LDAP authentication for Ops Manager users

`REMOVED_FROM_GROUP`
A user was removed from a Group.

`INVITED_TO_GROUP`
A user was invited to a Group

`MULTI_FACTOR_AUTH_RESET_EMAIL_SENT_AUDIT`
`MULTI_FACTOR_AUTH_RESET_EMAIL_SENT_AUDIT` is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user requested and was sent an email with a link that will allow them to reset their 2 factor authentication.

`MULTI_FACTOR_AUTH_RESET_AUDIT`
`MULTI_FACTOR_AUTH_RESET_AUDIT` is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user reset their two factor authentication.

`MULTI_FACTOR_AUTH_UPDATED_AUDIT`
`MULTI_FACTOR_AUTH_UPDATED_AUDIT` is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user updated their 2FA using the form in *My Profile*.

`PASSWORD_RESET_EMAIL_SENT_AUDIT`
`PASSWORD_RESET_EMAIL_SENT_AUDIT` is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user requested and was sent an email with a link that will allow them to reset their password.

PASSWORD_RESET_AUDIT
PASSWORD_RESET_AUDIT is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user successfully reset their password via the *reset password* flow.

PASSWORD_UPDATED_AUDIT
PASSWORD_UPDATED_AUDIT is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user successfully updated their password using the form in *My Profile*.

USER_EMAIL_ADDRESS_CHANGED_AUDIT
USER_EMAIL_ADDRESS_CHANGED_AUDIT is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user changed their email address.

USER_ROLES_CHANGED_AUDIT
A user's roles in a particular Group were changed.

SUCCESSFUL_LOGIN_AUDIT
SUCCESSFUL_LOGIN_AUDIT is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user successfully authenticated with their username and password.

UNSUCCESSFUL_LOGIN_AUDIT
UNSUCCESSFUL_LOGIN_AUDIT is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A user entered a valid username, but an invalid password.

ACCOUNT_LOCKED_AUDIT
Ops Manager locked a user's account from the system, as a result of manual action by the administrator, or because of a change in account locking policies.

ACCOUNT_UNLOCKED_AUDIT
An administrator unlocked a user's account.

USER_CREATED_AUDIT
USER_CREATED_AUDIT is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A new user was created.

Host Audits

DELETE_HOST_AUDIT
A host was suppressed by a user.

REACTIVATE_HOST_AUDIT
A host was manually reactivated by a user.

DEACTIVATE_HOST_AUDIT
A host was deactivated by the system.

ADD_HOST_AUDIT
A new host was added by a user, or auto-discovered by the system.

UNDELETE_HOST_AUDIT
A previously suppressed host was un-suppressed by a user.

HIDE_AND_DISABLE_HOST_AUDIT
The system determined that a host was a duplicate by the system, and hid that host from the interface.

DB_PROFILER_ENABLE_AUDIT
Database profiling was enabled for a host

DB_PROFILER_DISABLE_AUDIT
Database profiling data collection was disabled for a host

HOST_IP_CHANGED_AUDIT
A change in IP address was detected for a host.

Alert Config Audits

ALERT_ACKNOWLEDGED_AUDIT
A user acknowledged an open alert.

ALERT_UNACKNOWLEDGED_AUDIT
A user un-acknowledged an open alert.

ALERT_CONFIG_DISABLED_AUDIT
An alert configuration was disabled.

ALERT_CONFIG_ENABLED_AUDIT
An alert configuration was enabled.

ALERT_CONFIG_ADDED_AUDIT
An alert configuration was added.

ALERT_CONFIG_DELETED_AUDIT
An alert configuration was deleted.

ALERT_CONFIG_CHANGED_AUDIT
An alert configuration was edited.

Backup Audits

RS_STATE_CHANGED_AUDIT
A user started, stopped, or terminated backup for a replica set. is started, stopped. or terminated by a user

CLUSTER_STATE_CHANGED_AUDIT
A user started, stopped or terminated backup for a sharded cluster.

RESTORE_REQUESTED_AUDIT
A restore was requested.

SYNC_REQUIRED_AUDIT
A user initiates a resync of a replica set or config server.

CLUSTERSHOT_DELETED_AUDIT
A user deletes a clustershot (e.g. a cluster checkpoint.)

SNAPSHOT_DELETED_AUDIT
A user delted a snapshot for a replica set.

RS_CREDENTIAL_UPDATED_AUDIT
A user updates the authentication credentials for a replica set.

CLUSTER_CREDENTIAL_UPDATED_AUDIT
A user updates the authentication credentials for a sharded cluster.

RS_BLACKLIST_UPDATED_AUDIT
A user updates the excluded namespaces for a replica set.

CLUSTER_BLACKLIST_UPDATED_AUDIT
A user updates the excluded namespaces for a sharded cluster.

RS_SNAPSHOT_SCHEDULE_UPDATED_AUDIT
A user updates the snapshot schedule for a replica set.

CLUSTER_SNAPSHOT_SCHEDULE_UPDATED_AUDIT
A user updates the snapshot schedule for a sharded cluster.

CLUSTER_CHECKKPOINT_UPDATED_AUDIT
A user updates the checkpoint schedule for a sharded cluster.

Group Audits

GROUP_DELETED
`GROUP_DELETED` is only visible in the *Admin* section in the *General* tab on the *Audits* page.

A Group was deleted.

GROUP_CREATED
A new Group was created.

12.6 Monitoring Reference

On this page

This document contains references of the different types of hosts, databases, and other statuses that may occur in Monitoring.

Host Types

The possible values for the "Type" column in the *Deployment* page are:

- primary
- secondary
- standalone
- master
- slave
- unknown
- recovering

The "Host Type" selector on the advanced dashboard creator also includes:

- conf

- mongos

Note: The host type column may also have the value "no data," which means that Monitoring has not received any data from the Monitoring Agent for this host. For possible causes, see *Troubleshooting*.

Host Process Types

Monitoring can monitor the process types:

- `mongod` database processes
- `mongod` arbiter processes
- `mongos`
- Monitoring Agents

Event Types

Types of events in the Events section of the Ops Manager console:

- new host
- restart
- upgrade

Alert Types

The available alert types are:

- Old Host Version
- Host Down
- Agent Down
- Now Secondary
- Now Primary

Chart Colors

- A *red bar* indicates a server restart.
- A *purple bar* indicates the server is now a primary.
- A *yellow bar* indicates the server is now a secondary.

Status Page

- cpu time
- db storage
- page faults

- repl lag
- replica
- network
- cursors
- queues
- connections
- background flush avg
- lock %[1]
- btree
- non-mapped virtual memory
- memory
- asserts
- opcounters-repl
- opcounters

DB Stats Page

- collections
- objects
- average object size
- data size
- storage size
- num extents
- indexes
- index size
- file size

Database Commands Used by the Monitoring Agent

- serverStatus
- buildinfo
- getCmdLineOpts
- connPoolStats
- _isSelf
- getParameter
- ismaster

[1] For versions of MongoDB after 2.1.1, this chart has a drop-down menu next to the tile that lists available databases, including "global" to represent the global lock for this host. Select a database to see its lock utilization. See the documentation of lock reporting in serverStatus for more information.

- getShardVersion
- netstat
- replSetGetStatus
- shards.find
- mongos.find
- config.chunks.group
- oplog.find
- collstats - oplog.rs
- sources.find (slave)
- config.settings.find
- dbstats
- db.locks

12.7 Supported Browsers

To use Ops Manager, ensure that your browser is one of the following supported browsers:

- Chrome 8 and greater.
- Firefox 12 and greater.
- IE 9 and greater.
- Safari 6 and greater.

Ops Manager will display a warning on non-supported browsers.

12.8 Advanced Options for MongoDB Deployments

On this page

Overview

The following `mongod` and `mongos` configuration options are available through the Ops Manager *Advanced Options* field when you deploy MongoDB. You select advanced options when deploying *replica sets*, *sharded clusters*, and *standalone instances*.

Advanced Options

The Ops Manager *Advanced Options* map to the MongoDB configuration options as described here.

Network and HTTP

- *maxConns*: `net.maxIncomingConnections`
- *jsonp*: `net.http.JSONPEnabled`
- *nohttpinterface*: `net.http.enabled`
- *rest*: `net.http.RESTInterfaceEnabled`

Operation Profiling

- *profile*: `operationProfiling.mode`
- *slowms*: `operationProfiling.slowOpThresholdMs`

Process Management

- *pidfilepath*: `processManagement.pidFilePath`

Replication

- *oplogSize*: `replication.oplogSizeMB`

Storage and Journal

- *directoryperdb*: `storage.directoryPerDB`
- *noprealloc*: `storage.preallocDataFiles`
- *nssize*: `storage.nsSize`
- *smallfiles*: `storage.smallFiles`
- *syncdelay*: `storage.syncPeriodSecs`
- *journalCommitInterval*: `storage.journal.commitIntervalMs`
- *nojournal*: `storage.journal.enabled`

System Log

- *logappend*: `systemLog.logAppend`
- *quiet*: `systemLog.quiet`
- *syslog*: `systemLog.destination`

12.9 Automation Configuration

On this page

Overview

The Automation Agent uses an automation configuration to determine the desired state of a MongoDB deployment and to effect changes as needed. If you modify the deployment through the Ops Manager web interface, you never need manipulate this configuration.

If you are using the Automation Agent without Ops Manager, you can construct and distribute the configuration file manually.

Fields

Optional fields are marked as such.

A field that takes a `<number>` as its value can take integers and floating point numbers.

version

```
"version" : <integer>
```

Name	Type	Description
version	integer	The version of the configuration file.

agentVersion

```
"agentVersion" : {
    "name" : <string>,
    "directoryUrl" : <string>
}
```

Name	Type	Description
agentVersion	object	*Optional* The version of the Automation Agent to run. If the running version does not match this setting, the Automation Agent downloads the specified version, shuts itself down, and starts the new version.
- name	string	The desired version of the Automation Agent (e.g. "1.8.1.1042-1").
- directoryUrl	string	The URL from which to download Automation Agent.

monitoringVersions

```
"monitoringVersions" : [
    {
        "name" : <string>,
        "hostname" : <string>,
        "urls" : {
            <platform1> : {
```

```
                <build1> : <string>,
                ...,
                "default" : <string>
            },
            ...
        },
        "baseUrl" : <string>,
        "logPath" : <string>,
        "logRotate" : {
            "sizeThresholdMB" : <number>,
            "timeThresholdHrs" : <integer>,
            "numUncompressed": <integer>,
            "percentOfDiskspace" : <number>
        }
    },
    ...
]
```

Name	Type	Description
monitoringVersions	array of objects	*Optional*. Objects that define version information for each Monitoring Agent.
- name	string	The desired version of the Monitoring Agent (e.g. "2.9.1.176-1"). To monitor or back up MongoDB 3.0 deployments, you must install Ops Manager 1.6 or higher. To monitor a MongoDB 3.0 deployment, you must also run Monitoring Agent version 2.7.0 or higher.
- hostname	string	The hostname of the machine that runs the Monitoring Agent. If the Monitoring Agent is not running on the machine, Ops Manager installs the agent from the location specified in `monitoringVersions.urls`.
- urls	object	The platform- and build-specific URLs from which to download the Monitoring Agent.
- - <platform>	object	This field has a name that identifies an operating system and optionally a version. The field contains an object with key-value pairs, where each key is either the name of a build or `default` and each value is a URL for downloading the Monitoring Agent. The object must include the `default` key set to the default download URL for the platform.
- baseUrl	string	The base URL used for the `mmsBaseUrl` setting in the *Monitoring Agent Configuration*.
- logPath	string	*Optional*. The directory where the agent stores its logs. The default is to store logs in `/dev/null`.
- logRotate	object	*Optional*. Enables log rotation for the MongoDB logs for a process.
- - sizeThresholdMB	number	The maximum size in MB for an individual log file before rotation.
- - timeThresholdHrs	integer	The maximum time in hours for an individual log file before rotation.
- - numUncompressed	integer	*Optional*. The maximum number of total log files to leave uncompressed, including the current log file. The default is `5`.
- - percentOfDiskspace	number	*Optional*. The maximum percentage of total disk space all log files should take up before deletion. The default is `.02`.

backupVersions

```
"backupVersions" : [
    {
```

```
        "name" : <string>,
        "hostname" : <string>,
        "urls" : {
            <platform1> : {
                <build1> : <string>,
                ...,
                "default" : <string>
            },
            ...
        },
        "baseUrl" : <string>,
        "logPath" : <string>,
        "logRotate" : {
            "sizeThresholdMB" : <number>,
            "timeThresholdHrs" : <integer>,
            "numUncompressed": <integer>,
            "percentOfDiskspace" : <number>
        }
    },
    ...
],
```

Name	Type	Description
backupVersions	array of objects	*Optional*. Objects that define version information for each Backup Agent.
- name	string	The desired version of the Backup Agent (e.g. "3.1.1.263-1").
- hostname	string	The hostname of the machine that runs the Backup Agent. If the Backup Agent is not running on the machine, Ops Manager installs the agent from the location specified in `backupVersions.urls`.
- urls	object	The platform- and build-specific URLs from which to download the Backup Agent.
- - <platform>	object	This field has a name that identifies an operating system and optionally a version. The field contains an object with key-value pairs, where each key is either the name of a build or `default` and each value is a URL for downloading the Backup Agent. The object must include the `default` key set to the default download URL for the platform.
- baseUrl	string	The base URL used for the `mothership` and `https` settings in the *Backup Agent Configuration*. For example, for `"baseUrl"=https://cloud.mongodb.com`, the backup configuration fields would have these values: `mothership=api-backup.mongodb.com` and `https"=true`.
- logPath	string	*Optional*. The directory where the agent stores its logs. The default is to store logs in `/dev/null`.
- logRotate	object	*Optional*. Enables log rotation for the MongoDB logs for a process.
- - sizeThresholdMB	number	The maximum size in MB for an individual log file before rotation.
- - timeThresholdHrs	integer	The maximum time in hours for an individual log file before rotation.
- - numUncompressed	integer	*Optional*. The maximum number of total log files to leave uncompressed, including the current log file. The default is `5`.
- - percentOfDiskspace	number	*Optional*. The maximum percentage of total disk space all log files should take up before deletion. The default is `.02`.

processes

```
"processes" : [
    {
        "name" : <string>,
        "processType" : <string>,
        "version" : <string>,
        "<args>" : <object>,
        "disabled" : <Boolean>,
        "manualMode" : <Boolean>,
        "hostname" : <string>,
        "cluster": <string>,
        "numCores": <integer>,
        "logRotate" : {
            "sizeThresholdMB" : <number>,
            "timeThresholdHrs" : <integer>,
            "numUncompressed": <integer>,
            "percentOfDiskspace" : <number>
        },
        "authSchemaVersion": <integer>,
        "alias": <string>
    },
    ...
]
```

Name	Type	Description
processes	array of objects	The `processes` array contains objects that define the `mongos` and `mongod` instances that Ops Manager monitors. Each object defines a different instance.
- name	string	A unique name to identify the instance.
- processType	string	Either `mongod` or `mongos`.
- version	string	The name of the `mongoDbVersions` specification used with this instance.
- <args>	object	This field is named either `args2_6`, for MongoDB versions 2.6 and higher (including 3.0 and higher), or `args2_4`, for versions 2.4 and earlier. The field contains a MongoDB configuration object in the format appropriate to the version. For information on format and supported MongoDB options, see *supported configuration options*.
- disabled	Boolean	*Optional*. Set to `true` to shut down the process.
- manualMode	Boolean	*Optional*. Set to `true` to operate this process in manual mode. The Automation Agent will take no actions on the process.
- hostname	string	*Optional*. The name of the host this process should run on. This defaults to `localhost`.
- cluster	string	*Optional*. Required for a `mongos`. The name of the cluster. This must correspond to the `sharding.name` field in the `sharding` array for the `mongos`.
- numCores	integer	*Optional*. The number of cores the process should be bound to. The Automation Agent will spread processes out across the cores as evenly as possible.
- logRotate	object	*Optional*. Enables log rotation for the MongoDB logs for a process.
- - sizeThresholdMB	number	The maximum size in MB for an individual log file before rotation.
- - timeThresholdHrs	integer	The maximum time in hours for an individual log file before rotation.
- - numUncompressed	integer	*Optional*. The maximum number of total log files to leave uncompressed, including the current log file. The default is `5`.
- - percentOfDiskspace	number	*Optional*. The maximum percentage of total disk space all log files should take up before deletion. The default is `.02`.
- authSchemaVersion	integer	*Optional*. The schema version of the user credential objects. This should match all other elements of the `processes` array that belong to the same cluster. The possible values are `1`, `3`, and `5`. The default is `3` for `2.6` clusters and `1` for `2.4` clusters.
- alias	string	*Optional*. A hostname alias (often a DNS CNAME) for the server on which the process runs. If an alias is specified, the Automation Agent prefers the alias over the host specified in `processes.hostname` when connecting to the server. You can also specify this alias in `replicaSets.host` and `sharding.configServer`.

replicaSets

```
"replicaSets" : [
    {
        "_id" : <string>,
        "version" : <integer>
        "members" : [
            {
                "_id" : <integer>,
                "host" : <string>
            },
```

```
            ...
        ],
        "force" : {
            "currentVersion" : <integer>
        }
    },
    ...
]
```

Name	Type	Description
replicaSets	array of objects	*Optional*. Objects that define the configuration of each replica set. The Automation Agent uses the values in this array to create valid replica set configuration documents. The agent regularly checks that replica sets are configured correctly. If a problem occurs, the agent reconfigures the replica set according to its configuration document. The array can contain the following top-level fields from a replica set configuration document: `_id`; `version`; and `members`. For more information on the configuration documents, see replSetGetConfig in the MongoDB manual.
- _id	string	The name of the replica set.
- version	integer	The version of the replica set configuration.
- members	array of objects	Objects that define each member of the replica set. The `members.host` field must specify the host's name as listed in `processes.name`. The Automation Agent expands the `host` field to create a valid replica set configuration. For more information on `members` objects, see replSetGetConfig in the MongoDB manual.
- force	object	*Optional*. An object that contains the `currentVersion` field set to a version number. Automation will force a reconfiguration of the replica set if and only if the value of `currentVersion` equals the current version of the replica set. You can use `force` to reconfigure a replica set that has lost members and can't reach a majority of votes.

sharding

```
"sharding" : [
    {
        "name" : <string>,
        "configServer" : [ <string>, ... ],
        "collections" : [
            {
                "_id" : <string>,
                "key" : [
                    [ shard key ],
                    [ shard key ],
                    ...
                ]
            },
            ...
        ],
        "shards" : [
            {
                "_id" : <string>,
                "rs" : <string>
            },
```

```
            ...
        ]
    },
    ...
]
```

Name	Type	Description
sharding	array of objects	*Optional*. Objects that define the configuration of each sharded cluster. Each object in the array contains the specifications for one cluster. The Automation Agent regularly checks each cluster's state against the specifications. If the specification and cluster don't match, the agent will change the configuration of the cluster, which might cause the balancer to migrate chunks.
- name	string	The name of the cluster. This must correspond with the value in `processes.cluster` for a `mongos`.
- configServer	array	String values that provide the names of each config server's hosts. The host names are the same names as are used in each host's `processes.name` field.
- collections	array of objects	Objects that define the sharded collections and their shard keys.
- - _id	string	The namespace of the sharded collection. The namespace is the combination of the database name and the name of the collection. For example, `testdb.testcoll`.
- - key	array of arrays	The collection's shard keys. This "array of arrays" contains a single array if there is a single shard key and contains multiple arrays if there is a compound shard key.
- shards	array of objects	Objects that define the cluster's shards.
- - _id	string	The name of the shard.
- - rs	string	The name of the shard's replica set, as specified in the `replicaSets._id` field.

balancer

```
"balancer": {
    "<clusterName1>": <object>,
    "<clusterName2>": <object>,
    ...
}
```

Name	Type	Description
balancer	object	*Optional*. This object contains fields named according to clusters, each field containing an object with the desired balancer settings for the cluster. The object uses the `stopped` and `activeWindow` fields, as described in the procedure to schedule the balancing window in this tutorial in the MongoDB manual.

auth

```
"auth" : {
    "autoUser": <string>,
    "autoPwd": <string>,
    "key" : <string>,
    "keyfile" : <string>,
    "usersDeleted" : [
        {
            "user" : <string>,
            "dbs" : [ <string>, ... ],
            "allDbs" : <Boolean>
        },
        ...
    ],
    "usersWanted" : [
        {
            "db" : <string>,
            "user" : <string>,
            "roles" : [ <string>, ... ],
            "pwd" : <32-character hex string>,
            "initPwd" : <string>,
            "userSource" : <string>,
            "otherDBRoles" : {
                <string> : [ <string>, ....],
                ...
            }
        },
        ...
    ]
}
```

Name	Type	Description
auth	object	*Optional*. Defines authentication-related settings.
- autoUser	string	The username that the Automation agent uses when connecting to an instance.
- autoPwd	string	The password that the Automation agent uses when connecting to an instance.
- disabled	boolean	*Optional*. Indicates if auth is disabled. If not specified, `disabled` defaults to `false`.
- key	string	The contents of the key file that Ops Manager uses to authenticate to the MongoDB processes. The `key` is not required if `disabled` is `true`.
- keyfile	string	The path and name of the key file that Ops Manager uses to authenticate to the MongoDB processes. The `keyfile` is not required if `disabled` is `true`.
- usersDeleted	array of objects	*Optional*. Objects that define the authenticated users to be deleted from specified databases or from all databases. This array must contain two fields: the `auth.usersDeleted.user` field and then either the `auth.usersDeleted.dbs` or the `auth.usersDeleted.allDbs` field.
- - user	string	The user's name.
- - dbs	array	String values that list the names of the databases from which the authenticated user is to be deleted. If you use this field, **do not** use the `auth.usersDeleted.allDbs` field.
- - allDbs	Boolean	If set to `true`, the authenticated user is deleted from all databases. If you use this field, **do not** use the `auth.usersDeleted.dbs` field.
- usersWanted	array of objects	*Optional*. Contains objects that define authenticated users to add to specified databases. Each object must have the `auth.usersWanted.db`, `auth.usersWanted.user`, and `auth.usersWanted.roles` fields, and then have exactly one of the following fields: `auth.usersWanted.pwd`, `auth.usersWanted.initPwd`, or `auth.usersWanted.userSource`.
- - db	string	The database to which to add the user.
- - user	string	The name of the user.
- - roles	array	String values that list the roles to be assigned the user from the user's database, which is specified in `auth.usersWanted.db`.
- - pwd	32-character hex string	The *MONGODB-CR* hash of the password assigned to the user. If you set this field, **do not** set the `auth.usersWanted.initPwd` or `auth.usersWanted.userSource` fields.
- - initPwd	string	An initial cleartext password assigned to the user. If you set this field, **do not** set the `auth.usersWanted.pwd` or `auth.usersWanted.userSource` fields.
- - userSource	string	If you use MongoDB version 2.4, you can use this field to specify the database that contains the user's credentials. See the Privilege Documents page in the MongoDB 2.4 manual. If you set this field, **do not** set the `auth.usersWanted.pwd` or `auth.usersWanted.initPwd` fields.
- - otherDBRoles	object	*Optional*. If the `auth.usersWanted.db` field specifies `admin` as the user's database, then this object can assign to the user roles from other databases as well. The object contains key-value pairs where the key is the name of the database and the value is an array of string values that list the roles be assigned from that database.

ssl

SSL is available only in MongoDB Enterprise or a build of MongoDB compiled with SSL support.

```
"ssl" : {
    "CAFilePath" : <string>
}
```

Name	Type	Description
ssl	object	*Optional*. Enables SSL for encrypting connections. SSL is available only in MongoDB Enterprise or a build of MongoDB compiled with SSL support.
- CAFilePath	string	The path to the certificate used to authenticate through SSL.

roles

```
"roles" : [
    {
        "role" : <string>,
        "db" : <string>,
        "privileges" : [
            {
                "resource" : { ... },
                "actions" : [ <string>, ... ]
            },
            ...
        ],
        "roles" : [
            {
                "role" : <string>,
                "db" : <string>
            }
        ]

    },
    ...
]
```

Name	Type	Description
roles	array of objects	*Optional*. The `roles` array contains objects that describe the cluster's user-defined roles. Each object describes a different user-defined role. Objects in this array contain the same fields as documents in the :manual:‘ system roles collection </reference/system-roles-collection>‘, except for the `_id` field, which is not included here.

mongoDbVersions

The `mongoDbVersions` array defines specification objects for the MongoDB instances found in the `processes` array. Each MongoDB instance in the `processes` array must have a specification object in this array.

```
"mongoDbVersions" : [
    {
        "name" : <string>,
```

```
        "builds" : [
            {
                "platform" : <string>,
                "url" : <string>,
                "gitVersion" : <string>,
                "bits" : <integer>,
                "win2008plus" : <Boolean>,
                "winVCRedistUrl" : <string>,
                "winVCRedistOptions" : [ <string>, ... ],
                "winVCRedistDll" : <string>,
                "winVCRedistVersion" : <string>
            },
            ...
        ],
    },
    ...
]
```

Name	Type	Description
mongoDbVersions	array of objects	The `mongoDbVersions` array is required and defines specification objects for the MongoDB instances found in the `processes` array. Each MongoDB instance in `processes` must have a specification object in `mongoDbVersions`.
- name	string	The name of the specification object. The specification object is attached to a MongoDB instance through the instance's `processes.version` field in this configuration file.
- builds	array of objects	Objects that define the builds for this MongoDB instance.
- - platform	string	The platform for this MongoDB instance.
- - url	string	The URL from which to download MongoDB for this instance.
- - gitVersion	string	The commit identifier that identifies the state of the code used to build the MongoDB process. The MongoDB buildInfo command returns the gitVersion identifier.
- - bits	integer	The processor's bus width. Specify either `64` or `32`.
- - win2008plus	Boolean	*Optional*. Set to `true` if this is a Windows build that requires either Windows 7 later or Windows Server 2008 R2 or later.
- - winVCRedistUrl	string	*Optional*. The URL from which the required version of the Microsoft Visual C++ redistributable can be downloaded.
- - winVCRedistOptions	array	*Optional*. String values that list the command-line options to be specified when running the Microsoft Visual C++ redistributable installer. Each command-line option is a separate string in the array.
- - winVCRedistDll	string	*Optional*. The name of the Microsoft Visual C++ runtime DLL file that the agent will check to determine if a new version of the Microsoft Visual C++ redistributable is needed.
- - winVCRedistVersion	string	*Optional*. The minimum version of the Microsoft Visual C++ runtime DLL that must be present to skip over the installation of the Microsoft Visual C++ redistributable.

options

```
"options" : {
    "downloadBase" : <string>
}
```

Name	Type	Description
options	object	
- downloadBase	string	The path to the directory where automatic version downloads are targeted and scripts for starting processes are created.

kerberos

```
"kerberos": {
    "serviceName": <string>
}
```

Name	Type	Description
kerberos	object	*Optional*. A key-value pair that defines the kerberos service name agents use to authenticate via kerberos.
- serviceName	string	The service name agents use to authenticate to a `mongod` or `mongos` via kerberos. This name is also used to set the `saslServiceName` option in a MongoDB configuration, as described on the MongoDB Server Parameters page in the MongoDB manual.

indexConfigs

```
"indexConfigs" : [
    {
        "key" : [
            [ <string> : <val> ],
            ...
        ],
        "rsName" : <string>,
        "dbName" : <string>,
        "collectionName" : <string>
    },
    ...
]
```

Name	Type	Description
indexConfigs	array of objects	*Optional*. Objects that define specific indexes to be built for specific replica sets.
- key	array of arrays	The index's keys. This "array of arrays" contains a single array if the index has just one key.
- rsName	string	The replica set that the index is build on.
- dbName	string	The database the index applies to.
- collectionName	string	The collection the index applies to.

12.10 Supported MongoDB Options for Automation

On this page

Overview

The `processes.<args>` object in an *automation configuration* specifies the configuration options for each MongoDB instance. The supported options depend on the version of MongoDB.

MongoDB 2.6 and Later Configuration Options

The `processes.args2_6` object applies to MongoDB versions 2.6 and higher (including 3.0 and higher) and supports the following MongoDB options. The object uses the MongoDB configuration format.

The `processes.args2_6` object supports the following:

- `config`
- `net.bindIp`
- `net.http.JSONPEnabled`
- `net.http.RESTInterfaceEnabled`
- `net.http.enabled`
- `net.maxIncomingConnections`
- `net.port`
- `noscripting`
- `notablescan`
- `operationProfiling.mode`
- `operationProfiling.slowOpThresholdMs`
- `processManagement.fork`

- processManagement.pidFilePath
- replication.oplogSizeMB
- replication.replSet
- replication.replSetName
- replication.secondaryIndexPrefetch
- security.authorization
- security.keyFile
- setParameter.connPoolMaxConnsPerHost
- setParameter.connPoolMaxShardedConnsPerHost
- setParameter.enableTestCommands
- setParameter.enableLocalhostAuthBypass
- setParameter.failIndexKeyTooLong
- setParameter.internalQueryPlannerEnableIndexIntersection
- setParameter.logLevel
- setParameter.newCollectionsUsePowerOf2Sizes
- setParameter.releaseConnectionsAfterResponse
- setParameter.textSearchEnabled
- setParameter.ttlMonitorEnabled
- sharding.clusterRole
- sharding.configDB
- storage.dbPath
- storage.directoryPerDB
- storage.journal.commitIntervalMs
- storage.journal.enabled
- storage.nsSize
- storage.preallocDataFiles
- storage.quota.maxFilesPerDB
- storage.quota.enforced
- storage.smallFiles
- storage.syncPeriodSecs
- systemLog.destination
- systemLog.logAppend
- systemLog.path
- systemLog.quiet
- systemLog.timeStampFormat
- systemLog.verbosity

MongoDB 2.4 and Earlier Configuration Options

The `processes.args2_4` object applies to MongoDB versions 2.4 and earlier and supports the following MongoDB options. The object uses the 2.4 MongoDB configuration format.

The `processes.args2_4` object supports the following:

- `auth`
- `bind_ip`
- `config`
- `configdb`
- `configsvr`
- `dbpath`
- `directoryperdb`
- `fork`
- `journal`
- `journalCommitInterval`
- `jsonp`
- `keyFile`
- `logappend`
- `logpath`
- `maxConns`
- `nohttpinterface`
- `nojournal`
- `noprealloc`
- `noscripting`
- `nssize`
- `oplogSize`
- `pidfilepath`
- `port`
- `profile`
- `quiet`
- `quota`
- `quotaFiles`
- `replSet`
- `rest`
- `shardsvr`
- `slowms`
- `smallfiles`

- `syncdelay`
- `syslog`
- `v`
- `vv`
- `vv`

13 Release Notes

Ops Manager Server Changelog A record of changes to the Ops Manager Application.

Automation Agent Changelog A record of changes to the Automation Agent.

Monitoring Agent Changelog A record of changes to the Monitoring Agent.

Backup Agent Changelog A record of changes to the Backup Agent.

13.1 Ops Manager Server Changelog

Ops Manager Server 1.6.4

Released 2015-08-17

- Ops Manager no longer shuts down if the *Ops Manager Application Database* is unreachable. (This issue was erroneously reported as resolved in Ops Manager 1.6.3.)

Ops Manager Server 1.6.3

Released 2015-06-23

- Agent updates: *Automation Agent 1.4.18.1199-1*
- Added full support for restores of WiredTiger backups. Previously, Ops Manager only supported *SCP Individual File* restores for WiredTiger backups.
- Added optimization to prevent some Backup Daemon background tasks from doing excessive logging when databases are down.
- Fixed a user interface issue when displaying an empty Automation diff.

Ops Manager Server 1.6.2

Released 2015-04-28

- Fixed issue with grooms on a WiredTiger Backup Blockstore.
- Fixed a possible connection leak with the SCP Individual File restore type.
- LDAP users are now periodically synced with the LDAP server to prevent communications after a user is removed from a group.
- Fixed an issue with backups of MongoDB 3.0 `mongod` instances running with the `--setParameter failIndexKeyTooLong=0` option.

Ops Manager Server 1.6.1

Released 2015-03-26

- Upgraded Automation Agent to `1.4.15.999`. See: the *Automation Agent release notes*.
- **Security Update**: resolved an issue where users removed from LDAP groups were not always removed from corresponding Ops Manager groups. This upgrade is **highly recommended** for anyone using LDAP authentication.
- Selecting wildcards in the Version Manager is no longer supported when `automation.versions.source` is set to `` `local ``.
- Adds a 1 hour timeout to kill a Backup *head* database if it does not shutdown cleanly. You must perform a resync following a hard kill.
- Windows support for Backup Daemon using Windows 64-bit 2008 R2+ MongoDB builds.
- Fix for Backups stored in *WiredTiger* format in which a single collection grows from under 8 GB to over 8 GB in size.
- The time before an unreachable `mongos` process is deactivated is now configurable on a per group basis. See *Admin Only Group Settings*.
- The time before a standby Monitoring Agent takes over after the primary Monitoring Agent stops responding is now configurable to a minimum of 90 seconds. See the `mms.monitoring.agent.session.timeoutMillis` setting in *Ops Manager Configuration Files*.
- For Backup HTTP pull restore, the link expiration and the number of allowed uses of a link are now configurable. See *Advanced Backup Restore Settings*.

Ops Manager Server 1.6.0

Released 2015-03-02

New Features

- Initial release of *Automation*. Automation manages many basic administrative tasks for MongoDB deployments, including version upgrades, adding replica set members, adding shards, and changing oplog size. You can both *import existing deployments into Automation* and *create new deployments on your provisioned hardware*.
- Windows support (Monitoring and Backup only). You can *Install Ops Manager on Windows* using MSI files. Ops Manager supports Windows Server 2008 R2 and above.
- Support for MongoDB 3.0, including support for backups that use the *WiredTiger* storage engine.

  To monitor or back up MongoDB 3.0 deployments, you must install Ops Manager 1.6 or higher. To monitor a MongoDB 3.0 deployment, you must also run Monitoring Agent version 2.7.0 or higher.
- Support for using the SSL and MONGODB-X509 authentication mechanisms for the backing MongoDB databases. See *Configure the Connections to the Backing MongoDB Instances*.
- Public API endpoints to manage Automation configuration. For more information, see *Automation* in the API documentation.

Improvements

- The Ops Manager's *Administration* interface provides more information to make it easier to monitor the health of the Ops Manager installation.
- The Ops Manager Deployment tab now displays all deployment information on one page, with icons for selecting view options. The new Topology View groups all hosts by the replica set or sharded cluster they are part of. The new Servers View shows information about MongoDB processes and Ops Manager agents grouped by server.
- Fixed an isssue (MMS-2273) where, in certain situations, the Backup Agent was not reporting a cluster snapshot as potentially inconsistent.
- Improved handling of cursor timeouts by the Backup Agent. To use this improvement, upgrade to the latest Backup Agent, which is included with Ops Manager. The improvement became available with Backup Agent version `2.3.3.209-1`.

Considerations for Upgrade

- Ops Manager 1.8.0, when released, **will not** support MongoDB 2.4 for the *Ops Manager Application Database* and *Backup Blockstore Database*. Ops Manager Server 1.8.0 *will* continue to support MongoDB 2.4 for your monitored and backed-up databases.
- Ops Manager 1.6.0 supports direct upgrades only from MMS On Prem 1.3 and above.
- The procedure to configure Ops Manager to run with HTTPS has changed and is greatly simplified. The previous procedure no longer works. For the new procedure, see *Configure SSL Connections to Ops Manager*.
- The connection string to the Backup Blockstore database is now configured through the Administration interface's *Blockstores page* and not through the `mongo.backupdb.mongoUri` field in the `conf-daemon.properties` configuration file.
- Ops Manager no longer requires you to supply the replica set name of the backing MongoDB instances. The `mongo.replicaSet` and `mongo.backupdb.replicaSet` properties have been removed from the configuration files. These properties had previously controlled whether Ops Manager treated a connection to a backing instance as a standalone or replica set, for the purpose of setting the write concern. Ops Manager now sets write concern based on how many hosts are supplied in the connection string.
- You can disable Automation for the entire Ops Manager installation through the `mms.featureFlag.automation` setting in the `conf-daemon.properties` *configuration file*.
- Removed the *Dashboards* view from the Ops Manager UI. You can view monitoring metrics from the *Deployment* tab. See: *Monitoring Metrics* for an overview of the available metrics and how to access them.

MMS Onprem Server 1.5.5

Released 2015-03-26

- **Security Update**: resolved issue where users removed from LDAP groups were not always removed from corresponding Ops Manager groups. This upgrade is **highly recommended** for anyone using LDAP authentication.

MMS Onprem Server 1.5.4

Released 2015-03-18

- Fixed race condition that could cause the Backup Daemon to hang when the MongoDB process for a *head* database fails to start.

- Fixed an issue where a rollback occurring shortly after a terminate could step on the terminate.
- The time before an unreachable `mongos` process is deactivated is now configurable on a per group basis. See *Admin Only Group Settings*.
- The time before a standby Monitoring Agent takes over after the primary Monitoring Agent stops responding is now configurable to a minimum of 90 seconds. See the `mms.monitoring.agent.session.timeoutMillis` setting in *Ops Manager Configuration Files*.
- For Backup HTTP pull restore, the link expiration and the number of allowed uses of a link are now configurable. See *Advanced Backup Restore Settings*.

MMS OnPrem Server 1.5.3

Released 2014-12-17

Significant improvements in performance for the processing of MMS OnPrem Monitoring data for MMS OnPrem Groups with a large number of hosts

MMS OnPrem Server 1.5.2

Released 2014-11-18

- Added Support for archive restores (`.tar.gz`) for databases whose filenames exceed 100 characters.
- API: Skip missed points in metrics data, instead of returning empty data.
- API: Return correct number of data points when querying metric data with the period option.
- Backup Agent update to `2.3.3.209-1`

MMS OnPrem Server 1.5.1

Released 2014-09-26

- Fix cases where replica set member alerts (e.g. no primary, number of healthy members) could send false positives.
- Skip `backup-daemon` *`rootDirectory`* and `mongo.backupdb.mongoUri` overlap check when the `mongo.backupdb.mongoUri` is on a different host.
- `mms-gen-key` script handles user's effective group being different than the username.
- Security enhancements.

MMS OnPrem Server 1.5.0

Released 2014-09-02

Considerations for Upgrade

- MMS OnPrem *only* supports direct upgrades from 1.3 and 1.4.
- Change in configurations and policy for 2FA: Two-factor authentication must now be explicitly enabled using the *`mms.multiFactorAuth.require`* setting.

- The default LDAP group separator became `;;`. Previously the separator was `,`. See the *LDAP configuration* documentation for more information.
- Suppressed hosts will only remain suppressed for 30 minutes.

  Previously, if after deleting a host, from MMS OnPrem Monitoring the hostname and port combination would be added to a suppression list with an infinite lifetime. The suppression list prevented a race condition where host in a cluster would be auto-discovered by another member of a deployment before the host could was fully removed. Now, hostname and port combinations remain on the suppression list for only 30 minutes.
- Set the *`mms.remoteIp.header`* in the `conf-mms.properties` file if clients access the MMS OnPrem Application via a load balancer.
- `mongo.backupdb.mongoUri` is no longer in `conf-mms.properties`. This was previously a required field in this file. It remains in the backup daemons's `conf-daemon.properties`.
- Stored MongoDB profile data is not transferred between OnPrem 1.4 and OnPrem 1.5 during the upgrade process.

Improvements

- When an MMS OnPrem Backup job fails to bind, the system will periodically and automatically retry.
- All MMS OnPrem Backup jobs will retry indefinitely.
- Point in Time restores are now available with one second granularity.

New Features

- MMS OnPrem *Public API*.
- Explicit support for multiple MMS OnPrem Backup Blockstore databases and the ability to pin MMS OnPrem Groups to specific backup daemons and databases. See *Configure Multiple Blockstores in Multiple Data Centers* for more information.
- MMS OnPrem can authenticate using LDAP to both the database backing MMS OnPrem and the monitored and backed up MongoDB deployments. See *Configure Users and Groups with LDAP for Ops Manager*.
- Enhanced auditing. See *Audit Events* for more information.
- Ability to acknowledge alerts with comments.
- New cluster page that shows individual, sum or average metrics for all shards in a cluster.

MMS OnPrem Server 1.4.3

Released 2014-07-22

- Addressed issues related to Backup Job assignment for 2.6.x clusters that used the `clusterMonitor` role to support MMS OnPrem Monitoring.
- Fixed problem importing email addresses for users for deployments that use LDAP integration.
- Fixed rare race condition caused high CPU usage in the MMS OnPrem HTTP Service if the application cannot connect to one of the backing databases.
- Additional security enhancements.

MMS OnPrem Server `1.4.2`

Released 2014-05-29

- Critical bug fix for backing up MongoDB 2.6 deployments that include user or custom role definitions:
  - The `system.version` collection in the admin database will be included in all future snapshots.
  - The `system.roles` collection in the admin database will be included after a new initial sync is performed.

  Users capturing backups of MongoDB 2.6 replica sets or clusters with MMS OnPrem that include custom role definitions should perform a new initial sync. Taking a new initial sync will ensure that the role definitions are included in the backup.
- Disable MongoDB `usePowerOf2Sizes` for insert-only MMS OnPrem Backup collections.
- Speed optimization for MMS OnPrem Backup HTTP pull restores.
- Fix for LDAP integration, MMS OnPrem now passes full `dn` correctly when authenticating the user.

MMS OnPrem Server `1.4.1`

Released 2014-04-28

- Ability to Backup replica sets or clusters using Kerberos authentication
- Ability to Backup replica sets or clusters running specific custom MongoDB builds provided by MongoDB, Inc.
- Fix validation issue preventing Backup of MongoDB 2.6.0 clusters
- Reduced log noise from Monitoring Agent when monitoring MongoDB 2.0 or unreachable mongods

MMS OnPrem Server `1.4.0`

Released 2014-04-08

- Includes MMS OnPrem Backup: continuous backup with point-in-time recovery of replica sets and cluster-wide snapshots of sharded clusters.
- Finer-grained roles and permissions.
- Improved user interface for alerts.
- Enhanced Activity Feed for auditing of all activity.
- Monitoring Agent distributed as OS-specific binary. Python dependency removed.
- LDAP integration for managing users and groups.

MMS OnPrem 1.4.0 requires MongoDB 2.4.9+ instances for *backing storage*.

MMS OnPrem Server `1.3.0`

Released 2013-12-01

- Packaging/support for Debian and SUSE Linux.
- Kerberos authentication support between MMS OnPrem server and backing MongoDBs, as well as between Monitoring Agent and the MongoDBs it monitors.
- OnPrem users can be overall site administrators. (MMS OnPrem Admins)

- New admin section where MMS OnPrem Admins can manage user roles and message banners.
- Tunable advanced password and session management configurations.
- Encryption key rotation, more specific CORS policy, auth tokens removed from chart URLs, and other security enhancements.

MMS OnPrem Server `1.2.0`

Released 2013-07-24

- Redesigned user interface and enhanced algorithm to auto-discover hosts and derive host topology.
- SNMP monitoring.
- Ability to export charts.
- Option to store encrypted authentication credentials in the `mmsDb` property in the configuration file.
- Ability to classify users within an MMS OnPrem Group as group administrators or read-only users.

13.2 Automation Agent Changelog

Automation Agent `1.4.18.1199-1`

Released with Ops Manager 1.6.3 on 2015-06-23

- Added support for importing an existing deployment that contains authenticated arbiters on which the hostname does not resolve locally to the loopback interface.
- Fixed logic used for performing a rolling restart.
- Fixed issue with deriving the default port for config servers started with the `--configsvr` option but with no port specified. See: MMS-2489.

Automation Agent `1.4.16.1075`

Released 2015-04-28

- Fixed an issue with updating users created on MongoDB 2.4.
- Fixed an issue with config server repair that occurred if the third config server was out of sync.

Automation Agent `1.4.15.999`

Released 2015-03-26

- Fixed a rare edge-case that prevented the Automation Agent from successfully enabling authentication.

Automation Agent `1.4.14.983`

Released 2015-03-02

Initial release.

13.3 Monitoring Agent Changelog

Monitoring Agent `2.9.2.184`

Released 2015-04-28

- Added an explicit timeout for SSL connections to MongoDB instances.
- Upgraded the MongoDB Go driver (mgo) version, which fixed a rare deadlock issue that could occur when monitoring `mongos` instances.

Monitoring Agent `2.9.1.176`

- Adds support for non-default Kerberos service names.
- Added support for authentication using MongoDB 2.4 style client certificates.
- The Monitoring Agent now identifies itself to the MMS servers using the fully qualified domain name (FQDN) of the server on which it is running.
- Ops Manager now staggers the timing of DNS look-ups, to avoid triggering a rare issue in glibc 2.19 on Ubuntu 14.04.
- Adds support for RHEL7.
- Improved error handling on Windows.
- Improved connection management for monitored MongoDB processes.
- Improve correctness of the database statics collection.
- Now uses the `listDatabases` command to retrieve a list of databases.
- The default value for *`sslTrustedServerCertificates`* is now `true`. Users upgrading from 2.4.0 and using SSL will need to set the value of *`sslTrustedServerCertificates`* in their configuration file. See *`sslTrustedServerCertificates`* for more information.

Monitoring Agent `2.4.2.113`

Released with OnPrem 1.5.0

- Upgraded agent to use Go 1.3.
- Updated mgo driver, which includes fix for MGO-34. All DNS lookups should now timeout appropriately.
- Added support for connecting to hosts using LDAP authentication.
- Added support for `version` and `-version` command line options.
- Agent now displays git commit hash of Monitoring Agent in the log file.
- Updates to the configuration file format.

Monitoring Agent `2.3.1.89-1`

Released with OnPrem 1.4.3

- Improved logging for MongoDB 2.6 config servers when connecting with a user that has the built-in clusterMonitor role.
- Fixes issues with connecting to replica set members that use auth with an updated Go client library.

- Added support for HTTP proxy configuration in the agent configuration file.
- Agent includes support for an Offline data collection mode.

Monitoring Agent `2.1.4.51-1`

Released with \mms\ OnPrem 1.4.2

Prevent high CPU use when monitoring unreachable `mongod`.

Monitoring Agent `2.1.3.48-1`

Released with OnPrem 1.4.1

Reduction in unnecessary log messages for unsupported operations on monitored MongoDB 2.2 instances.

Monitoring Agent `2.1.1.41-1`

Released with OnPrem 1.4.0

Ability to monitor hosts using Kerberos authentication.

Monitoring Agent `1.6.6`

Released with OnPrem1.3

- Added kerberos support for agents running on Python 2.4.x.
- Added logging when the `dbstats` command fails.

13.4 Backup Agent Changelog

Backup Agent `3.1.2.274`

Released 2015-04-28

- Added an explicit timeout for SSL connections to MongoDB instances.
- Added an optimization for syncs of collections with lots of small documents.

Backup Agent `3.1.1.263`

Released 2015-03-02

- Adds support for non-default Kerberos service names.
- Adds support for authentication using MongoDB 2.4-style client certificates.
- The Backup Agent now identifies itself to the Ops Manager servers using the fully qualified domain name (FQDN) of the server on which it is running.
- The Backup Agent now captures a checkpoint even if it is unable to stop the balancer. These checkpoints are not guaranteed to be consistent, because of in-progress chunk migrations. The user interface identifies these checkpoints.

Backup Agent `2.3.3.209-1`

Released with OnPrem 1.5.2

Use no-timeout cursors to work around MGO-53.

Backup Agent `2.3.1.160`

Released with \mms\ OnPrem 1.5.0

- Backup Agent now sends oplog slices in batches.
- Improved stability around oplog tokens for environments with unstable networks.
- Support for a new API that allows Ops Manager to ingest oplog entries before the entire payload has reached the Ops Manager servers.
- Upgraded agent to use to Go 1.3.
- Added support for `version` and `-version` command line options.
- Added support for connecting to hosts using LDAP authentication.
- Agent now provides additional logging information when the Backup Agent manipulates the balancer.
- Agent now supports configuring HTTP proxies with the config file.

Backup Agent `1.5.1.83-1`

Released with \mms\ OnPrem 1.4.2

Critical update for users running the MongoDB 2.6 series that use authorization.

The Backup Agent now includes `system.version` and `system.role` collections from the admin database in the initial sync.

Backup Agent `1.5.0.57-1`

Released with OnPrem 1.4.1

Support for backing up Kerberos-authenticated replica sets and clusters

Backup Agent `1.4.6.42-1`

Released with OnPrem 1.4.0

- Major stability update.
- Prevent a file descriptor leak.
- Correct handling of timeouts for connections hung in the SSL handshaking phase.