

---

# MongoDB Ops Manager Manual

*Release 2.0*

**MongoDB, Inc.**

September 02, 2016

© MongoDB, Inc. 2008 - 2016

# Contents

<b>1</b>	<b>Ops Manager Introduction</b>	<b>13</b>
1.1	Ops Manager Overview . . . . .	13
Overview . . . . .	14	
Monitoring . . . . .	14	
Automation . . . . .	14	
Backup . . . . .	15	
1.2	Ops Manager Components . . . . .	15
Network Diagram . . . . .	16	
Ops Manager . . . . .	16	
Dedicated MongoDB Databases for Operational Data . . . . .	17	
Default Ports . . . . .	18	
1.3	Example Deployment Topologies . . . . .	18
Considerations . . . . .	18	
Test Install on a Single Server . . . . .	18	
Production Installs . . . . .	19	
1.4	Install a Simple Test Ops Manager Installation . . . . .	22
Overview . . . . .	22	
Procedure . . . . .	23	
<b>2</b>	<b>Install Ops Manager</b>	<b>26</b>
2.1	Installation Checklist . . . . .	26
Overview . . . . .	27	
Topology Decisions . . . . .	27	
Security Decisions . . . . .	29	
Backup Decisions . . . . .	29	
2.2	Ops Manager Hardware and Software Requirements . . . . .	30
Hardware Requirements . . . . .	30	
Network Requirements . . . . .	32	
Software Requirements . . . . .	35	
2.3	Install the Ops Manager Application Database and Backup Database . . . . .	37
Overview . . . . .	37	
Replica Set Topology . . . . .	37	
Replica Sets Requirements . . . . .	38	
Server Prerequisites . . . . .	38	
Choosing a Storage Engine for the Backing Databases . . . . .	38	
Procedure . . . . .	39	
2.4	Install Ops Manager . . . . .	39
Install Ops Manager with an rpm Package . . . . .	39	
Install Ops Manager with a deb Package . . . . .	43	
Install Ops Manager from a tar.gz or zip Archive . . . . .	47	
Install Ops Manager on Microsoft Windows . . . . .	51	
2.5	Advanced Configuration Options . . . . .	55
Configure Local Mode if Servers Have No Internet Access . . . . .	55	
Configure a Highly Available Ops Manager Application . . . . .	60	
Configure a Highly Available Ops Manager Backup Service . . . . .	62	
Assign Snapshot Stores to Specific Data Centers . . . . .	63	
Configure Ops Manager to Pass Outgoing Traffic through an HTTP or HTTPS Proxy . . . . .	66	
2.6	Upgrade Ops Manager . . . . .	67
Upgrade Ops Manager with an rpm Package . . . . .	67	
Upgrade Ops Manager with a deb Package . . . . .	71	
Upgrade Ops Manager from a tar.gz or zip Archive . . . . .	75	

Upgrade Ops Manager on Microsoft Windows Server . . . . .	78
<b>3 Create or Import a MongoDB Deployment</b>	<b>82</b>
3.1 Getting Started with Deployments . . . . .	82
Get Started with Ops Manager . . . . .	82
3.2 Provision Servers . . . . .	83
Provision Servers for Automation . . . . .	83
Provision Servers for Monitoring . . . . .	84
3.3 Add Existing MongoDB Processes to Ops Manager . . . . .	85
Overview . . . . .	85
Considerations . . . . .	86
Add MongoDB Processes . . . . .	87
3.4 Add Monitored Processes to Automation . . . . .	87
Overview . . . . .	87
Considerations . . . . .	87
Prerequisites . . . . .	88
Procedure . . . . .	88
3.5 Deploy a Replica Set . . . . .	89
Overview . . . . .	89
Unique Names for Deployment Items . . . . .	89
Prerequisites . . . . .	89
Procedure . . . . .	90
3.6 Deploy a Sharded Cluster . . . . .	91
Overview . . . . .	91
Unique Names for Deployment Items . . . . .	91
Prerequisites . . . . .	91
Procedure . . . . .	92
3.7 Deploy a Standalone MongoDB Instance . . . . .	93
Overview . . . . .	93
Prerequisites . . . . .	93
Procedure . . . . .	93
3.8 Connect to a MongoDB Process . . . . .	94
Overview . . . . .	94
Firewall Rules . . . . .	94
Procedures . . . . .	94
<b>4 Manage Deployments</b>	<b>95</b>
4.1 Edit a Deployment's Configuration . . . . .	96
Overview . . . . .	96
Considerations . . . . .	96
Procedure . . . . .	97
4.2 Create Indexes . . . . .	98
Overview . . . . .	99
Procedures . . . . .	99
4.3 Calculate Suggested Indexes . . . . .	101
Overview . . . . .	101
Prerequisites . . . . .	101
Procedure . . . . .	101
4.4 Edit a Replica Set . . . . .	102
Overview . . . . .	102
Procedures . . . . .	102
Additional Information . . . . .	105
4.5 Migrate a Replica Set Member to a New Server . . . . .	105
Overview . . . . .	105

Considerations . . . . .	106
Procedure . . . . .	106
4.6 Convert Config Servers to a Replica Set . . . . .	107
Overview . . . . .	107
Prerequisites . . . . .	108
Procedure . . . . .	108
4.7 MongoDB Processes . . . . .	108
Shut Down a MongoDB Process . . . . .	109
Restart a MongoDB Process . . . . .	110
Suspend or Resume Automation for a Process . . . . .	110
Remove a Process from Management or Monitoring . . . . .	112
Start MongoDB Processes with Init Scripts . . . . .	113
4.8 Move or Add a Monitoring or Backup Agent . . . . .	115
Overview . . . . .	115
Procedures . . . . .	115
4.9 MongoDB Versions . . . . .	116
Change the Version of MongoDB . . . . .	116
Configure Available MongoDB Versions . . . . .	118
<b>5 Monitoring and Alerts</b>	<b>118</b>
5.1 View Diagnostics . . . . .	119
Overview . . . . .	119
Procedure . . . . .	119
5.2 Profile Databases . . . . .	120
Overview . . . . .	121
Considerations . . . . .	121
Procedures . . . . .	122
5.3 View Logs . . . . .	123
Overview . . . . .	123
MongoDB Real-Time Logs . . . . .	123
MongoDB On-Disk Logs . . . . .	124
Agent Logs . . . . .	125
5.4 Manage Alerts . . . . .	126
Overview . . . . .	126
Procedures . . . . .	126
5.5 Manage Alert Configurations . . . . .	129
Overview . . . . .	129
Considerations . . . . .	130
Procedures . . . . .	131
5.6 Alert Conditions . . . . .	135
Overview . . . . .	136
Host Alerts . . . . .	136
Replica Set Alerts . . . . .	141
Agent Alerts . . . . .	142
Backup Alerts . . . . .	142
User Alerts . . . . .	144
Group Alerts . . . . .	144
5.7 Global Alerts . . . . .	144
Overview . . . . .	144
Procedures . . . . .	145
5.8 System Alerts . . . . .	148
Overview . . . . .	148
System Alerts . . . . .	148
Procedures . . . . .	149

<b>6 Back Up and Restore Deployments</b>	<b>151</b>
6.1 Back up MongoDB Deployments . . . . .	151
Backup Flows . . . . .	151
Backup Preparations . . . . .	156
Back up a Deployment . . . . .	158
6.2 Manage Backups . . . . .	160
Edit a Backup's Settings . . . . .	161
Stop, Restart, or Terminate a Backup . . . . .	163
View a Backup's Snapshots . . . . .	164
Delete a Snapshot . . . . .	165
Resync a Backup . . . . .	165
Generate a Key Pair for SCP Restores . . . . .	166
Disable the Backup Service . . . . .	167
6.3 Restore MongoDB Deployments . . . . .	168
Restore Overview . . . . .	168
Restore Flows . . . . .	169
Restore a Sharded Cluster from a Backup . . . . .	172
Restore a Replica Set from a Backup . . . . .	182
Restore a Single Database . . . . .	188
Seed a New Secondary from Backup Restore . . . . .	192
<b>7 Security</b>	<b>194</b>
7.1 Security Overview . . . . .	194
Overview . . . . .	194
Security Options Available in the Current Version of Ops Manager . . . . .	194
Supported User Authentication by Ops Manager Version . . . . .	195
Supported MongoDB Security Features on Linux . . . . .	195
Supported MongoDB Security Features on Windows . . . . .	196
7.2 Firewall Configuration . . . . .	197
Overview . . . . .	197
Accessible Ports . . . . .	197
Ports Required to Use Ops Manager . . . . .	197
Ports Needed to Administer Ops Manager . . . . .	201
Ports Needed to Integrate Ops Manager with SNMP . . . . .	201
Ports Needed to Authenticate with Ops Manager . . . . .	201
Ports Needed to Authenticate with MongoDB . . . . .	201
7.3 Manage Ops Manager Ports . . . . .	202
Overview . . . . .	202
Procedures . . . . .	202
7.4 Configure SSL Connections to Ops Manager . . . . .	204
Overview . . . . .	204
Run the Ops Manager Application Over HTTPS . . . . .	204
7.5 Configure the Connections to the Backing MongoDB Instances . . . . .	205
Overview . . . . .	205
Prerequisites . . . . .	205
Procedures . . . . .	206
7.6 Enable SSL for a Deployment . . . . .	208
Overview . . . . .	208
Procedures . . . . .	208
7.7 Configure users and groups using LDAP with Ops Manager . . . . .	210
Overview . . . . .	210
Prerequisites . . . . .	211
Procedure . . . . .	211
7.8 Enable Authentication for an Ops Manager Group . . . . .	214

Configure MongoDB Authentication and Authorization . . . . .	215
Enable SCRAM-SHA-1 / MONGODB-CR Authentication for your Ops Manager Group . . . . .	218
Enable LDAP Authentication for your Ops Manager Group . . . . .	220
Enable Kerberos Authentication for your Ops Manager Group . . . . .	222
Enable x.509 Authentication for your Ops Manager Group . . . . .	224
Clear Security Settings . . . . .	228
7.9 Manage Two-Factor Authentication for Ops Manager . . . . .	229
Overview . . . . .	229
Procedures . . . . .	229
<b>8 Groups and Users</b>	<b>231</b>
8.1 Manage Groups . . . . .	231
Create a Group . . . . .	231
Edit a Group's Configuration . . . . .	232
Remove a Group . . . . .	235
8.2 Manage Ops Manager Users and Roles . . . . .	235
Manage Ops Manager Users . . . . .	236
Ops Manager Roles . . . . .	237
8.3 Manage MongoDB Users and Roles . . . . .	240
Enable MongoDB Role-Based Access Control . . . . .	240
Manage MongoDB Users and Roles . . . . .	242
Manage Custom Roles . . . . .	244
<b>9 Account Management</b>	<b>247</b>
9.1 Access Your User Account . . . . .	247
Overview . . . . .	247
Account . . . . .	247
Personalization . . . . .	247
Public API Access . . . . .	248
My Groups . . . . .	248
9.2 Edit Your User Account . . . . .	248
Overview . . . . .	248
Change Password . . . . .	248
Change Email Address . . . . .	248
Change Mobile Phone Number . . . . .	248
Change Display Settings and Newsletter Preference . . . . .	249
Additional Information . . . . .	249
9.3 Manage Your Two-Factor Authentication Options . . . . .	249
Overview . . . . .	249
Configure Two-Factor Authentication with Text or Voice . . . . .	250
Configure Two-Factor Authentication with Google Authenticator . . . . .	251
Generate New Recovery Codes . . . . .	252
<b>10 Administer Ops Manager</b>	<b>252</b>
10.1 Administration Overview . . . . .	252
Introduction . . . . .	253
General Tab . . . . .	253
Backup Tab . . . . .	254
Alerts Tab . . . . .	258
Control Panel Tab . . . . .	259
10.2 Administer Backups . . . . .	259
Manage Blockstore Snapshot Storage . . . . .	259
Manage File System Snapshot Storage . . . . .	262
Configure the Size of the Blocks in the Blockstore . . . . .	264

Move Jobs from a Lost Backup Daemon to another Backup Daemon . . . . .	265
10.3 Add a Message to the Interface . . . . .	266
Overview . . . . .	266
Procedures . . . . .	266
10.4 Start and Stop Ops Manager Application . . . . .	267
Start the Ops Manager Server . . . . .	267
Stop the Ops Manager Application . . . . .	267
Startup Log File Output . . . . .	267
Optional: Run as Different User . . . . .	268
Optional: Ops Manager Application Server Port Number . . . . .	268
10.5 Back Up Ops Manager . . . . .	268
Back Up with the Public API . . . . .	269
Shut Down and Back Up . . . . .	269
<b>11 API</b>	<b>269</b>
11.1 Public API Principles . . . . .	270
Overview . . . . .	270
HTTP Methods . . . . .	270
JSON . . . . .	270
Linking . . . . .	271
Lists . . . . .	272
Envelopes . . . . .	273
Pretty Printing . . . . .	273
Response Codes . . . . .	273
Errors . . . . .	274
Authentication . . . . .	274
Automation . . . . .	274
Additional Information . . . . .	275
11.2 Public API Resources . . . . .	275
Root . . . . .	275
Hosts . . . . .	276
Agents . . . . .	284
Metrics . . . . .	287
Clusters . . . . .	292
Groups . . . . .	296
Users . . . . .	301
Alerts . . . . .	306
Alert Configurations . . . . .	312
Maintenance Windows . . . . .	324
Backup Configurations . . . . .	328
Snapshot Schedule . . . . .	333
Snapshots . . . . .	335
Checkpoints . . . . .	341
Restore Jobs . . . . .	345
Whitelist . . . . .	354
Automation Configuration Resource . . . . .	356
Automation Status . . . . .	374
11.3 Enable the Public API . . . . .	377
Overview . . . . .	377
Procedure . . . . .	377
11.4 Configure Public API Access . . . . .	377
Overview . . . . .	378
Procedures . . . . .	378
11.5 Public API Tutorials . . . . .	379

Deploy a Cluster through the API . . . . .	379
Update the Automation Configuration . . . . .	387
Update the MongoDB Version of a Deployment . . . . .	388
Automate Backup Restoration through the API . . . . .	390
<b>12 Troubleshooting . . . . .</b>	<b>392</b>
12.1 Getting Started Checklist . . . . .	392
Authentication Errors . . . . .	392
Check Agent Output or Log . . . . .	393
Confirm Only One Agent is Actively Monitoring . . . . .	393
Ensure Connectivity Between Agent and Monitored Hosts . . . . .	393
Ensure Connectivity Between Agent and Ops Manager Server . . . . .	393
Allow Agent to Discover Hosts and Collect Initial Data . . . . .	393
12.2 Installation . . . . .	393
The monitoring server does not start up successfully . . . . .	393
12.3 Monitoring . . . . .	394
Alerts . . . . .	394
Deployments . . . . .	395
Monitoring Agent Fails to Collect Data . . . . .	395
Hosts . . . . .	395
Groups . . . . .	396
Munin . . . . .	396
12.4 Authentication . . . . .	397
Two-Factor Authentication . . . . .	397
LDAP . . . . .	398
12.5 Backup . . . . .	398
Logs Display MongodVersionException . . . . .	398
Insufficient Oplog Size Error . . . . .	399
12.6 System . . . . .	399
Logs Display OutOfMemoryError . . . . .	399
Obsolete Config Settings . . . . .	400
12.7 Automation Checklist . . . . .	400
Automation Runs Only on 64-bit Architectures . . . . .	400
Using Own Hardware . . . . .	400
Networking . . . . .	401
Automation Configuration . . . . .	401
Sizing . . . . .	401
<b>13 Frequently Asked Questions . . . . .</b>	<b>401</b>
13.1 What versions of MongoDB does Ops Manager manage? . . . . .	401
13.2 Automation FAQs . . . . .	401
How does Ops Manager manage MongoDB deployments? . . . . .	402
How many Automation Agents do I need? . . . . .	402
Is any MongoDB data transferred by the Automation Agent? . . . . .	402
Will Ops Manager handle failures during an upgrade, such as Ops Manager going down or a network partition? . . . . .	402
What types of deployment can I create in Ops Manager? . . . . .	402
13.3 Monitoring FAQs . . . . .	402
Host Configuration . . . . .	402
Monitoring Agent . . . . .	403
Data Presentation . . . . .	404
Data Retention . . . . .	405
13.4 Backup FAQs . . . . .	405
Requirements . . . . .	405

Interface . . . . .	406
Operations . . . . .	406
Configuration . . . . .	408
Restoration . . . . .	409
13.5 Administration FAQs . . . . .	411
Can I reset my password? . . . . .	411
How do I add a user to my company/group? . . . . .	411
How can I configure multiple Google Authenticator apps to use the same account? . . . . .	411
What do the alert conditions mean? . . . . .	411
What alerts are configured by default? . . . . .	411
13.6 Miscellaneous . . . . .	411
What open source projects does Ops Manager use? . . . . .	411
<b>14 Reference</b>	<b>411</b>
14.1 Ops Manager Configuration . . . . .	412
Overview . . . . .	412
Web Server Settings . . . . .	413
Email Settings . . . . .	414
User Authentication Method . . . . .	415
Authentication through Ops Manager Application Database . . . . .	416
Authentication through LDAP . . . . .	416
Multi-Factor Authentication (MFA) Settings . . . . .	420
Other Authentication Options . . . . .	421
HTTP/HTTPS Proxy Settings . . . . .	421
Twilio Integration Settings . . . . .	422
MongoDB Version Management . . . . .	422
Backup Snapshots . . . . .	423
Public API . . . . .	423
Monitoring Agent Session Failover . . . . .	424
SNMP Heartbeat Settings . . . . .	424
Backup Settings . . . . .	425
Backup Daemon . . . . .	426
Ops Manager Application Database Connection String . . . . .	426
SSL Connection to the Application Database . . . . .	428
Kerberos Authentication to the Application Database . . . . .	428
Encrypt User Credentials . . . . .	429
14.2 Automation Agent . . . . .	430
Install the Automation Agent . . . . .	430
Automation Agent Configuration . . . . .	444
14.3 Monitoring Agent . . . . .	446
Install Monitoring Agent . . . . .	447
Monitoring Agent Configuration . . . . .	460
Required Access for Monitoring Agent . . . . .	464
Configure Monitoring Agent for Access Control . . . . .	465
Configure Monitoring Agent for SSL . . . . .	471
Configure Hardware Monitoring with <code>munin-node</code> . . . . .	473
Start or Stop the Monitoring Agent . . . . .	475
Remove Monitoring Agents from Ops Manager . . . . .	476
14.4 Backup Agent . . . . .	477
Install Backup Agent . . . . .	477
Backup Agent Configuration . . . . .	491
Required Access for Backup Agent . . . . .	494
Configure Backup Agent for Access Control . . . . .	495
Configure Backup Agent for SSL . . . . .	502

Start or Stop the Backup Agent . . . . .	504
Remove the Backup Agent from Ops Manager . . . . .	505
14.5 Database Commands Used by Monitoring Agent . . . . .	506
14.6 Audit Events . . . . .	507
User Audits . . . . .	507
Host Audits . . . . .	508
Alert Config Audits . . . . .	509
Backup Audits . . . . .	509
Group Audits . . . . .	510
14.7 MongoDB Compatibility . . . . .	510
Automation and MongoDB . . . . .	510
Monitoring and MongoDB . . . . .	510
Backup and MongoDB . . . . .	511
MongoDB Deployment Types . . . . .	511
14.8 Supported Browsers . . . . .	511
14.9 Advanced Options for MongoDB Deployments . . . . .	511
Overview . . . . .	511
Advanced Options . . . . .	511
14.10 Automation Configuration . . . . .	515
Overview . . . . .	515
Configuration Version . . . . .	515
Download Base . . . . .	516
MongoDB Versions Specifications . . . . .	516
Automation Agent . . . . .	517
Monitoring Agent . . . . .	517
Backup Agent . . . . .	519
MongoDB Instances . . . . .	520
Replica Sets . . . . .	523
Sharded Clusters . . . . .	523
Cluster Balancer . . . . .	524
Authentication . . . . .	525
SSL . . . . .	527
MongoDB Roles . . . . .	527
Kerberos . . . . .	527
Indexes . . . . .	528
14.11 Supported MongoDB Options for Automation . . . . .	528
Overview . . . . .	528
MongoDB 2.6 and Later Configuration Options . . . . .	528
MongoDB 2.4 and Earlier Configuration Options . . . . .	531
14.12 SNMP Traps and Ops Manager Severities . . . . .	532
14.13 Glossary . . . . .	533
<b>15 Release Notes</b>	<b>536</b>
15.1 Ops Manager Server Changelog . . . . .	536
Ops Manager Server 2.0.6 . . . . .	536
Ops Manager Server 2.0.5 . . . . .	536
Ops Manager Server 2.0.4 . . . . .	537
Ops Manager Server 2.0.3 . . . . .	537
Ops Manager Server 2.0.2 . . . . .	538
Ops Manager Server 2.0.1 . . . . .	538
Ops Manager Server 2.0.0 . . . . .	538
Ops Manager Server 1.8.3 . . . . .	541
Ops Manager Server 1.8.2 . . . . .	541
Ops Manager Server 1.8.1 . . . . .	541

Ops Manager Server 1.8.0	541
Ops Manager Server 1.6.4	543
Ops Manager Server 1.6.3	543
Ops Manager Server 1.6.2	544
Ops Manager Server 1.6.1	544
Ops Manager Server 1.6.0	544
MMS Onprem Server 1.5.5	546
MMS Onprem Server 1.5.4	546
MMS OnPrem Server 1.5.3	546
MMS OnPrem Server 1.5.2	546
MMS OnPrem Server 1.5.1	546
MMS OnPrem Server 1.5.0	547
MMS OnPrem Server 1.4.3	548
MMS OnPrem Server 1.4.2	548
MMS OnPrem Server 1.4.1	548
MMS OnPrem Server 1.4.0	548
MMS OnPrem Server 1.3.0	549
MMS OnPrem Server 1.2.0	549
15.2 Automation Agent Changelog	549
Automation Agent 2.5.20.1755	549
Automation Agent 2.5.19.1732	549
Automation Agent 2.5.18.1647	550
Automation Agent 2.5.17.1604	550
Automation Agent 2.5.16.1552	550
Automation Agent 2.5.15.1526	550
Automation Agent 2.5.11.1484	550
Automation Agent 2.0.14.1398	551
Automation Agent 2.0.12.1238	551
Automation Agent 2.0.9.1201	551
Automation Agent 1.4.18.1199-1	552
Automation Agent 1.4.16.1075	552
Automation Agent 1.4.15.999	552
Automation Agent 1.4.14.983	552
15.3 Monitoring Agent Changelog	552
Monitoring Agent 3.9.1.326	552
Monitoring Agent 3.9.1.238	553
Monitoring Agent 3.7.1.227	553
Monitoring Agent 3.7.0.212	553
Monitoring Agent 3.3.1.193	553
Monitoring Agent 2.9.2.184	553
Monitoring Agent 2.9.1.176	553
Monitoring Agent 2.4.2.113	554
Monitoring Agent 2.3.1.89-1	554
Monitoring Agent 2.1.4.51-1	554
Monitoring Agent 2.1.3.48-1	554
Monitoring Agent 2.1.1.41-1	554
Monitoring Agent 1.6.6	555
15.4 Backup Agent Changelog	555
Backup Agent 3.9.0.336	555
Backup Agent 3.9.0.336	555
Backup Agent 3.4.2.314	555
Backup Agent 3.4.1.283	555
Backup Agent 3.1.2.274	555
Backup Agent 3.1.1.263	556

---

Backup Agent 2.3.3.209-1 . . . . .	556
Backup Agent 2.3.1.160 . . . . .	556
Backup Agent 1.5.1.83-1 . . . . .	556
Backup Agent 1.5.0.57-1 . . . . .	556
Backup Agent 1.4.6.42-1 . . . . .	557

---

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 or Import Deployments** Provision servers, and create or import MongoDB deployments.

**Manage Deployments** Manage and update your MongoDB deployments.

**Monitoring and Alerts** Monitor your MongoDB deployments and manage alerts.

**Backup and Restore** Initiate and restore backups.

**Security** Describes Ops Manager security features.

**Groups and Users** Manage Ops Manager users, and manage MongoDB users.

**Account Management** Manage your Ops Manager user account.

**Administer Ops Manager** 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

**Ops Manager Overview** Describes Ops Manager services and operations.

**Ops Manager Components** Describes Ops Manager components.

**Example Deployment Topologies** Describes common Ops Manager topologies.

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

### 1.1 Ops Manager Overview

## On this page

- [Overview](#)
- [Monitoring](#)
- [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.

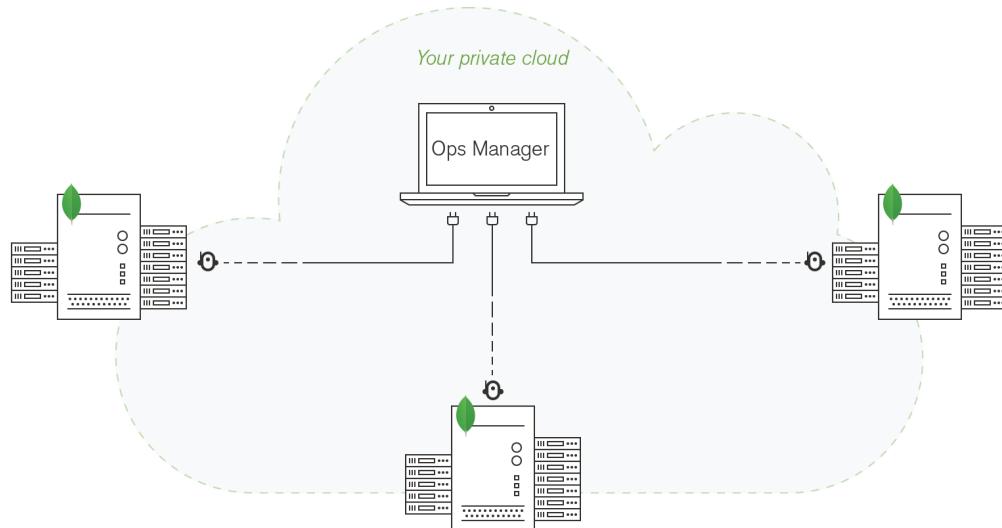
## Monitoring

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 snapshot storage and transfers files either through an HTTPS download link or by sending them via HTTPS or SCP.

When you restore from a checkpoint or point in time, Ops Manager first creates a local restore of a snapshot from the snapshot storage and then applies stored oplogs until the specified point is reached. Ops Manager delivers the backup via the same HTTPS or SCP mechanisms. To enable checkpoints, see *Enable Cluster Checkpoints*.

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

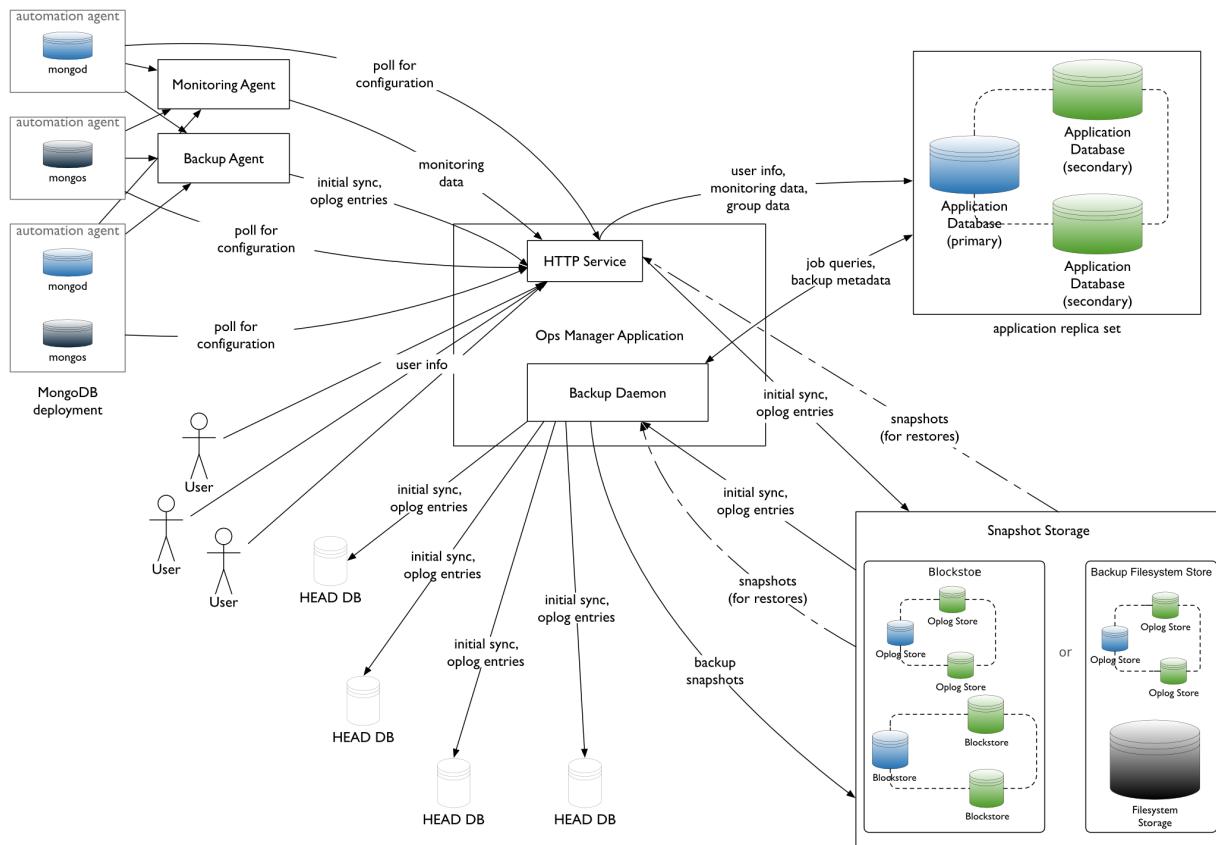
## 1.2 Ops Manager Components

## On this page

- Network Diagram
- Ops Manager
- Dedicated MongoDB Databases for Operational Data
- Default Ports

An Ops Manager installation includes servers running Ops Manager and servers hosting dedicated MongoDB databases for storing application data and snapshots.

## Network Diagram



## Ops Manager

The Ops Manager installation package consists of the *Ops Manager Application* and *Backup Daemon Service*. The Ops Manager application requires an Ops Manager Application Database (a dedicated MongoDB database to hold operational data) as well as dedicated MongoDB Backup Database(s).

### Ops Manager Application

The Ops Manager Application contains the user interface and the HTTP services the Agents use to transmit data to and from Ops Manager. These are all stateless and start automatically when the Ops Manager Application starts. Multiple

instances of the Ops Manager Application can run as long as each instance has the same configuration. Users and agents can interact with any instance.

The HTTP Service runs on port 8080 by default and 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 Daemon Service

Any Ops Manager instance can be configured to run the Backup Daemon service to provide backup capabilities. The Backup Daemon service manages both the local copies of the backed-up databases and each backup's snapshots. The daemon does scheduled work based on data coming into the 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 takes scheduled snapshots and stores the snapshots in either databases (the *Backup Data Storage*) or on the file system (the file system store). It also acts on restore requests by retrieving data from the Backup Database and delivering it to the requested destination.

*Multiple Backup Daemons* scale horizontally to increase your storage and can provide **manual** failover.

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

## 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 operational backup data. The backing databases run as *replica sets* to ensure redundancy and high availability. The replica sets host **only** Ops Manager data. You must *provision the replica sets* before installing Ops Manager.

## Ops Manager Application Database

This database contains *Ops Manager Application* metadata:

- 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 Requirements*.

## Backup Data Storage

Ops Manager backs up snapshots of deployments. These snapshots are stored in either databases or file systems. There can be more than one database and/or file system paths to which snapshots are written. The recent history of the deployment database is written to a separate database regardless of where the snapshots are written.

The components of backup data storage include:

- Snapshots. Snapshots can be written to one of two destinations:
  - A *Blockstore*, a MongoDB database.

- A *File System Store* a file system, in the directory of your choosing.
- *Oplog Slices*. The *Oplog Store* retains oplog entries the Backup Daemon applies to its local copies of your backed-up deployments.

The **Blockstore** backup option requires:

- Two databases that run on a dedicated MongoDB deployment that holds no other data:
  - A large database to store the backup data (the Blockstore).
  - A small database to store the operations logs (the Oplog Store).
- Disk space proportional to the deployments being backed up. See [Backup Database Hardware Requirements](#).

If redundancy and high availability for the backup data is desired, configure the Blockstores and Oplog Stores as a replica sets with three data-bearing members.

---

**Important:** Ops Manager cannot be backed up using Ops Manager Backup. See [Back Up Ops Manager](#).

---

The **File System Store** backup option requires:

- A small backup database (the Oplog Store) in a MongoDB deployment.
- A filesystem path that can be mounted from all Ops Manager application instances so they can read and write backup snapshots.

## Default Ports

For a list of Ops Manager’s default ports and health-check endpoints, see [Firewall Configuration](#).

## 1.3 Example Deployment Topologies

### On this page

- Considerations
- Test Install on a Single Server
- Production Installs

The following examples illustrate some possible MongoDB and Ops Manager deployments.

### Considerations

For best performance on any of these installs, configure each Backup server with two disk partitions: one for the Backup Database or File System Store and one for the head databases.

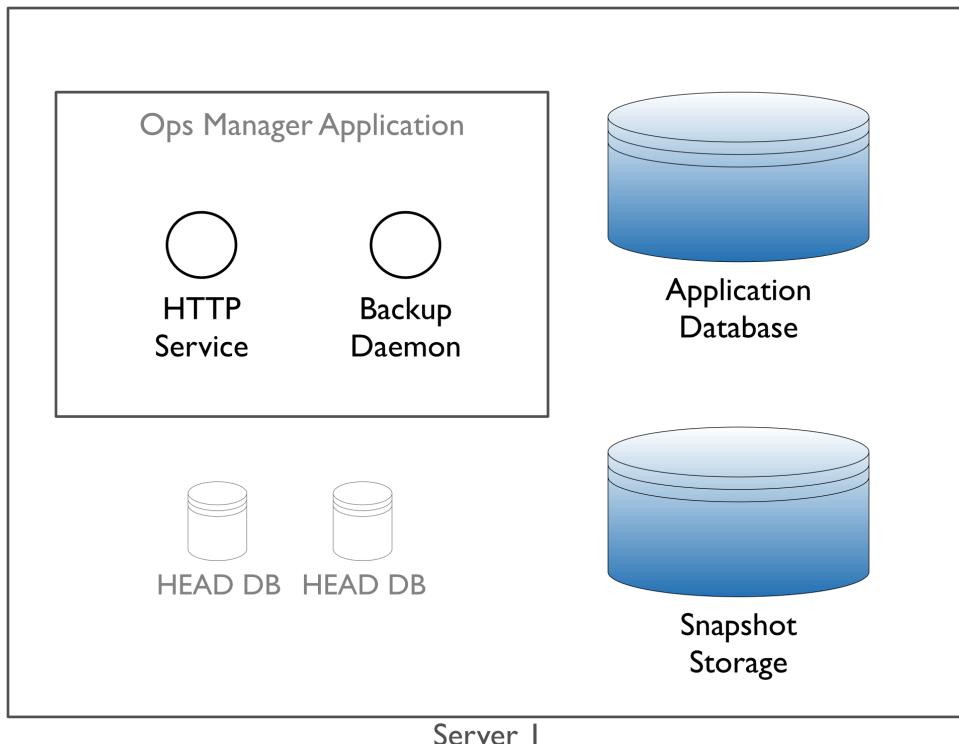
### Test Install on a Single Server

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](#).

---

**Note:** If you would like to test backup services, you can configure them using the Ops Manager Application. During configuration, you can specify the *head directory*. The Backup Daemon service creates the *head databases* dynamically in that directory. The Backup Daemon service then manages these head databases.

---



## Production Installs

### Redundant Metadata and Snapshots

This deployment provides redundancy for the *Ops Manager Application Database* and *Backup Data Storage*, in the event of server failure. The deployment keeps a redundant copy of each database by running each as a MongoDB *replica set* with two data-bearing members and an arbiter.

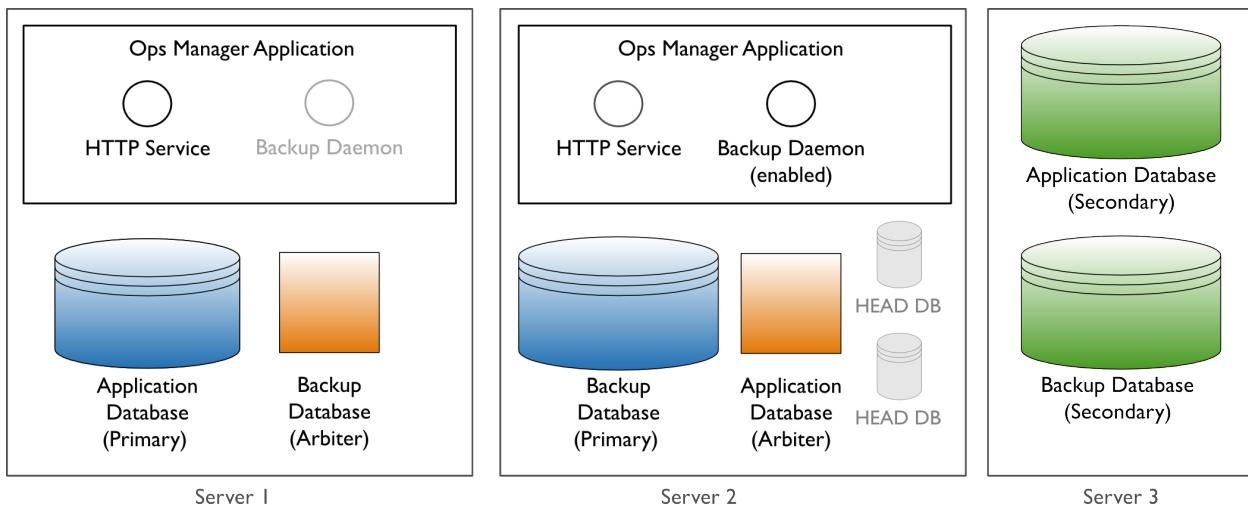
---

**Important:** This deployment does not provide high availability for the Ops Manager Application. Ops Manager uses `w:2` write concern, which requests acknowledgement of a write operation from both data-bearing members. The loss of one data-bearing node from the *Ops Manager Application Database* must be remedied before Ops Manager can resume healthy operation. To make the deployment durable, run each with three data-bearing members and enable journaling.

---

**Note:** All servers must satisfy the combined *hardware and software requirements* for both the systems specified in the **Meets System Requirements for** column.

---



Server	Meets System Requirements for	Purpose
1	<ul style="list-style-type: none"> <li>• Ops Manager Application</li> <li>• Ops Manager Application database</li> </ul>	Hosts the primary Ops Manager Application database and the arbiter for the backup database.
2	<ul style="list-style-type: none"> <li>• Ops Manager Application</li> <li>• Backup database</li> </ul>	Hosts the primary backup database and the arbiter for the Ops Manager Application database.
3	<ul style="list-style-type: none"> <li>• Ops Manager Application database</li> <li>• Backup database</li> </ul>	<p>Hosts replica set members for the Ops Manager Application Database and Backup Database.</p> <p>Replica sets provide data redundancy and are strongly recommended, but are not required for Ops Manager.</p>

For an example tutorial on installing the minimally viable Ops Manager installation, see <http://docs.opsmanager.mongodb.com//tutorial/install-basic-deployment>.

### Highly Available Ops Manager Application and Multiple Backup Databases

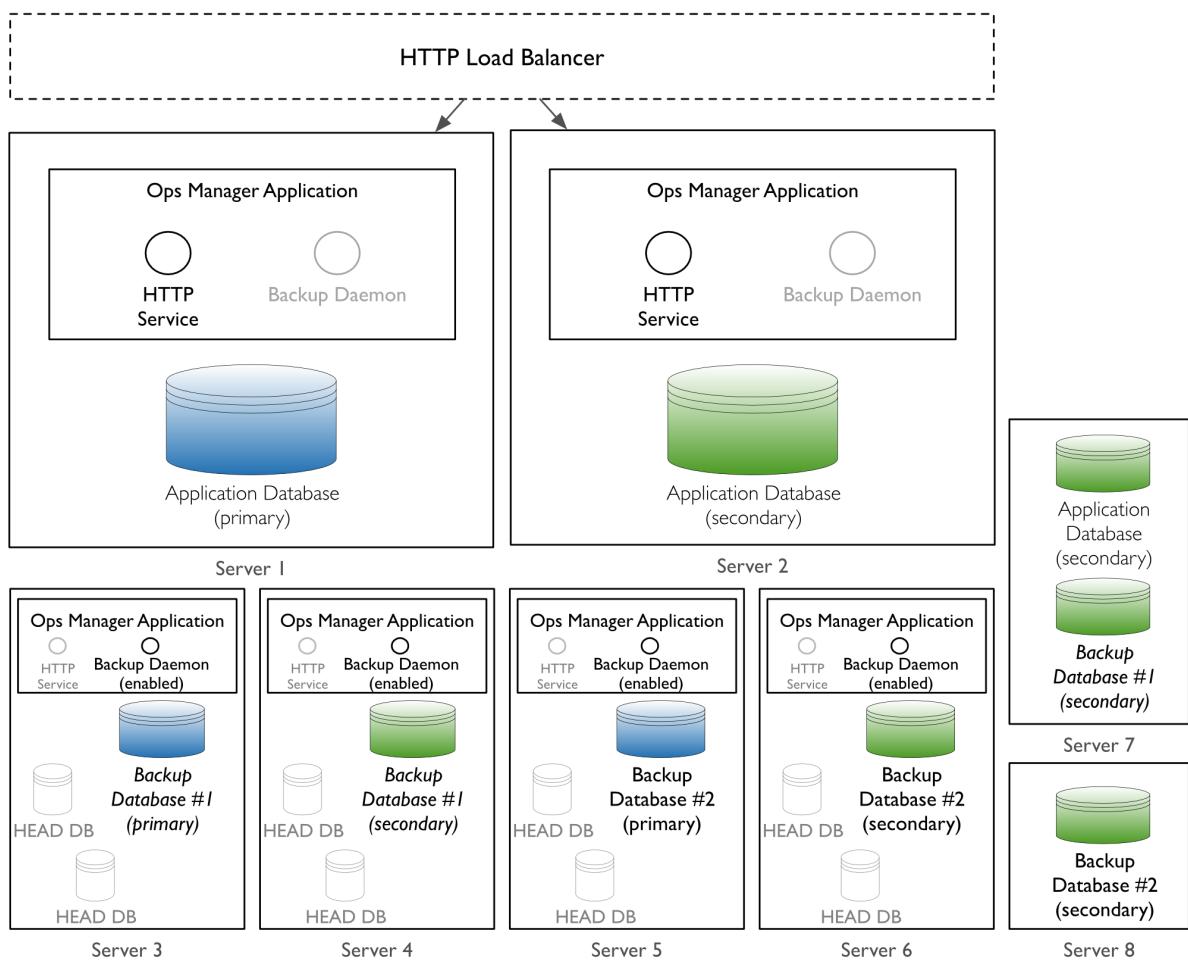
This Ops Manager deployment provides high availability for the Ops Manager Application by running multiple instances behind a load balancer. This deployment scales out to add an additional Backup Database.

The deployment includes:

- two servers that host the Ops Manager Application and the Ops Manager Application Database
- four servers that host Ops Manager Application with Backup enabled and Backup Databases
- additional servers to host the remaining members of each replica set

Deploy an HTTP Load Balancer to balance the HTTP traffic for the Ops Manager Application. Ops Manager does not supply an HTTP Load Balancer: you must deploy and configure it yourself. A load balancer placed in front of the Ops Manager Application servers must not return cached content.

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



**Note:** All servers must satisfy the combined *hardware and software requirements* for both the systems specified in the **System Requirements** column.

Server	Meets System Requirements for	Purpose
<b>1 &amp; 2</b>	<ul style="list-style-type: none"><li>• Ops Manager Application</li><li>• Ops Manager Application Database</li></ul>	Hosts the primary and secondary for the Ops Manager Application database.
<b>3, 4, 5 &amp; 6</b>	<ul style="list-style-type: none"><li>• Ops Manager Application</li><li>• Backup Database</li></ul>	Hosts the primary and secondary for the two backup databases. Only the Backup Daemon needs to communicate with the head databases. As such, their <code>net.bindIp</code> value is <code>127.0.0.1</code> to prevent external communication. <code>net.bindIp</code> specifies the IP address that <code>mongod</code> and <code>mongos</code> listens to for connections coming from applications.
<b>7 &amp; 8</b>	<ul style="list-style-type: none"><li>• Ops Manager Application database</li><li>• Backup database</li></ul>	Host the remaining replica set members for the Ops Manager Application Database and for the two Backup Databases.

For the procedure to install Ops Manager with high availability, see [Configure a Highly Available Ops Manager Application](#).

## 1.4 Install a Simple Test Ops Manager Installation

### On this page

- [Overview](#)
- [Procedure](#)

### Overview

To evaluate Ops Manager, you can install 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.

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 activate the Backup feature, in which case you would install *Backup Data Storage* on the same server as the other Ops Manager components. The Backup 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:

**15 GB memory and 50 GB disk space:** 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 to 50 GB. When you log into the instance, execute `df -h` to verify the root partition has 50 GB of space.

**Root access:** 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-org-3.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/amazon/2013.03/mongodb-org/3.2/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/appdb
sudo chown -R mongod:mongod /data
```

### Step 5: Create the backup directory. (Optional)

To also configure the Backup feature, issue following additional commands for the Backup Database:

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

### Step 6: Start MongoDB for the Application Database.

Start MongoDB as the `mongod` user, specifying:

- Port 27017
- The /data/appdb path for data files (--dbpath)
- The /data/appdb/mongodb.log path for logs (--logpath)
- The --fork option to run the process in the background and maintain control of the terminal.

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

#### **Step 6: Start MongoDB for the Backup Database. (Optional)**

Start MongoDB as the mongod user, specifying:

- Port 27018
- The /data/backup path for data files (--dbpath)
- The /data/backup/mongodb.log path for logs (--logpath)
- The --fork option to run the process in the background and maintain control of the terminal.

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

#### **Step 7: Download the Ops Manager package.**

1. In a browser, go to <http://www.mongodb.com> and click on the **Download** button.
2. Complete the form.
3. On the *MongoDB Download Center* page, click on the *Ops Manager* tab.
4. Select RedHat 5+ / CentOS 5+ / SUSE 11+ / Amazon Linux from the *Platforms* drop-down menu.
5. Select RPM from the *Packages* drop-down menu.
6. Click *Download*.

---

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

---

#### **Step 8: Install Ops Manager.**

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
```

The install creates the following:

- The base directory for the Ops Manager software, which is:  
`/opt/mongodb/mms/`
- A new system user, `mongodb-mms`, under which the server runs.
- The `/opt/mongodb/mms/conf/conf-mms.properties` file, which contains the connection string to access the Application Database. The default is `localhost`, port 27017, so no changes are necessary.

### **Step 9: Only if you installed to RHEL 7.1 or 7.2, replace symlinks with scripts.**

As a result of [RHEL Bug 1285492](#), the version of `systemd` found in RHEL 7.1 and 7.2 Ops Manager will not start automatically on boot up.

To work around this:

1. Replace the symbolic links to the scripts with the scripts themselves. Copy the scripts from `/opt/mongodb/mms/bin/mongodb-mms` to `/etc/init.d`.
2. In the copied scripts, change the line:

```
ABS_PATH=$( resolvepath $0 )"
```

to:

```
SCRIPTPATH=/opt/mongodb/mms/bin/mongodb-mms
ABS_PATH=$( resolvepath $SCRIPTPATH )"
```

### **Step 10: Start Ops Manager.**

Issue the following:

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

### **Step 11: Get your server's host name.**

If you are using an EC2 instance, the host name is the Public DNS listed on the EC2 instance's Description tab.

If you do not have a hostname, you can instead use the public IP address. To get the public IP address, issue the following:

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

### **Step 12: Open the Ops Manager home page and register the first user.**

1. In a browser, enter the following URL using the server's host name. If do not have a hostname, use the public IP address instead:

```
http://<host_name>:8080
```

2. 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.

### **Step 13: Configure Ops Manager.**

Ops Manager walks you through several configuration pages. Required settings are marked with an asterisk. Enter information as appropriate. When configuration is complete, Ops Manager opens the *Deployment* page.

#### **Step 14: If you will backup your instance, configure the backup capabilities. (Optional)**

1. Create a directory to store the *head databases*.

---

**Important:** Do not use this directory for any other purpose:

---

```
mkdir /data/backupDaemon
```

2. Make the directory writable by the `mongodb-mms` user:

```
sudo chown mongodb-mms:mongodb-mms /data/backupDaemon
```

3. In Ops Manager, while logged in as the user you registered during installation, click the *Admin* link at the top right of the page.
4. Click the *Backup* tab.
5. Follow the prompts to configure the Backup storage. Ops Manager walks you through the configuration.

For *snapshot storage*, select either the local filesystem or the Backup Database. If you use a File System Store, you still need a small MongoDB database for *oplog storage*. At the prompt to configure the connection string to the Backup Database, enter the following:

```
mongodb://localhost:27018
```

#### **Step 15: Set up your first deployment.**

1. Click *MongoDB Ops Manager* in the upper left corner of the page to return to the *Deployment* page.
2. Click *Add* to deploy a MongoDB instance.

## **2 Install Ops Manager**

**Installation Checklist** Prepare for your installation.

**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 MongoDB replica sets that host the Ops Manager Application Database and Backup Database.

**Install Backing Databases** Set up the Ops Manager Application Database and Backup Database.

**Install Ops Manager** Operating-system specific instructions for installing Ops Manager.

**Advanced Configuration Options** Procedures for optional configurations performed during installation.

**Upgrade Ops Manager** Operating-system specific instructions for upgrading Ops Manager.

### **2.1 Installation Checklist**

## On this page

- [Overview](#)
- [Topology Decisions](#)
- [Security Decisions](#)
- [Backup Decisions](#)

## 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 [Introduction](#), please do so now. The introduction describes the [Ops Manager components](#) and [common topologies](#).

The sequence for installing Ops Manager is to:

1. Plan your installation according to the questions on this page.
2. Provision servers that meet the [Ops Manager Hardware and Software Requirements](#).

**Warning:** Failure to configure servers according to the [Ops Manager Hardware and Software Requirements](#), including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

3. [Install the Application Database and optional Backup Database](#).
4. [Install Ops Manager](#).

---

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

---

## Topology Decisions

### Do you require redundancy and/or high availability?

The *topology* you choose for your deployment affects the redundancy and availability of both your metadata and snapshots, and the availability of the Ops Manager Application.

Ops Manager stores application metadata and snapshots in the Ops Manager Application Database and Backup Database respectively. To provide data redundancy, 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. A load balancer placed in front of the Ops Manager Application must not return cached content. For more information, see [Configure a Highly Available Ops Manager Application](#).

The following tables describe the pros and cons for different topologies.

**Test Install** This deployment runs on one server and has no data-redundancy. If you lose the server, you must start over from scratch.

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

**Production Install with Redundant Metadata and Snapshots** This install runs on at least three servers and provides redundancy 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.

<b>Pros:</b>	Can run on as few as three servers. Ops Manager metadata and backups are redundant from the perspective of the Ops Manager Application.
<b>Cons:</b>	No high availability, neither for the databases nor the application: <ol style="list-style-type: none"><li>1. If the Ops Manager Application Database or the Backup Database loses a data-bearing member, 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.</li><li>2. Loss of the Ops Manager instance requires you to manually start a new Ops Manager instance. No Ops Manager functionality is available while the application is down.</li></ol>

**Production Install with Highly Available Metadata and Snapshots** 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 *Production Install with Redundant Metadata and Snapshots*.

<b>Pros:</b>	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.
<b>Cons:</b>	Loss of the Ops Manager instance requires you to manually start a new Ops Manager instance. No Ops Manager functionality is available while the application is down.

**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](#).

<b>Pros:</b>	Ops Manager continues to be available even when any individual server is lost.
<b>Cons:</b>	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 Servers Have 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 using LDAP with 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 instance, Backup Daemon, and Backup Agent. The separate Ops Manager 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 instance, which you can configure through either different hostnames or split-horizon DNS. For detailed requirements, see [\*Assign Snapshot Stores to Specific Data Centers\*](#).

## 2.2 Ops Manager Hardware and Software Requirements

### On this page

- [Hardware Requirements](#)
- [Network Requirements](#)
- [Software Requirements](#)

This page describes the hardware and software requirements for the servers that run the *Ops Manager components*. Before deploying servers, use the [Installation Checklist](#) to plan your configuration.

For requirements for the MongoDB instances running as the Ops Manager Application Database and the Backup Database, see [Install the Ops Manager Application Database and Backup Database](#).

### Hardware Requirements

Each server must meet the sum of **RAM** and **disk space** requirements for all its components. For example, a server that runs Ops Manager and hosts the Ops Manager Application Database must meet the sum of the RAM and disk space required for Ops Manager plus the RAM and disk space required for the Application Database.

The servers also must meet the MongoDB ports requirements described in [Firewall Configuration](#). Each server's firewall rules must allow access to the required ports.

If you install on AWS servers, see also [Network Requirements](#) section on this page.

### Ops Manager Hardware Requirements

Every server that runs Ops Manager must meet the following hardware requirements:

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 Executive	Contact MongoDB Account Executive

### Ops Manager Application Database Hardware Requirements

The *Ops Manager Application Database* runs as a three-member *replica set* that runs on dedicated servers. Each server that hosts a MongoDB process for the Application Database **must** comply with the [Production Notes](#) in the MongoDB manual, which include information on NUMA, ulimits, and other configuration options.

**Warning:** Failure to configure servers according to the [MongoDB Production Notes](#) can lead to production failure.

Every server that hosts the Ops Manager Application Database must meet the following hardware requirements. For the best results use SSD-backed storage.

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

### Backup Daemon Hardware Requirements

Each server on which you activate the Backup Daemon must meet the requirements here **in addition to** the *requirements for Ops Manager*. Each server also must meet the configuration requirements in the [MongoDB Production Notes](#).

**Warning:** Failure to configure servers according to the [MongoDB Production Notes](#) can lead to production failure.

Before getting started with Backup, **contact your MongoDB Account Executive** for assistance in estimating the storage requirements for your Backup Daemon server.

The Backup Daemon creates *head databases* that replicate data for each replica set assigned to the Daemon. Typically, each server on which you activate 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. Specifically each server must have:

- the disk space and write capacity to maintain each head database, **plus**
- the disk space to store an additional copy of the data for each head database in order to support *point-in-time restores*.

Each server that hosts an active Backup Daemon has the following hardware requirements.

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

### Backup Database Hardware Requirements

If you use Ops Manager Backup, you must provision servers for the Backup Database. Server requirements for the Backup Database vary, depending on whether you will store your snapshots there. The Backup Database always holds oplog data.

If you store snapshots in the Backup Database, its servers typically must have enough capacity to store 2 to 3 times the total backed-up production data size. Snapshots are compressed and de-duplicated at the block level in the *Backup Blockstore Database*. Your specific requirements depend on your data compressibility and change rate. **Contact your MongoDB Account Manager** for assistance in estimating the use-case and workload-dependent storage requirements for your Backup Database servers.

Each server for the Backup Database **must** comply with the [Production Notes](#) in the MongoDB manual, which include information on NUMA, ulimits, and other configuration options.

**Warning:** Failure to configure servers according to the [MongoDB Production Notes](#) can lead to production failure.

If you store snapshots in the Backup Database, each data-bearing member must meet the following requirements:

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.

If you will **not** store snapshots in the Backup Database, each data-bearing member must meet the following requirements:

CPU Cores	Disk Space
4 x 2ghz+	The size of your oplogs compressed for the <i>configured point in time window</i> . The default is 24 hours.

## Network Requirements

### Accessible Ports

The Ops Manager application must be able to connect to users and Ops Manager agents over or Secure HTTP (HTTPS). Ops Manager agents must be able to connect to MongoDB client MongoDB databases.

Though Ops Manager only requires open HTTP(S) and MongoDB network ports to connect with users and to databases, what ports are opened on a firewall depend upon what capabilities are enabled: encryption, authentication and monitoring.

This page defines which systems need to connect to which ports on other systems.

---

**Important:** To run Ops Manager without an Internet connection, see [Configure Local Mode if Servers Have No Internet Access](#) to ensure you have all of the necessary binaries to run Ops Manager without an Internet connection.

---

### Ports Required to Use Ops Manager

- Both Ops Manager users and Ops Manager agents must be able to connect to Ops Manager application over HTTP(S). See [Manage Ops Manager Ports](#) to set a non-default port for Ops Manager.
- Ops Manager must be able to connect to the backing MongoDB databases running mongod.
- For each Ops Manager group, Ops Manager agents must be able to connect to all client MongoDB processes (mongod or mongos).
- The Ops Manager application must also be able to send email to Ops Manager users.

To use Ops Manager, open the following ports to the specified servers.

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
HTTP	8080	TCP	Inbound	Web connection to Ops Manager from users and Ops Manager agents.	No
HTTPS	8443	TCP	Inbound	Web connection to Ops Manager from users and Ops Manager agents.	Yes
HTTP(S)	8090	TCP	Inbound	<p>Provides a health-check endpoint for monitoring Ops Manager through a monitoring service like Zabbix or Nagios. This is disabled by default.</p> <p>To enable it, see <a href="#">Enable the Health Check Endpoint</a>. When enabled, you can access the endpoint at:</p> <pre>http://&lt;opsmanagerhost&gt;:8090/healthcheck</pre> <p>The API endpoint provides the ability to check connections from the HTTP Service to the <a href="#">Ops Manager Application Database</a> and the <a href="#">Backup Snapshot Storage</a>.</p> <p>A successful response returns the following:</p> <pre>{   "mms_db": "OK",   "backup_db": "OK" }</pre>	Optional
MongoDB	27000 - 28000	TCP	Both	Connect to MongoDB application, backup and client databases.	Optional
SMTP	25	TCP	Outbound	Send emails from Ops Manager.	No
SMTPS	465	TCP	Outbound	Send emails from Ops Manager.	Yes

**Ports Needed to Administer Ops Manager** Most Ops Manager administration can be performed through the user interface. Some procedures require access to the operating system. To permit your administrators to access your Ops Manager and MongoDB servers, open the following ports to those servers.

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
Secure Shell (ssh)	22	TCP	Inbound	Linux System administration.	Yes
Remote Desktop Connection (RDP)	3389	TCP	Inbound	Windows System administration.	No

**Ports Needed to Integrate Ops Manager with SNMP** To send and receive SNMP messages to and from your MongoDB deployments must open the following ports between Ops Manager and your SNMP Manager.

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
SNMP	162	UDP	Outbound	Send Traps to SNMP Manager.	No
SNMP	11611	UDP	Inbound	Receive requests from SNMP Manager.	No

See [SNMP Heartbeat Settings](#) to set SNMP to use non-standard ports.

**Ports Needed to Authenticate with Ops Manager** MongoDB Enterprise users [can use LDAP](#) to authenticate Ops Manager users. To authenticate using LDAP, open the following ports on Ops Manager and your LDAP server.

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
LDAP	389	UDP	Both	Authenticate and/or authorize Ops Manager users against LDAP server.	No
LDAPS	636	UDP	Both	Authenticate and/or authorize Ops Manager users against LDAP server.	Yes

See [Authentication through LDAP](#) to configure LDAP URI strings including ports.

**Ports Needed to Authenticate with MongoDB** MongoDB Enterprise users can use Kerberos or LDAP to authenticate MongoDB users. To authenticate using LDAP or Kerberos, open the following ports between the MongoDB client databases, Ops Manager and the Kerberos or LDAP server(s).

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
Kerberos	88	TCP / UDP	Out-bound	Request authentication for MongoDB users against Kerberos server.	No
Kerberos	88	UDP	In-bound	Receive authentication for MongoDB users against Kerberos server.	No
LDAP	389	UDP	Both	Authenticate and/or authorize MongoDB users against LDAP server.	No
LDAPS	636	UDP	Both	Authenticate and/or authorize MongoDB users against LDAP server.	Yes

See [Kerberos Authentication to the Application Database](#) to configure Kerberos for authentication to the Ops Manager application database.

### Use resolvable hostnames

Each Agent and Ops Manager instance must be able to resolve the hostname for each server hosting a MongoDB instance or Ops Manager agent.

On each server, set their hostnames to fully qualified domain names (FQDN) whenever possible. Consult the documentation for your operating system as to how to find and set the hostname as an FQDN.

Setting the FQDN on each server helps you know which server you are using when logged into that server. To enable other servers to know what the other servers' hostnames are, you need to provide a way for those servers to resolve hostnames.

There are two ways to configure hostname resolution.

**Use a Domain Name Service** To make the servers' hostnames resolvable, run a server with a domain name service (DNS). DNS maps IP addresses to hostnames with a given domain (such as `example.com`). This DNS server should have an entry for each server in the deployment: Ops Manager, Ops Manager agent and MongoDB. Entries for LDAP, Kerberos, SNMP and email servers as well as load balancers would be recommended.

**Edit Host Files** If a DNS setup is not possible, add entries for each server in the `hosts` file of each system.

Table 1: Locations of hosts files

Operating System	hosts Location
Linux	<code>/etc/hosts</code>
Mac OS X	<code>/private/etc/hosts</code>
Windows	<code>%SystemRoot%\System32\drivers\etc\hosts</code> This normally resolves to: <code>C:\Windows\System32\drivers\etc\hosts</code>

The `hosts` file is a root-readable plain text and must be edited with `root` or `Administrator` permissions. The entry format is written as:

```
127.0.0.1 localhost
10.15.0.5 opsmgr.example.dev
10.15.10.15 rs1.example.dev
10.15.10.16 rs2.example.dev
10.15.10.17 rs3.example.dev
```

## Software Requirements

Servers that run Ops Manager components must meet the software requirements described here.

### Ops Manager Operating System

Servers that run Ops Manager must run on a 64-bit version of one 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

### MongoDB Version for Ops Manager Application Database and Backup Database

The Ops Manager Application Database and the Backup Database must run at least MongoDB 2.6 with the following specific requirements:

- If running MongoDB 2.6 version, must run MongoDB version 2.6.6 or later 2.6 versions.
- If running MongoDB 3.0 version, must run MongoDB version 3.0.6, or later.
- If running MongoDB 3.2 or later, can run any MongoDB version 3.2 or later.

---

**Important:** This requirement is for the MongoDB instances that serve as the Ops Manager Application Database and the Backup Database. The requirement is **not** for the MongoDB deployments that can be managed by Ops Manager. For the minimum MongoDB versions required for MongoDB deployments for management, see [MongoDB Compatibility](#).

---

### Ulimits Requirements

For ulimit requirements for the servers that run MongoDB (i.e., the Backup Daemon servers and the servers that host the Ops Manager Application Database or the Backup Database), see all of the following:

- [MongoDB Production Notes](#)
- [MongoDB Ulimit Settings](#) in the MongoDB manual

The Ops Manager package automatically raises the following ulimits:

- open files
- max user processes
- virtual memory

RHEL places a max process limitation of 1024 which overrides ulimit settings. If /etc/security/limits.d/99-mongodb-nproc.conf does not exist, create the file with new soft nproc and hard nproc values to increase the process limit. See /etc/security/limits.d/90-nproc.conf file as an example.

### 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](#).

## Client Web Browsers

Clients should use Ops Manager-supported browsers and will display a warning on non-supported browsers. Ops Manager supports the following browsers:

- Chrome latest stable
- Firefox latest stable
- Internet Explorer 10 (IE10) and greater
- Safari latest stable version on MacOS 10.10.5+

## 2.3 Install the Ops Manager Application Database and Backup Database

### On this page

- [Overview](#)
- [Replica Set Topology](#)
- [Replica Sets Requirements](#)
- [Server Prerequisites](#)
- [Choosing a Storage Engine for the Backing Databases](#)
- [Procedure](#)

### Overview

Before you install Ops Manager you must install the [\*Ops Manager Application Database\*](#), which holds Ops Manager operational data. If you will use the Ops Manager Backup feature, you must also install the Backup Database, which stores oplog data, temporary sync data, and, depending on your configuration, stores your snapshots.

Both databases run on dedicated MongoDB [\*replica sets\*](#). If you use multiple Backup Databases, set up separate replica sets for each database. The replica set for each database is dedicated to that database only and **must store no other data**.

The application database and the backup database(s) run as separate replica sets to ensure that the Ops Manager Application would keep running should you accidentally fill the Backup Database's volume. If that occurred and both databases were members of the same replica set, Ops Manager would become inoperable.

---

**Note:** You cannot use Ops Manager to deploy the backing database. You must deploy and manage the backing databases manually.

---

### Replica Set Topology

Each replica set usually comprises three data-bearing members to provide high availability. 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 only two data-bearing members, and if you lose one of the data-bearing members, MongoDB blocks write operations.

For the Ops Manager Application Database, you can optionally run one member of the replica set on the same physical server as Ops Manager. For *testing only*, you may use a standalone MongoDB deployment in place of each replica set.

## Replica Sets Requirements

The replica sets that host the Ops Manager Application Database and Backup Database(s) must:

- **Dedicated Databases:** The replica sets must store data **only** in support of Ops Manager operations. They must store **no other data**.
- **MongoDB Version:** The replica sets must run MongoDB 2.6.6 or later. If they run a 3.0+ version of MongoDB, the replica sets must run MongoDB version 3.0.6 or later.
- **Storage Engine:** Use the storage engine appropriate to your workload. In most cases, this is MMAPv1. See the *storage engine considerations* below.
- **Settings:**
  - Set the MongoDB `security.javascriptEnabled` option 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.
  - **Do not** run the MongoDB `notableScan` option.
- **Credentials:** The replica sets must include user credentials for accessing the backing database. When you install Ops Manager, you will include the user credentials in the `mongo.mongoUri` setting. You will also specify the authentication mechanism if the database uses a mechanism other than default MongoDB authentication.

## Server Prerequisites

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

- The hardware requirements described in *Ops Manager Application Database Hardware Requirements* or *Backup Database Hardware Requirements*, 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 information on ulimits, NUMA, 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:**

RHEL places a max process limitation of 1024 which overrides ulimit settings. If `/etc/security/limits.d/99-mongodb-nproc.conf` does not exist, create the file with new soft nproc and hard nproc values to increase the process limit. See `/etc/security/limits.d/90-nproc.conf` file as an example.

For more information, see [UNIX ulimit Settings](#) in the MongoDB manual.

## Choosing a Storage Engine for the Backing Databases

MongoDB provides both the MMAPv1 and WiredTiger storage engines. The Ops Manager Application and Backup Database schemas have been designed to achieve maximum performance on MMAPv1. Therefore, while WiredTiger outperforms MMAPv1 on typical workloads, we recommend MMAPv1 for the unique Ops Manager workload.

The MMAPv1 engine also minimizes storage in the Backup Database's blockstore, as all the blocks are already compressed and padding is disabled. There is typically no further storage benefit to be gained by taking advantage of WiredTiger compression.

In one case, WiredTiger might be preferable. Backups of insert-only MongoDB workloads that benefit from high levels of block-level de-duplication in the blockstore may see a compression-based storage reduction when running the Backup Database on the WiredTiger storage engine.

The storage engine used by the Backup Database is independent from the storage engine chosen to store a deployment's snapshots. If the Backup Database uses the MMAPv1 storage engine, it can store backup snapshots for WiredTiger backup jobs in its blockstore.

## Procedure

To deploy MongoDB replica sets to host the Ops Manager Application and Backup Databases:

### Step 1: Provision servers.

Use servers that meet the above *Server Prerequisites*.

**Warning:** Failure to configure servers according to the [MongoDB Production Notes](#) can lead to production failure.

### Step 2: Install MongoDB on each server.

To install MongoDB, see [Install MongoDB](#) in the MongoDB manual. If you choose to install [MongoDB Enterprise](#) for the backing database, you must install the MongoDB Enterprise dependencies, as described in the install procedures.

### Step 3: Deploy the replica set for each database.

Deploy separate replica sets for the Ops Manager Application Database and for each Backup Database. To deploy replica sets, see [Deploy a Replica Set](#) in the MongoDB manual.

## 2.4 Install Ops Manager

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

**Install with DEB Package** Install Ops Manager on Debian and Ubuntu systems.

**Install from Archive 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 an `rpm` Package

#### On this page

- Overview
- Prerequisites
- Install Ops Manager
- Next Steps

## Overview

This tutorial describes how to install Ops Manager using an `rpm` package. If you are instead upgrading an existing deployment, please see [Upgrade Ops Manager](#).

## Prerequisites

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

Before you install Ops Manager, you must:

1. Plan your configuration. See [Installation Checklist](#).
2. Deploy servers that meet the [Ops Manager Hardware and Software Requirements](#).

**Warning:** Failure to configure servers according to the [Ops Manager Hardware and Software Requirements](#), including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

3. *Install the Ops Manager Application Database and optional Backup Database.* The databases require dedicated MongoDB instances. Do **not** use MongoDB installations that store other data. The Backup Database is required only if you will use the Backup feature.

The Ops Manager Application and Backup Daemon must authenticate to the backing databases as a MongoDB user with appropriate access. See `mongo.mongoUri` for more information.

---

**Note:** Ops Manager cannot deploy its own backing databases. You must deploy those databases manually.

---

## Install Ops Manager

To install Ops Manager:

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

1. In a browser, go to <http://www.mongodb.com> and click on the [Download](#) button.
2. Complete the form.
3. On the [MongoDB Download Center](#) page, click on the [Ops Manager](#) tab.
4. Select RedHat 5+ / CentOS 5+ / SUSE 11+ / Amazon Linux from the *Platforms* drop-down menu.
5. Select RPM from the *Packages* drop-down menu.
6. Click [Download](#).

---

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

---

**Step 2: Install the Ops Manager package on each server being used for Ops Manager.** 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: Only if you installed to RHEL 7.1 or 7.2, replace symlinks with scripts.** As a result of [RHEL Bug 1285492](#), the version of `systemd` found in RHEL 7.1 and 7.2 Ops Manager will not start automatically on boot up.

To work around this:

1. Replace the symbolic links to the scripts with the scripts themselves. Copy the scripts from `/opt/mongodb/mms/bin/mongodb-mms` to `/etc/init.d`.
2. In the copied scripts, change the line:

```
ABS_PATH=$( resolvepath $0 )"
```

to:

```
SCRIPTPATH=/opt/mongodb/mms/bin/mongodb-mms
ABS_PATH=$( resolvepath $SCRIPTPATH )"
```

**Step 4: Configure the Ops Manager connection to the Ops Manager Application Database.** On a server that is to run the Ops Manager, open `/opt/mongodb/mms/conf/conf-mms.properties` with root privileges and configure the settings described here, as appropriate.

Configure the following setting to provide the connection string Ops Manager uses to connect to the database:

- `mongo.mongoUri`

If you will configure Ops Manager to use the Ops Manager Application Database over SSL, configure the following *SSL settings*:

- `mongo.ssl`
- `mongodb.ssl.CAFile`
- `mongodb.ssl.PEMKeyFile`
- `mongodb.ssl.PEMKeyFilePassword`

Ops Manager also uses these settings for SSL connections to Backup Databases

If you will configure Ops Manager to use Kerberos to manage access to the Ops Manager Application Database, configure the following *Kerberos settings*:

- `jvm.java.security.krb5.kdc`
- `jvm.java.security.krb5.realm`
- `mms.kerberos.principal`
- `mms.kerberos.keyTab`

**Step 5: On the same server, start Ops Manager.** Issue the following command:

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

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

1. Enter the following URL in a browser, where <host> is the fully qualified domain name of the server:

`http://<host>:8080`

2. 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.

**Step 7: Configure Ops Manager.** Ops Manager walks you through several configuration pages. Required settings are marked with an asterisk. Enter information as appropriate. When configuration is complete, Ops Manager opens the *Deployment* page.

In addition to the common required settings, the following are required for particular deployment configurations. For more information on a setting, see *Ops Manager Configuration*.

Configuration	Required Settings
If you are running multiple Ops Manager instances behind a load balancer	Set <i>Load Balancer Remote IP Header</i> to the name of the header the load balancer will use when forwarding the client's IP address to the application server. If you set this, do not allow clients to connect directly to any of the application servers. The load balancer must not return cached content. You will set up the additional servers as part of the next steps in this procedure.
If you are using Automation or Backup without an internet connection	Set the <i>MongoDB Version Management</i> settings. You will need to put the tarballs for every MongoDB release used in your deployment in the configured release directory on every Ops Manager server. For more information, see <i>Configure Local Mode if Servers Have No Internet Access</i> .

**Step 8: Copy the gen.key file from the current server to the other servers.** Ops Manager requires an identical `gen.key` file be stored on both servers running Ops Manager and uses the file to encrypt data at rest in the Ops Manager Application Database and Backup Database.

You must copy the `gen.key` file from the current server, on which you just installed Ops Manager, to every server that will run Ops Manager. You must copy the `gen.key` to the other servers **before** starting Ops Manager on them.

Use `scp` to copy the `gen.key` file from the `/etc/mongodb-mms/` directory on the current server to the same directory on the other servers.

**Step 9: If you will run multiple Ops Manager Applications behind a load balancer, configure and start the applications.** For each Ops Manager instance, repeat the step to configure the connection to the Ops Manager Application Database and the step to start the application.

For more information on running multiple applications behind a load balancer, see *Configure a Highly Available Ops Manager Application*.

#### **Step 10: If you will run Ops Manager Backup, configure the Backup Daemon and Backup Storage.**

1. On each Ops Manager server that will be activated as a Backup Daemon, create the directory that will store the *head databases*. The directory must be:
  - a dedicated disk partition that is not be used for any other purpose.
  - sized appropriately according to the *Ops Manager Hardware and Software Requirements*.
  - writable by the `mongodb-mms` user.
2. If you will store snapshots on a filesystem instead of the Backup Database, create the directory that will store snapshots.

3. Open Ops Manager and make sure you are logged in as the user you registered when installing Ops Manager. This user is the *global owner*.
4. Click the *Admin* link at the top right of the page.
5. Click the *Backup* tab.
6. Follow the prompts to configure the Backup Daemon and Backup Storage. Ops Manager walks you through configuration of the daemon and snapshot storage.

After you select how to store snapshots, you are prompted to configure the connection string to the Backup Database. If you use filesystem storage for your snapshots, the Backup Database is used only for the *oplog store*.

**Warning:** Once the connection string is saved, any change to the string requires you to restart all the Ops Manager instances, including those running activated Backup Daemons. Making the change and clicking *Save* is not sufficient. Ops Manager will continue to use the previous string until you restart the instances.

<p>&lt;hostname&gt;:&lt;port&gt;</p> <p><i>MongoDB Auth Username</i> and <i>MongoDB Auth Password</i> <i>Encrypted Credentials</i></p> <p><i>Use SSL</i></p> <p><i>Connection Options</i></p>	<p>Enter a comma-separated list of the fully qualified domain names and port numbers for all <i>replica set</i> members for the Backup Database. Enter the user credentials if the database uses authentication.</p> <p>Check this if the user credentials use the Ops Manager <i>credentialstool</i>. For more information, see <i>Encrypt User Credentials</i>.</p> <p>Check this if the database uses SSL. If you select this, you must configure SSL settings Ops Manager. See <i>Ops Manager Configuration</i>.</p> <p>To add additional connection options, enter them using the MongoDB <i>Connection String URI Format</i>. This field supports un-escaped values only.</p>
---	---

## Next Steps

Once you have installed the Ops Manager web application to the Ops Manager server, you must next do the following:

1. Install Ops Manager agents on the servers that run your MongoDB deployments. You can install agents on servers running existing MongoDB deployments or on servers on which you will create new MongoDB deployments. Servers that run your MongoDB deployments must meet the requirements in the *MongoDB Production Notes* in the MongoDB Manual.

To install agents, see *Provision Servers*.

### See also:

*Install the Automation Agent*, *Install Monitoring Agent*, *Install Backup Agent*

2. After you install agents, deploy MongoDB to your servers to test connections. If you use Ops Manager Automation, you can deploy MongoDB through the Ops Manager interface. For example, see *Deploy a Replica Set*.

## Install Ops Manager with a deb Package

## On this page

- [Overview](#)
- [Prerequisites](#)
- [Install Ops Manager](#)
- [Next Steps](#)

## Overview

This tutorial describes how to install Ops Manager using a `deb` package. If you are instead upgrading an existing deployment, please see [Upgrade Ops Manager](#).

## Prerequisites

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

Before you install Ops Manager, you must:

1. Plan your configuration. See [Installation Checklist](#).
2. Deploy servers that meet the [Ops Manager Hardware and Software Requirements](#).

**Warning:** Failure to configure servers according to the [Ops Manager Hardware and Software Requirements](#), including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

3. *Install the Ops Manager Application Database and optional Backup Database.* The databases require dedicated MongoDB instances. Do **not** use MongoDB installations that store other data. The Backup Database is required only if you will use the Backup feature.

The Ops Manager Application and Backup Daemon must authenticate to the backing databases as a MongoDB user with appropriate access. See `mongo.mongoUri` for more information.

**Note:** Ops Manager cannot deploy its own backing databases. You must deploy those databases manually.

## Install Ops Manager

To install Ops Manager:

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

1. In a browser, go to <http://www.mongodb.com> and click on the [Download](#) button.
2. Complete the form.
3. On the [MongoDB Download Center](#) page, click on the [Ops Manager](#) tab.
4. Select Ubuntu 12.04+ from the [Platforms](#) drop-down menu.
5. Select DEB from the [Packages](#) drop-down menu.
6. Click [Download](#).

---

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

---

**Step 2: Install the Ops Manager package on each server being used for Ops Manager.** 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 the Ops Manager connection to the Ops Manager Application Database.** On a server that is to run the Ops Manager, open `/opt/mongodb/mms/conf/conf-mms.properties` with root privileges and configure the settings described here, as appropriate.

Configure the following setting to provide the connection string Ops Manager uses to connect to the database:

- `mongo.mongoUri`

If you will configure Ops Manager to use the Ops Manager Application Database over SSL, configure the following *SSL settings*:

- `mongo.ssl`
- `mongodb.ssl.CAFile`
- `mongodb.ssl.PEMKeyFile`
- `mongodb.ssl.PEMKeyFilePassword`

Ops Manager also uses these settings for SSL connections to Backup Databases

If you will configure Ops Manager to use Kerberos to manage access to the Ops Manager Application Database, configure the following *Kerberos settings*:

- `jvm.java.security.krb5.kdc`
- `jvm.java.security.krb5.realm`
- `mms.kerberos.principal`
- `mms.kerberos.keyTab`

**Step 4: On the same server, start Ops Manager.** Issue the following command:

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

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

1. Enter the following URL in a browser, where `<host>` is the fully qualified domain name of the server:  
`http://<host>:8080`
2. 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.

**Step 6: Configure Ops Manager.** Ops Manager walks you through several configuration pages. Required settings are marked with an asterisk. Enter information as appropriate. When configuration is complete, Ops Manager opens the *Deployment* page.

In addition to the common required settings, the following are required for particular deployment configurations. For more information on a setting, see *Ops Manager Configuration*.

Configuration	Required Settings
If are running multiple Ops Manager instances behind a load balancer	Set <i>Load Balancer Remote IP Header</i> to the name of the header the load balancer will use when forwarding the client's IP address to the application server. If you set this, do not allow clients to connect directly to any of the application servers. The load balancer must not return cached content. You will set up the additional servers as part of the next steps in this procedure.
If you are using Automation or Backup without an internet connection	Set the <i>MongoDB Version Management</i> settings. You will need to put the tarballs for every MongoDB release used in your deployment in the configured release directory on every Ops Manager server. For more information, see <i>Configure Local Mode if Servers Have No Internet Access</i> .

**Step 7: Copy the `gen.key` file from the current server to the other servers.** Ops Manager requires an identical `gen.key` file be stored on both servers running Ops Manager and uses the file to encrypt data at rest in the Ops Manager Application Database and Backup Database.

You must copy the `gen.key` file from the current server, on which you just installed Ops Manager, to every server that will run Ops Manager. You must copy the `gen.key` to the other servers **before** starting Ops Manager on them.

Use `scp` to copy the `gen.key` file from the `/etc/mongodb-mms/` directory on the current server to the same directory on the other servers.

**Step 8: If you will run multiple Ops Manager Applications behind a load balancer, configure and start the applications.** For each Ops Manager instance, repeat the step to configure the connection to the Ops Manager Application Database and the step to start the application.

For more information on running multiple applications behind a load balancer, see *Configure a Highly Available Ops Manager Application*.

### Step 9: If you will run Ops Manager Backup, configure the Backup Daemon and Backup Storage.

1. On each Ops Manager server that will be activated as a Backup Daemon, create the directory that will store the *head databases*. The directory must be:
  - a dedicated disk partition that is not be used for any other purpose.
  - sized appropriately according to the *Ops Manager Hardware and Software Requirements*.
  - writable by the `mongodb-mms` user.
2. If you will store snapshots on a filesystem instead of the Backup Database, create the directory that will store snapshots.
3. Open Ops Manager and make sure you are logged in as the user you registered when installing Ops Manager. This user is the *global owner*.
4. Click the *Admin* link at the top right of the page.
5. Click the *Backup* tab.
6. Follow the prompts to configure the Backup Daemon and Backup Storage. Ops Manager walks you through configuration of the daemon and snapshot storage.

After you select how to store snapshots, you are prompted to configure the connection string to the Backup Database. If you use filesystem storage for your snapshots, the Backup Database is used only for the *oplog store*.

**Warning:** Once the connection string is saved, any change to the string requires you to restart all the Ops Manager instances, including those running activated Backup Daemons. Making the change and clicking *Save* is not sufficient. Ops Manager will continue to use the previous string until you restart the instances.

<i>&lt;hostname&gt;:&lt;port&gt;</i>	Enter a comma-separated list of the fully qualified domain names and port numbers for all <i>replica set</i> members for the Backup Database.
<i>MongoDB Auth Username</i> and <i>MongoDB Auth Password</i> <i>Encrypted Credentials</i>	Enter the user credentials if the database uses authentication.
<i>Use SSL</i>	Check this if the user credentials use the Ops Manager <i>credentials</i> tool. For more information, see <i>Encrypt User Credentials</i> .
<i>Connection Options</i>	Check this if the database uses SSL. If you select this, you must configure SSL settings Ops Manager. See <i>Ops Manager Configuration</i> . To add additional connection options, enter them using the MongoDB <i>Connection String URI Format</i> . This field supports un-escaped values only.

## Next Steps

Once you have installed the Ops Manager web application to the Ops Manager server, you must next do the following:

1. Install Ops Manager agents on the servers that run your MongoDB deployments. You can install agents on servers running existing MongoDB deployments or on servers on which you will create new MongoDB deployments. Servers that run your MongoDB deployments must meet the requirements in the *MongoDB Production Notes* in the MongoDB Manual.

To install agents, see *Provision Servers*.

### See also:

*Install the Automation Agent*, *Install Monitoring Agent*, *Install Backup Agent*

2. After you install agents, deploy MongoDB to your servers to test connections. If you use Ops Manager Automation, you can deploy MongoDB through the Ops Manager interface. For example, see *Deploy a Replica Set*.

## Install Ops Manager from a `tar.gz` or `zip` Archive

### On this page

- Overview
- Prerequisites
- Install Ops Manager
- Next Steps

## Overview

This tutorial describes how to install Ops Manager to Linux servers from a `.tar.gz` or `zip` archive. If you are instead upgrading an existing deployment, please see [Upgrade Ops Manager](#).

## Prerequisites

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

Before you install Ops Manager, you must:

1. Plan your configuration. See [Installation Checklist](#).
2. Deploy servers that meet the [Ops Manager Hardware and Software Requirements](#).

**Warning:** Failure to configure servers according to the [Ops Manager Hardware and Software Requirements](#), including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

3. *Install the Ops Manager Application Database and optional Backup Database.* The databases require dedicated MongoDB instances. Do **not** use MongoDB installations that store other data. The Backup Database is required only if you will use the Backup feature.

The Ops Manager Application and Backup Daemon must authenticate to the backing databases as a MongoDB user with appropriate access. See `mongo.mongoUri` for more information.

---

**Note:** Ops Manager cannot deploy its own backing databases. You must deploy those databases manually.

---

## Install Ops Manager

To install Ops Manager:

### Step 1: Download the latest version of the Ops Manager archive.

1. In a browser, go to <http://www.mongodb.com> and click on the [Download](#) button.
2. Complete the form.
3. On the [MongoDB Download Center](#) page, click on the [Ops Manager](#) tab.
4. Select RedHat 5+ / CentOS 5+ / SUSE 11+ / Amazon Linux or Ubuntu 12.04+ from the [Platforms](#) drop-down menu.
5. Select TAR.GZ from the [Packages](#) drop-down menu.
6. Click [Download](#).

---

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

---

**Step 2: Install the Ops Manager package on each server being used for Ops Manager.** Navigate to the directory to which to install Ops Manager. Extract the archive to that directory:

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

When complete, Ops Manager is installed.

**Step 3: Configure the Ops Manager connection to the Ops Manager Application Database.** On a server that is to run Ops Manager, open <install\_directory>/conf/conf-mms.properties with root privileges and configure the settings described here, as appropriate.

Configure the following setting to provide the connection string Ops Manager uses to connect to the database:

- mongo.mongoUri

If you will configure Ops Manager to use the Ops Manager Application Database over SSL, configure the following *SSL settings*.

- mongo.ssl
- mongodb.ssl.CAFile
- mongodb.ssl.PEMKeyFile
- mongodb.ssl.PEMKeyFilePassword

Ops Manager also uses these settings for SSL connections to Backup Databases

If you will configure Ops Manager to use Kerberos to manage access to the Ops Manager Application Database, configure the following *Kerberos settings*:

- jvm.java.security.krb5.kdc
- jvm.java.security.krb5.realm
- mms.kerberos.principal
- mms.kerberos.keyTab

**Step 4: Start Ops Manager.** Issue the following command:

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

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

1. Enter the following URL in a browser, where <host> is the fully qualified domain name of the server:

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

2. 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.

**Step 6: Configure Ops Manager.** Ops Manager walks you through several configuration pages. Required settings are marked with an asterisk. Enter information as appropriate. When configuration is complete, Ops Manager opens the *Deployment* page.

In addition to the common required settings, the following are required for particular deployment configurations. For more information on a setting, see *Ops Manager Configuration*.

Configuration	Required Settings
If you are running multiple Ops Manager instances behind a load balancer	Set <i>Load Balancer Remote IP Header</i> to the name of the header the load balancer will use when forwarding the client's IP address to the application server. If you set this, do not allow clients to connect directly to any of the application servers. The load balancer must not return cached content. You will set up the additional servers as part of the next steps in this procedure.
If you are using Automation or Backup without an internet connection	Set the <i>MongoDB Version Management</i> settings. You will need to put the tarballs for every MongoDB release used in your deployment in the configured release directory on every Ops Manager server. For more information, see <a href="#">Configure Local Mode if Servers Have No Internet Access</a> .

**Step 7: Copy the `gen.key` file from the current server to the other servers.** Ops Manager requires an identical `gen.key` file be stored on both servers running Ops Manager and uses the file to encrypt data at rest in the Ops Manager Application Database and Backup Database.

You must copy the `gen.key` file from the current server, on which you just installed Ops Manager, to every server that will run Ops Manager. You must copy the `gen.key` to the other servers **before** starting Ops Manager on them.

Use `scp` to copy the `gen.key` file from the  `${HOME} / .mongodb-mms /` directory on the current server to the same directory on the other servers.

**Step 8: If you will run multiple Ops Manager Applications behind a load balancer, configure and start the applications.** For each Ops Manager instance, repeat the step to configure the connection to the Ops Manager Application Database and the step to start the application.

For more information on running multiple applications behind a load balancer, see [Configure a Highly Available Ops Manager Application](#).

#### **Step 9: If you will run Ops Manager Backup, configure the Backup Daemon and Backup Storage.**

1. On each Ops Manager server that will be activated as a Backup Daemon, create the directory that will store the *head databases*. The directory must be:
  - a dedicated disk partition that is not be used for any other purpose.
  - sized appropriately according to the [Ops Manager Hardware and Software Requirements](#).
  - writable by the `mongodb-mms` user.
2. If you will store snapshots on a filesystem instead of the Backup Database, create the directory that will store snapshots.
3. Open Ops Manager and make sure you are logged in as the user you registered when installing Ops Manager. This user is the *global owner*.
4. Click the *Admin* link at the top right of the page.
5. Click the *Backup* tab.
6. Follow the prompts to configure the Backup Daemon and Backup Storage. Ops Manager walks you through configuration of the daemon and snapshot storage.

After you select how to store snapshots, you are prompted to configure the connection string to the Backup Database. If you use filesystem storage for your snapshots, the Backup Database is used only for the *oplog store*.

**Warning:** Once the connection string is saved, any change to the string requires you to restart all the Ops Manager instances, including those running activated Backup Daemons. Making the change and clicking *Save* is not sufficient. Ops Manager will continue to use the previous string until you restart the instances.

<i>&lt;hostname&gt;:&lt;port&gt;</i>	Enter a comma-separated list of the fully qualified domain names and port numbers for all <i>replica set</i> members for the Backup Database.
<i>MongoDB Auth Username</i> and <i>MongoDB Auth Password</i> <i>Encrypted Credentials</i>	Enter the user credentials if the database uses authentication.
<i>Use SSL</i>	Check this if the user credentials use the Ops Manager <i>credentialstool</i> . For more information, see <a href="#">Encrypt User Credentials</a> .
<i>Connection Options</i>	Check this if the database uses SSL. If you select this, you must configure SSL settings Ops Manager. See <a href="#">Ops Manager Configuration</a> . To add additional connection options, enter them using the MongoDB <a href="#">Connection String URI Format</a> . This field supports un-escaped values only.

## Next Steps

Once you have installed the Ops Manager web application to the Ops Manager server, you must next do the following:

1. Install Ops Manager agents on the servers that run your MongoDB deployments. You can install agents on servers running existing MongoDB deployments or on servers on which you will create new MongoDB deployments. Servers that run your MongoDB deployments must meet the requirements in the [MongoDB Production Notes](#) in the MongoDB Manual.

To install agents, see [Provision Servers](#).

### See also:

[Install the Automation Agent](#), [Install Monitoring Agent](#), [Install Backup Agent](#)

2. After you install agents, deploy MongoDB to your servers to test connections. If you use Ops Manager Automation, you can deploy MongoDB through the Ops Manager interface. For example, see [Deploy a Replica Set](#).

## Install Ops Manager on Microsoft Windows

### On this page

- [Overview](#)
- [Prerequisites](#)
- [Single Instance Install Procedure](#)
- [Multiple Instance Install Procedure](#)
- [Backup Configuration Procedure](#)
- [Next Steps](#)

### Overview

This tutorial describes how to install Ops Manager using a Microsoft Windows Installer package (.msi). If you are instead upgrading an existing deployment, please see [Upgrade Ops Manager](#).

## Prerequisites

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

Before you install Ops Manager, you must:

1. Plan your configuration. See *Installation Checklist*.
2. Deploy servers that meet the *Ops Manager Hardware and Software Requirements*.

**Warning:** Failure to configure servers according to the *Ops Manager Hardware and Software Requirements*, including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

3. *Install the Ops Manager Application Database and optional Backup Database.* The databases require dedicated MongoDB instances. Do **not** use MongoDB installations that store other data. The Backup Database is required only if you will use the Backup feature.

The Ops Manager Application and Backup Daemon must authenticate to the backing databases as a MongoDB user with appropriate access. See [mongo.mongoUri](#) for more information.

---

**Note:** Ops Manager cannot deploy its own backing databases. You must deploy those databases manually.

---

## Single Instance Install Procedure

To install Ops Manager:

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

1. In a browser, go to <http://www.mongodb.com> and click on the [Download](#) button.
2. Complete the form.
3. On the *MongoDB Download Center* page, click on the *Ops Manager* tab.
4. Select Windows Server 2008 R2+ from the *Platforms* drop-down menu.
5. Select MSI from the *Packages* drop-down menu.
6. Click *Download*.

---

**Note:** The downloaded package is named `mongodb-mms-<version>.msi`, where `<version>` is the version number.

---

### Step 2: Install the Ops Manager package on each server being used for Ops Manager.

1. Double click the MSI package.
2. Follow the instructions in the *Setup Wizard*.
3. During setup, the *Configuration/Log Folder* step prompts you to specify a folder where the configuration and log files will be stored.

The installation restricts access to the folder to users with the *Administrator* access privileges only.

**Step 3: Install the Ops Manager package.** On a server that is to run the Ops Manager, open <configLogPath>\Server\Config\conf-mms.properties with *Administrator* access privileges.

<configLogPath> is C:\MMSData by default.

Configure the following setting to provide the connection string Ops Manager uses to connect to the database:

- mongo.mongoUri

If you will configure Ops Manager to use the Ops Manager Application Database over SSL, configure the following *SSL settings*:

- mongo.ssl
- mongodb.ssl.CAFile
- mongodb.ssl.PEMKeyFile
- mongodb.ssl.PEMKeyFilePassword

Ops Manager also uses these settings for SSL connections to Backup Databases

If you will configure Ops Manager to use Kerberos to manage access to the Ops Manager Application Database, configure the following *Kerberos settings*:

- jvm.java.security.krb5.kdc
- jvm.java.security.krb5.realm
- mms.kerberos.principal
- mms.kerberos.keyTab

#### **Step 4: Start the Ops Manager instances.**

1. To start the service, click the *Start* button.
2. Click *Administrative Tools*.
3. Click *Services*.
4. In the *Services* list, right-click the *MongoDB Ops Manager HTTP Service* and select *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. See *Manage Ops Manager Users*.

#### **Multiple Instance Install Procedure**

To configure a multi-host Ops Manager:

**Step 1: Complete Installation Procedures Steps 1 and 2 on each Ops Manager server.** Complete the first two steps under Single Instance Install Procedure on each server that will host Ops Manager.

**Step 2: Copy the `gen.key` file from the current server to the other servers.** Ops Manager requires an identical `gen.key` file be stored on both servers running Ops Manager and uses the file to encrypt data at rest in the Ops Manager Application Database and Backup Database.

**Before** starting Ops Manager on any other server, you must copy the `gen.key` file from the `<installPath>\MMSData\Secrets` directory on the first Ops Manager server to the same directory on all other Ops Manager servers.

**Step 3: If you will run multiple Ops Manager Applications behind a load balancer, configure and start the applications.** For each Ops Manager instance, repeat the **Step 3** to configure the connection to the Ops Manager Application Database and **Step 4** to start the application.

When you run behind a load balancer, do not allow clients to connect directly to any Ops Manager Application server. The load balancer must not return cached content.

For more information on running multiple applications behind a load balancer, see [Configure a Highly Available Ops Manager Application](#).

### Backup Configuration Procedure

If you will run Ops Manager Backup, configure your Ops Manager installation to backup your MongoDB deployments:

**Step 1: Configure Ops Manager to manage backups.** To enable Ops Manager to configure and manage MongoDB backups:

1. To start the service, click the *Start* button.
2. Click *Control Panel*.
3. Click *Administrative Tools*.
4. Click *Services*.
5. Right-click on the *MongoDB Backup Daemon Service* and select *Start*.

### Step 2: Configure the Backup Daemon and Backup Storage.

1. On each Ops Manager server that will be activated as a Backup Daemon, create the directory that will store the *head databases*. The directory must be:
  - a dedicated disk partition that is not be used for any other purpose.
  - sized appropriately according to the [Ops Manager Hardware and Software Requirements](#).
  - writable by any user with *SYSTEM* access privileges.
2. If you will store snapshots on a filesystem instead of the Backup Database, create the directory that will store snapshots.
3. Open Ops Manager and make sure you are logged in as the user you registered when installing Ops Manager. This user is the *global owner*.
4. Click the *Admin* link at the top right of the page.
5. Click the *Backup* tab.
6. Follow the prompts to configure the Backup Daemon and Backup Storage. Ops Manager walks you through configuration of the daemon and snapshot storage.

After you select how to store snapshots, you are prompted to configure the connection string to the Backup Database. If you use filesystem storage for your snapshots, the Backup Database is used only for the *oplog store*.

<i>&lt;hostname&gt;:&lt;port&gt;</i>	Enter a comma-separated list of the fully qualified domain names and port numbers for all <i>replica set</i> members for the Backup Database.
<i>MongoDB Auth Username</i> and <i>MongoDB Auth Password</i>	Enter the user credentials if the database uses authentication.
<i>Encrypted Credentials</i>	Check this if the user credentials use the Ops Manager <i>credentialsstool</i> . See <i>Encrypt User Credentials</i> .
<i>Use SSL</i>	Check this if the database uses SSL. If you check this, you must configure SSL settings Ops Manager. See <i>Ops Manager Configuration</i> .
<i>Connection Options</i>	To add additional connection options, enter them using the MongoDB <i>Connection String URI Format</i> . This field supports un-escaped values only.

## Next Steps

*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 <http://docs.opsmanager.mongodb.com/tutorial/test-new-deployment>. The tutorial creates a replica set on Linux servers, populates the replica set with test data, registers a user, and installs the Monitoring and Backup Agents on a client machine to monitor the test replica set.

---

## 2.5 Advanced Configuration Options

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

**Deploy Highly Available Application** Outlines the process for achieving a highly available Ops Manager deployment.

**Deploy Highly Available Backups** Make the Backup system highly available.

**Assign Snapshot Stores to Specific Data Centers** Assign snapshot stores to specific data centers.

**Pass Outgoing Traffic through HTTP Proxy** Configure Ops Manager to pass all outgoing requests through an HTTP or HTTPS proxy.

### Configure Local Mode if Servers Have No Internet Access

#### On this page

- Overview
- Required Access
- Prerequisites
- Configure Local Mode
- Start Ops Manager When All Versions Are Not Downloaded

## Overview

If the servers that run your Automation Agents or Backup Daemons do not have internet access, you must configure Ops Manager to run in Local Mode, and you must manually provide the MongoDB binaries and [version manifest](#). As MongoDB Inc. releases new versions, you must update the binaries and manifest.

The Automation Agents require MongoDB binaries to install MongoDB on new deployments and to change MongoDB versions on existing ones. The Backup Daemons require the binaries to run the correct versions of MongoDB for the databases being backed up.

In a default configuration, the agents and daemons access the binaries over the internet from MongoDB Inc. The [Version Manifest Source](#) configuration setting determines whether Ops Manager runs in Internet Mode or Local Mode.

**Versions Directory** The versions directory on the Ops Manager server(s) holds .tgz archive files of the MongoDB binaries that the agents and daemons require in order to manage and back up your deployments. You specify this directory in the [Versions Directory](#) setting.

If you run in Local Mode, you must manually download the binaries from MongoDB Inc. and place them in this directory. The “mongodb-mms” user must have permission to read the archives in this directory.

**Version Manifest** The MongoDB version manifest makes Ops Manager aware of all released MongoDB versions. In Local Mode, the Automation Agents and Backup Daemons can deploy versions for which archives exist in the [versions directory](#). For a given Ops Manager *group*, the version must also be selected in the group’s [Version Manager](#).

In Internet Mode, Ops Manager automatically updates the version manifest from MongoDB, Inc. In Local Mode, you must update the version manifest as MongoDB releases new versions.

## Required Access

To configure Local Mode, you must have [Global Owner](#) access to Ops Manager.

## Prerequisites

Before configuring Ops Manager to run in Local Mode, you must meet the prerequisites described here.

**Determine Which Binaries to Store** The [versions directory](#) requires archives of following versions of MongoDB:

- Versions that are used by existing deployments that you will import into Ops Manager.
- Versions you will use to create new deployments through Ops Manager Automation.
- Versions you will use during an intermediary step in an upgrade.

---

### Example

If you will import an existing deployment of MongoDB 2.6 Community and upgrade it 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.

These are required archives for specific versions:

Edition	Version	Archive
Community	2.6+, 2.4+	Linux archive at <a href="http://www.mongodb.org/downloads">http://www.mongodb.org/downloads</a> .
Community	3.2+, 3.0+	Platform-specific archive available from <a href="http://www.mongodb.org/downloads">http://www.mongodb.org/downloads</a> .
Enterprise	3.2+, 3.0+, 2.6+, 2.4+	Platform-specific archive available from <a href="http://mongodb.com/download">http://mongodb.com/download</a> .

**Warning:** Go to *Deployment* and then *Version Manager*. Uncheck every version you do not plan to deploy before enabling Local Mode. If you leave unused versions checked and do not download their associated binaries, Ops Manager fails a pre-flight check when it starts.

If you did not uncheck unused versions or download the associated binaries, you can take Ops Manager out of Local Mode temporarily to pass the Ops Manager pre-flight check. See [Start Ops Manager When All Versions Are Not Downloaded](#) for the procedure.

**Install Enterprise Dependencies (MongoDB Enterprise Only)** If you will run *MongoDB Enterprise* and use Linux servers, then you must manually install a set of dependencies to each server **before installing MongoDB**. The MongoDB manual provides the appropriate command to install the dependencies. See the link for the server's operating system:

- *Red Hat*
- *Ubuntu*
- *Debian*
- *SUSE*
- *Amazon AMI*

## Configure Local Mode

### Step 1: Configure Local Mode.

1. In Ops Manager, click *Admin* in the upper right corner to open system administration.
2. Click the *General* tab.
3. Click *Ops Manager Config*.
4. Click the *Miscellaneous* button at the top of the page.
5. Set the *Version Manifest Source* setting to *Local*.
6. Note the directory specified in *Versions Directory*. This is the directory on your Ops Manager servers that stores the MongoDB binaries.

**Step 2: Populate all the Ops Manager servers with the .tgz files for the MongoDB binaries.** On all your Ops Manager servers, including those with enabled Backup Daemons:

1. Navigate to the versions directory, which is the directory specified in the *Versions Directory* setting.
2. Populate the versions directory 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 a MongoDB 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 3: Ensure that the `mongodb-mms` user has appropriate access to the MongoDB binaries.** At minimum, the `mongodb-mms` user must be able to **read** the .tgz files placed in the directory specified in the [Versions Directory](#) setting. If a server with no internet connection has an enabled Backup Daemon, the `mongodb-mms` user must also be able to **write** to this directory so that the daemon can extract the .tgz files.

For example, on a Linux server that runs the Backup Daemon, the directory's long output might look like this:

```
-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 51163601 May 22 10:05 mongodb-linux-x86_64-amazon-3.0.3.tgz
-rw-r----- 1 mongodb-mms staff 50972165 May 22 10:06 mongodb-linux-x86_64-suse11-3.0.3.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
```

If the `mongodb-mms` does not have the correct permissions, you must provide them.

For example, on a Linux platform, you could use the Linux `chown` command to change ownership for **all** files in the directory to `mongodb-mms`:

```
sudo chown mongodb-mms:mongodb-mms <path-to-the-versions-directory>
```

#### **Step 4: Copy the version manifest to Ops Manager.**

1. In Ops Manager, click *Admin* in the upper right corner to open system administration.
2. Click the *General* tab.
3. Click *Version Manifest*.
4. Click the *Update the MongoDB Version Manifest* button.

If you cannot access the Internet, you must copy the manifest using a system that can, and you must then paste the manifest here. Copy the manifest from [https://opsmanager.mongodb.com/static/version\\_manifest/2.0.json](https://opsmanager.mongodb.com/static/version_manifest/2.0.json).

#### **Step 5: For each group, specify which versions are available for download by Automation Agents.**

1. In Ops Manager, click *Ops Manager* in the upper left to exit system administration.
2. Click the *Group* link at the top of the page and select the desired group.
3. 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*.
5. Click *Confirm & Deploy*.

## Start Ops Manager When All Versions Are Not Downloaded

Ops Manager sets the default source of MongoDB binaries to be over the Internet from MongoDB. This enables you to install all versions of MongoDB. This allows Ops Manager to deploy any supported version of MongoDB.

When you set Ops Manager to *Local Mode*, Ops Manager checks to see if the binaries for all of the versions you set in the *Version Manager* can be found in the [Versions Directory](#) on the Ops Manager server. If these binaries are not found, Ops Manager fails this “pre-flight” check and stops.

To allow you to start Ops Manager, you can either install all of the binaries for every version of MongoDB or change the configuration of Ops Manager temporarily to allow it to start and let you change the versions of MongoDB your Ops Manager server supports.

**Change Configuration File to Allow Ops Manager to Start** To change the Ops Manager configuration temporarily:

1. Login to the Ops Manager server.
2. Open the Ops Manager configuration properties file.

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

---

**Note:** This path is for *RPM* and *DEB* package installs. The *Linux archive* and *Windows* binaries install in different paths.

---

3. Add a line changing the source of MongoDB versions to be MongoDB, Inc. via the Internet.

Remove this line:

```
automation.versions.source=mongodb
```

4. Save and close the file.
5. If you run Ops Manager with high availability, you must repeat steps 1-4 on each Ops Manager server.
6. Start the Ops Manager service on each Ops Manager server.

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

7. Change the versions of MongoDB that Automation Agents can install. See [how to set the MongoDB versions that automation can install](#) for details.

**Warning:** You need to change the MongoDB version list for every group in Ops Manager.

8. Once you have changed which versions you want to support, continue to the next procedure.

**Revert the Configuration File to Return to Local Mode** If these steps are not completed, Ops Manager remains in Internet mode and displays a notice in the configuration page notifying you that the *Local Mode* setting in your `conf-mms.properties` overrides this setting.

To revert the Ops Manager configuration change:

1. Stop the Ops Manager service on each Ops Manager server.

```
sudo service mongodb-mms stops
```

2. Login to the Ops Manager server.

3. Open the Ops Manager configuration properties file.

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

4. Delete this line. Removing it reverts the change you made in the previous procedure and re-enables *Local Mode*.

```
automation.versions.source=mongodb
```

5. Save and close the file.

6. If you run Ops Manager with high availability, you must repeat steps 2-5 on each Ops Manager server.

7. Start the Ops Manager service on each Ops Manager server.

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

## Configure a Highly Available Ops Manager Application

### On this page

- Overview
- Prerequisites
- Procedure
- Additional Information

### Overview

The Ops Manager Application provides high availability through use of multiple Ops Manager Application servers behind a load balancer and through use of a *replica set* to host the *Ops Manager Application Database*.

**Multiple Ops Manager Application Servers** The Ops Manager Application's *components* are stateless between requests. Any Ops Manager Application server can handle requests as long as all the servers read from the same Ops Manager Application Database. 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. The load balancer must not return cached content.

In Ops Manager, set the [URL to Access Ops Manager](#) property to the load balancer URL. Set the [Load Balancer Remote IP Header](#) property to the HTTP header field the load balancer uses to identify the originating client's IP address (for example, X-Forwarded-For). The Ops Manager Application uses the client's IP address for auditing, logging, and white listing for the API.

**Replica Set for the Ops Manager Application Database** Deploy a *replica set* rather than a standalone to host the *Ops Manager Application Database*. Replica sets have automatic failover if the *primary* becomes unavailable.

If the replica set has 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](#).

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.

To deploy the replica set that hosts the Ops Manager Application Database, see [backing MongoDB instance](#).

**The gen.key File** `gen.key` file is a 24-byte binary file used to encrypt and decrypt Ops Manager's backing databases and user credentials. An identical `gen.key` file must be stored on every server that is part of a highly available Ops Manager deployment.

The `gen.key` file can be generated automatically or manually.

**To have Ops Manager generate the file:** Start *one* Ops Manager server. Ops Manager will create a `gen.key` file if none exists.

**To create the file manually:** Generate a 24-byte binary file. For example, the following creates the `gen.key` file using `openssl`:

```
openssl rand 24 > /<keyPath>/gen.key
```

Protect the `gen.key` file like any sensitive file. Change the owner to the user running Ops Manager and set the file permission to read and write for the owner only.

Once you have the `gen.key` file (either created automatically or manually), *before* starting the other Ops Manager servers, copy the file to the appropriate directory on the current server and to the appropriate directory on the other Ops Manager servers:

- `/etc/mongodb-mms/` for RPM or Ubuntu installations
- `<installPath>/.mongodb-mms/` for an archive (`.tar`) file installations
- `<installPath>\MMSData\Secrets` for Microsoft Windows Server installations

---

#### Important:

- Any shared storage resource that stores the `gen.key` file should be configured for high availability so as not to introduce a potential single point of failure.
  - Any Ops Manager server that does not have the `gen.key` file installed cannot connect to the backing databases and become part of an HA Ops Manager instance.
  - Once you have generated the `gen.key` for your Ops Manager instance on the first Ops Manager server, back up the `gen.key` file to a secure location.
- 

#### Prerequisites

Deploy the replica set that hosts the *Ops Manager Application Database*. To deploy a replica set, see Deploy a Replica Set in the MongoDB manual.

#### Procedure

The following procedure assumes you will let one of the Ops Manager Applications create the `gen.key`. If you instead create your own `gen.key`, distribute it to the servers before starting any of the Ops Manager Applications.

---

**Important:** The load balancer placed in front of the Ops Manager Application servers must not return cached content. The load balancer must have caching turned off.

---

To configure multiple Ops Manager Applications 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: Configure Ops Manager to use the load balancer.**

1. In Ops Manager, click *Admin*, then the *General* tab, and then *Ops Manager Config*.
2. Configure the [URL to Access Ops Manager](#) property to point to the load balancer URL.
3. Set the [Load Balancer Remote IP Header](#) property to the name of the HTTP header field the load balancer uses to identify the client's IP address.

**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>
```

**Step 4: Start one of the Ops Manager Applications.** For example, if you installed the Ops Manager Application with an `rpm` or `deb` package, issue the following:

```
service mongodb-mms start
```

**Step 5: Copy the `gen.key` file.** The `gen.key` file is located in `/etc/mongodb-mms/` for installations from a package manager and in  `${HOME} / .mongodb-mms/` for installations from an archive.

Copy the `gen.key` file from the running Ops Manager Application's server to the appropriate directory on the other Ops Manager Application servers.

## **Step 6: Start the remaining Ops Manager Applications.**

### **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](#)
- [Additional Information](#)

### **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. This scales by increasing the available storage for the *head databases*. This does not increase the available space for snapshot storage.

With multiple daemons, Ops Manager binds each backed-up replica set or shard to a particular Backup Daemon. For example, if you run two 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 Daemon to another Backup Daemon](#) for a description of this process.

Ops Manager reconstructs the data using a snapshot and the oplog from the *Backup Data Storage*.

The Backup Daemon is installed with Ops Manager package but must be specifically activated to run. For more information, see [Install Ops Manager](#) and 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 Daemon to another Backup Daemon](#).

For information on making the Ops Manager Application highly available, see [Configure a Highly Available Ops Manager Application](#).

## Assign Snapshot Stores to Specific Data Centers

### On this page

- Overview
- Prerequisites
- Procedures

## Overview

Additional Snapshot Stores can be added to meet storage requirements. These additional stores can be deployed in the same data center as the first store. There are times where these stores need to be deployed to other data centers due to either network performance issues or regulatory requirements.

You can bind specific Ops Manager *groups* to specific snapshot stores and particular data centers. This assigns the backups of particular MongoDB deployments to specific data centers.

This tutorial sets up two snapshot stores, one in each of two separate data centers, and attaches a separate group to each.

## Prerequisites

- Configure the two Ops Manager Application instances to share a single dedicated *Ops Manager Application Database*.
  - The members of the Ops Manager Application Database replica set can be put in each data center.
- Configure each Backup Agent to use the URL for its local Ops Manager Application.
  - Each Ops Manager Application can use a different hostname or a split-horizon DNS to point each agent to its local Ops Manager Application.

---

**Note:** 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 data center must host its own *Backup Blockstore Database* or File System Store, *Ops Manager Application* and *Backup Agent*.

Each server must meet the cumulative hardware and software requirements for the components it runs. See [Ops Manager Hardware and Software Requirements](#).

All servers running Backup and Ops Manager Application Databases use MongoDB. They must meet the configuration requirements in the [MongoDB Production Notes](#).

**Install MongoDB** Install MongoDB on the servers that host the:

- Ops Manager Application Database
- 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 Blockstore Databases, see [Deploy a Replica Set](#) in the MongoDB manual.

**Install and Start the Ops Manager Application** Install the Ops Manager Application in each data center.

---

**Note:** See [Install Ops Manager](#) for instructions for your operating system.

---

**Important:** Each set of installation instructions for each operating system covers how to create a multiple instances of an Ops Manager Application. These activities need to be completed before binding groups to the backup resources.

---

## Bind Groups to Backup Resources

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

**Step 2: Create a new Ops Manager group for Data Center #1.**

1. Click the *Settings* tab.
2. Click the *My Groups* page.
3. Click *Add Group*.
4. Enter the group name.
5. Click *Create Group*.

**Step 3: Create a second Ops Manager group for the Data Center #2.**

**Step 4: Click the *Admin* link at the upper right of the page to open the Admin interface.**

**Step 5: Configure backup resources.**

1. Click the *Backup* tab.
2. Click the *Daemons* page and check that there are two daemons listed.
3. Click the *Snapshot Storage* page.
4. Create a *file system store* or a *blockstore* using network storage or a MongoDB database in Data Center #2.
5. Click *Save*.
6. Click the *Oplog Storage* page.
7. Add an oplog store using a MongoDB database in Data Center #2.
8. Click *Save*.

**Step 6: Assign resources to the data centers.**

1. Click the *General* tab.
2. Click the *Groups* page.
3. Select the name of the group assigned to Data Center 1.
4. Click the *View* link to the right of *Backup Configuration*.
5. Select the local options for Group 1 / Data Center 1.

Menu	Option
<i>Backup Daemons</i>	Choose <i>Select Backup Daemons</i> then check the daemon that is in Data Center 1.
<i>Oplog Stores</i>	Choose <i>Select Oplog Stores</i> then check the oplog store that is in Data Center 1.
<i>Snapshot Stores</i>	Choose <i>Select Snapshot Stores</i> then check the snapshot store that is in Data Center 1.

6. Repeat steps a to e for Group 2.

**Step 7: Install agents.** If you are using Automation:

1. *Install the Automation Agent* for the group in Data Center 1 on each server in Data Center 1.
2. Install the Automation Agent for Data Center 2 on each server in Data Center 2.
3. The Automation Agent installs the Monitoring and Backup agents as needed.

If you are not using Automation:

1. Select the group assigned to Data Center 1 from the drop-down menu in the top navigation bar.
2. Click on the *Settings* tab.
3. Click on the *Agents* page.
4. Download and install the Monitoring and Backup agents for the group assigned to Data Center 1.
5. Select the group in Data Center 2 from the drop-down menu in the top navigation bar.
6. Download and install its *Monitoring* and *Backup* agents.

## Configure Ops Manager to Pass Outgoing Traffic through an HTTP or HTTPS Proxy

### On this page

- Overview
- Procedure

### Overview

Ops Manager can pass all outgoing requests through an HTTP or HTTPS proxy. This lets Ops Manager deployments without direct access to external resources access outside notification services and the MongoDB *version manifest*.

### Procedure

**Step 1: Stop Ops Manager** 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 configure the proxy settings.** Open *Ops Manager Configuration* with root privileges. Set the `http.proxy.host` and `http.proxy.port` settings to the hostname and port of the HTTP or HTTPS proxy.

If the proxy requires authentication, use `http.proxy.username` and `http.proxy.password` to specify the credentials.

**Step 3: Start Ops Manager.** 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
```

## 2.6 Upgrade Ops Manager

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

**Upgrade with DEB Package** Upgrade Ops Manager on Debian and Ubuntu systems.

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

**Upgrade on Windows** Upgrade Ops Manager on Windows systems.

### Upgrade Ops Manager with an `rpm` Package

#### On this page

- Overview
- Considerations
- Prerequisite
- Upgrade Path
- Procedure

#### Overview

This tutorial describes how to upgrade an existing Ops Manager installation using an `rpm` package. Ops Manager supports direct upgrades from version 1.6 and later.

#### Considerations

**Backup Database** There are data migrations that touch the various backup data stores that make up the *Backup Data Storage*. The data stores must all be online when you upgrade. Any data stores that are no longer in use should be deleted through the Ops Manager UI before upgrading.

**Backup Daemon** Beginning with Ops Manager 2.0, there is no separate Backup Daemon package. The Ops Manager package also installs the Backup Daemon. When started, the Ops Manager package automatically starts two services: the Ops Manager Application and the Backup Daemon. You choose on which servers to “activate” the Backup Daemon. The daemon always runs, but it performs no operations unless activated.

After upgrade, a server that runs **only** the Ops Manager Application continues to do so but now also runs a “dormant” Backup Daemon service. The Backup Daemon remains dormant as long as you do not activate it.

A server that runs **only** a Backup Daemon runs Ops Manager with an “activated” Backup Daemon and a “dormant” Ops Manager Application. The Ops Manager Application remains dormant as long as you do not direct HTTP traffic to it.

**Backup HTTP Service** Beginning with Ops Manager 2.0, there is no Backup HTTP Service on port 8081. Any Backup Agents that are managed by Automation will be automatically updated to use the new port, 8080. For Backup Agents that were installed **manually**, you must edit the agent's configuration file, as described in the procedure below. You must have access to the servers running any manually installed Backup Agents.

**Warning:** You must configure the new port for any **manually installed** Backup Agents, or the agents will have no access to Ops Manager.

**Agent Updates** Do not update the agents before upgrade. If you use Automation, Ops Manager prompts you to update the agents after you upgrade. Follow the prompts to update the agents through the Ops Manager UI. Do *not* update the agents manually.

**conf-mms.properties** Beginning in 2.0, Ops Manager stores global configuration settings in the Ops Manager Application Database and stores only local settings in the Ops Manager server's `conf-mms.properties` file. The upgrade procedure uses the existing `conf-mms.properties` file to connect to the Ops Manager Application Database before replacing the existing file with the new, smaller 2.0 file.

**Restore properties** The following properties no longer apply and are replaced by settings specified when initiating a restore:

- `mms.backup.restore.linkExpirationHours`
- `mms.backup.restore.linkUnlimitedUses`

**mms.conf** If you have modified the `mms.conf` file in your current installation (which is not typical), back up the file. You must use the new `mms.conf` file installed by the upgrade. You can redo any modifications once the upgrade is complete.

## Prerequisite

**Hardware and Software Requirements** Your servers must meet the *Ops Manager Hardware and Software Requirements*.

**Warning:** Failure to configure servers according to the *Ops Manager Hardware and Software Requirements*, including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

**Administrator Privileges** You must have administrator privileges on the servers on which you perform the upgrade.

**Download Link** 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>.

## Upgrade Path

The version of your existing Ops Manager installation determines your upgrade path. The following table lists upgrade paths per version:

Existing Version	Upgrade Path
1.6 or later	Use this procedure to upgrade directly to the latest version.
1.5	<ol style="list-style-type: none"> <li>1. Upgrade to 1.8 using the <a href="#">upgrade procedure in the v1.8 documentation</a>.</li> <li>2. Use this procedure to upgrade to the latest version.</li> </ol>
1.4 or earlier	<ol style="list-style-type: none"> <li>1. Upgrade to 1.6 using the <a href="#">upgrade procedure in the v1.6 documentation</a>.</li> <li>2. Upgrade to 1.8 using the <a href="#">upgrade procedure in the v1.8 documentation</a>.</li> <li>3. Use this procedure to upgrade to the latest version.</li> </ol>

There are no supported downgrade paths for Ops Manager.

---

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

---

## Procedure

To upgrade Ops Manager:

**Step 1: Take a full backup of the Ops Manager backing databases.**

**Step 2: Stop all existing Ops Manager services.**

1. Shut down the Ops Manager Application:

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

2. If you are upgrading from a pre-2.0 Ops Manager Application, and you are also running a Backup Daemon on the machine, shut down the Backup Daemon service:

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

**Step 3: Uninstall the Backup Daemon package. (Pre-2.0 only)** Do the following on **every server** where the Backup Daemon is installed:

1. Create a backup copy of each `conf-daemon.properties` file to preserve the existing configuration.
2. Uninstall the Backup Daemon package.

This config file is located at `/opt/mongodb/mms-backup-daemon/conf/conf-daemon.properties`.

**Step 4: Download the latest version of the Ops Manager package.**

1. In a browser, go to <http://www.mongodb.com> and click on the [Download](#) button.
2. Complete the form.
3. On the *MongoDB Download Center* page, click on the *Ops Manager* tab.

4. Select RedHat 5+ / CentOS 5+ / SUSE 11+ / Amazon Linux from the *Platforms* drop-down menu.
5. Select RPM from the *Packages* drop-down menu.
6. Click *Download*.

---

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

---

#### **Step 5: Install the Ops Manager package on each server being used for Ops Manager services, including backup.**

---

**Important:** The Backup Daemon is now a function of Ops Manager and not a separate service. This means that every server that previously only had a Backup Daemon service installed will need to install the full Ops Manager server to provide backup capabilities.

---

Install the `.rpm` package on each Ops Manager Application server by issuing the following command, where `<version>` is the version of the `.rpm` package:

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

The installer preserves the existing `conf-mms.properties` configuration file and names the new configuration file `conf-mms.properties.rpmnew`. Both are located in `/opt/mongodb/mms/conf/`.

#### **Step 6: Copy the connection string to the new Ops Manager instances.**

1. Copy the `mongo.mongoUri` string from the `conf-mms.properties` file of the server that was upgraded to the current version of Ops Manager.
2. Replace the `mongo.mongoUri` string in each `conf-mms.properties` file in each servers which previously only hosted a Backup Daemon.

---

**Note:** The former Backup Daemon only servers used a different config file, but the upgrade to Ops Manager on those servers created a `conf-mms.properties` file. For these new instances to properly connect to the databases, this string must be replaced.

---

#### **Step 7: Only if you installed to RHEL 7.1 or 7.2, replace symlinks with scripts.** As a result of [RHEL Bug 1285492](#), the version of `systemd` found in RHEL 7.1 and 7.2 Ops Manager will not start automatically on boot up.

To work around this:

1. Replace the symbolic links to the scripts with the scripts themselves. Copy the scripts from `/opt/mongodb/mms/bin/mongodb-mms` to `/etc/init.d`.
2. In the copied scripts, change the line:

```
ABS_PATH=$( resolvepath $0 )"
```

to:

```
SCRIPTPATH=/opt/mongodb/mms/bin/mongodb-mms
ABS_PATH=$( resolvepath $SCRIPTPATH )"
```

**Step 8: Start Ops Manager on every server.** Issue the following command:

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

**Step 9: Log into Ops Manager as a user with the Global Owner role.** Ops Manager displays a setup wizard for entering the Ops Manager configuration in the Application Database.

**Step 10: Complete the Ops Manager configuration setup.** The setup wizard walks you through several pages to configure the Application Database. Refer to your existing `conf-mms.properties` file as needed.

**See also:**

*Ops Manager Configuration* for the mapping of `conf-mms.properties` parameters to setup wizard fields.

---

**Important:** Make sure you click *Save* on each tab, even if you do not make any changes.

---

**Step 11: For any Backup Agents that were installed manually, update their configurations to use port 8080.** Beginning with Ops Manager 2.0, Backup Agents access Ops Manager through port 8080. The Automation Agent automatically updates the ports on managed Backup Agents. For any Backup Agents that were installed **manually**, you must edit the agent's *configuration file* and set the `mothership` property to access Ops Manager on port 8080.

## Upgrade Ops Manager with a deb Package

### On this page

- [Overview](#)
- [Considerations](#)
- [Prerequisite](#)
- [Upgrade Path](#)
- [Procedure](#)

### Overview

This tutorial describes how to upgrade an existing Ops Manager installation using a `deb` package. Ops Manager supports direct upgrades from version 1.6 and later.

### Considerations

**Backup Database** There are data migrations that touch the various backup data stores that make up the *Backup Data Storage*. The data stores must all be online when you upgrade. Any data stores that are no longer in use should be deleted through the Ops Manager UI before upgrading.

**Backup Daemon** Beginning with Ops Manager 2.0, there is no separate Backup Daemon package. The Ops Manager package also installs the Backup Daemon. When started, the Ops Manager package automatically starts two services: the Ops Manager Application and the Backup Daemon. You choose on which servers to “activate” the Backup Daemon. The daemon always runs, but it performs no operations unless activated.

After upgrade, a server that runs **only** the Ops Manager Application continues to do so but now also runs a “dormant” Backup Daemon service. The Backup Daemon remains dormant as long as you do not activate it.

A server that runs **only** a Backup Daemon runs Ops Manager with an “activated” Backup Daemon and a “dormant” Ops Manager Application. The Ops Manager Application remains dormant as long as you do not direct HTTP traffic to it.

**Backup HTTP Service** Beginning with Ops Manager 2.0, there is no Backup HTTP Service on port 8081. Any Backup Agents that are managed by Automation will be automatically updated to use the new port, 8080. For Backup Agents that were installed **manually**, you must edit the agent’s configuration file, as described in the procedure below. You must have access to the servers running any manually installed Backup Agents.

**Warning:** You must configure the new port for any **manually installed** Backup Agents, or the agents will have no access to Ops Manager.

**Agent Updates** Do not update the agents before upgrade. If you use Automation, Ops Manager prompts you to update the agents after you upgrade. Follow the prompts to update the agents through the Ops Manager UI. Do *not* update the agents manually.

**conf-mms.properties** Beginning in 2.0, Ops Manager stores global configuration settings in the Ops Manager Application Database and stores only local settings in the Ops Manager server’s `conf-mms.properties` file. The upgrade procedure uses the existing `conf-mms.properties` file to connect to the Ops Manager Application Database before replacing the existing file with the new, smaller 2.0 file.

**Restore properties** The following properties no longer apply and are replaced by settings specified when initiating a restore:

- `mms.backup.restore.linkExpirationHours`
- `mms.backup.restore.linkUnlimitedUses`

**mms.conf** If you have modified the `mms.conf` file in your current installation (which is not typical), back up the file. You must use the new `mms.conf` file installed by the upgrade. You can redo any modifications once the upgrade is complete.

## Prerequisite

**Hardware and Software Requirements** Your servers must meet the *Ops Manager Hardware and Software Requirements*.

**Warning:** Failure to configure servers according to the *Ops Manager Hardware and Software Requirements*, including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

**Administrator Privileges** You must have administrator privileges on the servers on which you perform the upgrade.

**Download Link** 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>.

## Upgrade Path

The version of your existing Ops Manager installation determines your upgrade path. The following table lists upgrade paths per version:

Existing Version	Upgrade Path
1.6 or later	Use this procedure to upgrade directly to the latest version.
1.5	<ol style="list-style-type: none"> <li>1. Upgrade to 1.8 using the <a href="#">upgrade procedure in the v1.8 documentation</a>.</li> <li>2. Use this procedure to upgrade to the latest version.</li> </ol>
1.4 or earlier	<ol style="list-style-type: none"> <li>1. Upgrade to 1.6 using the <a href="#">upgrade procedure in the v1.6 documentation</a>.</li> <li>2. Upgrade to 1.8 using the <a href="#">upgrade procedure in the v1.8 documentation</a>.</li> <li>3. Use this procedure to upgrade to the latest version.</li> </ol>

There are no supported downgrade paths for Ops Manager.

---

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

---

## Procedure

To upgrade Ops Manager:

**Step 1: Take a full backup of the Ops Manager backing databases.**

**Step 2: Stop all existing Ops Manager services.**

1. Shut down the Ops Manager Application:

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

2. If you are upgrading from a pre-2.0 Ops Manager Application, and you are also running a Backup Daemon on the machine, shut down the Backup Daemon service:

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

**Step 3: Uninstall the Backup Daemon package. (Pre-2.0 only)** Do the following on **every server** where the Backup Daemon is installed:

1. Create a backup copy of each `conf-daemon.properties` file to preserve the existing configuration.
2. Uninstall the Backup Daemon package.

This config file is located at `/opt/mongodb/mms-backup-daemon/conf/conf-daemon.properties`.

**Step 4: Download the latest version of the Ops Manager package.**

1. In a browser, go to <http://www.mongodb.com> and click on the [Download](#) button.
2. Complete the form.
3. On the *MongoDB Download Center* page, click on the *Ops Manager* tab.
4. Select Ubuntu 12.04+ from the *Platforms* drop-down menu.

5. Select DEB from the *Packages* drop-down menu.
6. Click *Download*.

---

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

---

#### **Step 5: Install the Ops Manager package on each server being used for Ops Manager.**

1. Install the `.deb` package on each Ops Manager Application server and each Backup Daemon server by issuing the following command, where `<version>` is the version of the `.deb` package:  
`sudo dpkg -i mongodb-mms-<version>.x86_64.deb`
2. When prompted whether to overwrite the currently installed version of `conf-mms.properties`, type N, to keep the currently installed version.

The installer preserves the existing `conf-mms.properties` configuration file and names the new configuration file `conf-mms.properties.dpkg-dist`. Both are located in `/opt/mongodb/mms/conf/`.

#### **Step 6: Copy the connection string to the new Ops Manager instances.**

1. Copy the `mongo.mongoUri` string from the `conf-mms.properties` file of the server that was upgraded to the current version of Ops Manager.
2. Replace the `mongo.mongoUri` string in each `conf-mms.properties` file in each servers which previously only hosted a Backup Daemon.

---

**Note:** The former Backup Daemon only servers used a different config file, but the upgrade to Ops Manager on those servers created a `conf-mms.properties` file. For these new instances to properly connect to the databases, this string must be replaced.

---

#### **Step 7: Start Ops Manager on every server.** Issue the following command:

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

#### **Step 8: Log into Ops Manager as a user with the Global Owner role.** Ops Manager displays a setup wizard for entering the Ops Manager configuration in the Application Database.

**Step 9: Complete the Ops Manager configuration setup.** The setup wizard walks you through several pages to configure the Application Database. Refer to your existing `conf-mms.properties` file as needed.

**See also:**

[Ops Manager Configuration](#) for the mapping of `conf-mms.properties` parameters to setup wizard fields.

---

**Important:** Make sure you click *Save* on each tab, even if you do not make any changes.

---

#### **Step 10: Restart all Ops Manager services.**

**Step 11: For any Backup Agents that were installed manually, update their configurations to use port 8080.** Beginning with Ops Manager 2.0, Backup Agents access Ops Manager through port 8080. The Automation Agent automatically updates the ports on managed Backup Agents. For any Backup Agents that were installed **manually**, you must edit the agent's *configuration file* and set the *mothership* property to access Ops Manager on port 8080.

## Upgrade Ops Manager from a `tar.gz` or `zip` Archive

### On this page

- [Overview](#)
- [Considerations](#)
- [Prerequisite](#)
- [Upgrade Path](#)
- [Procedure](#)

### Overview

This tutorial describes how to upgrade an existing Ops Manager installation using a tar.gz or zip file. Ops Manager supports direct upgrades from version 1.6 and later.

### Considerations

**Backup Database** There are data migrations that touch the various backup data stores that make up the *Backup Data Storage*. The data stores must all be online when you upgrade. Any data stores that are no longer in use should be deleted through the Ops Manager UI before upgrading.

**Backup Daemon** Beginning with Ops Manager 2.0, there is no separate Backup Daemon package. The Ops Manager package also installs the Backup Daemon. When started, the Ops Manager package automatically starts two services: the Ops Manager Application and the Backup Daemon. You choose on which servers to “activate” the Backup Daemon. The daemon always runs, but it performs no operations unless activated.

After upgrade, a server that runs **only** the Ops Manager Application continues to do so but now also runs a “dormant” Backup Daemon service. The Backup Daemon remains dormant as long as you do not activate it.

A server that runs **only** a Backup Daemon runs Ops Manager with an “activated” Backup Daemon and a “dormant” Ops Manager Application. The Ops Manager Application remains dormant as long as you do not direct HTTP traffic to it.

**Backup HTTP Service** Beginning with Ops Manager 2.0, there is no Backup HTTP Service on port 8081. Any Backup Agents that are managed by Automation will be automatically updated to use the new port, 8080. For Backup Agents that were installed **manually**, you must edit the agent's configuration file, as described in the procedure below. You must have access to the servers running any manually installed Backup Agents.

**Warning:** You must configure the new port for any **manually installed** Backup Agents, or the agents will have no access to Ops Manager.

**Agent Updates** Do not update the agents before upgrade. If you use Automation, Ops Manager prompts you to update the agents after you upgrade. Follow the prompts to update the agents through the Ops Manager UI. Do *not* update the agents manually.

**conf-mms.properties** Beginning in 2.0, Ops Manager stores global configuration settings in the Ops Manager Application Database and stores only local settings in the Ops Manager server's `conf-mms.properties` file. The

upgrade procedure uses the existing `conf-mms.properties` file to connect to the Ops Manager Application Database before replacing the existing file with the new, smaller 2.0 file.

**Restore properties** The following properties no longer apply and are replaced by settings specified when initiating a restore:

- `mms.backup.restore.linkExpirationHours`
- `mms.backup.restore.linkUnlimitedUses`

**mms.conf** If you have modified the `mms.conf` file in your current installation (which is not typical), back up the file. You must use the new `mms.conf` file installed by the upgrade. You can redo any modifications once the upgrade is complete.

## Prerequisite

**Hardware and Software Requirements** Your servers must meet the *Ops Manager Hardware and Software Requirements*.

**Warning:** Failure to configure servers according to the *Ops Manager Hardware and Software Requirements*, including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

**Administrator Privileges** You must have administrator privileges on the servers on which you perform the upgrade.

**Download Link** 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>.

## Upgrade Path

The version of your existing Ops Manager installation determines your upgrade path. The following table lists upgrade paths per version:

Existing Version	Upgrade Path
1.6 or later	Use this procedure to upgrade directly to the latest version.
1.5	<ol style="list-style-type: none"><li>1. Upgrade to 1.8 using the <a href="#">upgrade procedure in the v1.8 documentation</a>.</li><li>2. Use this procedure to upgrade to the latest version.</li></ol>
1.4 or earlier	<ol style="list-style-type: none"><li>1. Upgrade to 1.6 using the <a href="#">upgrade procedure in the v1.6 documentation</a>.</li><li>2. Upgrade to 1.8 using the <a href="#">upgrade procedure in the v1.8 documentation</a>.</li><li>3. Use this procedure to upgrade to the latest version.</li></ol>

There are no supported downgrade paths for Ops Manager.

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

## Procedure

To upgrade Ops Manager:

**Step 1: Take a full backup of the Ops Manager backing databases.**

**Step 2: Stop all existing Ops Manager services.** Shut down the Ops Manager Application:

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

If you are upgrading from a pre-2.0 Ops Manager Application, and you are also running a Backup Daemon on the machine, shut down the Backup Daemon service:

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

**Step 3: On each Ops Manager server, back up your existing configuration files and logs to a directory other than the install directory.** The following example backs up the configuration files and logs to the server's home directory:

```
cp -a <install_dir>/conf ~/mms_conf.backup  
cp -a <install_dir>/logs ~/mms_logs.backup
```

---

**Important:** You will need to access the backed-up `<install_dir>/conf/conf-mms.properties` file later in this procedure.

---

**Step 4: Remove the installed Ops Manager Application and Backup Daemon (Pre-2.0 only) packages.** On servers where there is an existing Ops Manager Application or Backup Daemon, first remove the existing application or daemon. **Do not remove your backups of the configuration files.**

**Step 5: Download the latest version of the Ops Manager archive.**

1. In a browser, go to <http://www.mongodb.com> and click on the **Download** button.
2. Complete the form.
3. On the *MongoDB Download Center* page, click on the *Ops Manager* tab.
4. Select RedHat 5+ / CentOS 5+ / SUSE 11+ / Amazon Linux or Ubuntu 12.04+ from the *Platforms* drop-down menu.
5. Select TAR.GZ from the *Packages* drop-down menu.
6. Click *Download*.

---

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

---

**Step 6: Install the Ops Manager package on each server being used for Ops Manager.** Navigate to the directory to which to install Ops Manager. Extract the archive to that directory:

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

### **Step 7: Copy the connection string to the new Ops Manager instances.**

1. Copy the `mongo.mongoUri` string from the `conf-mms.properties` file of the server that was upgraded to the current version of Ops Manager.
2. Replace the `mongo.mongoUri` string in each `conf-mms.properties` file in each servers which previously only hosted a Backup Daemon.

---

**Note:** The former Backup Daemon only servers used a different config file, but the upgrade to Ops Manager on those servers created a `conf-mms.properties` file. For these new instances to properly connect to the databases, this string must be replaced.

---

### **Step 8: Start Ops Manager on every server.** Issue the following command:

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

### **Step 9: Log into Ops Manager as a user with the Global Owner role.** Ops Manager displays a setup wizard for entering the Ops Manager configuration in the Application Database.

**Step 10: Complete the Ops Manager configuration setup.** The setup wizard walks you through several pages to configure the Application Database. Refer to your existing `conf-mms.properties` file as needed.

#### **See also:**

*Ops Manager Configuration* for the mapping of `conf-mms.properties` parameters to setup wizard fields.

---

**Important:** Make sure you click *Save* on each tab, even if you do not make any changes.

---

### **Step 11: For any Backup Agents that were installed manually, update their configurations to use port 8080.**

Beginning with Ops Manager 2.0, Backup Agents access Ops Manager through port 8080. The Automation Agent automatically updates the ports on managed Backup Agents. For any Backup Agents that were installed **manually**, you must edit the agent's *configuration file* and set the `mothership` property to access Ops Manager on port 8080.

## **Upgrade Ops Manager on Microsoft Windows Server**

### **On this page**

- Overview
- Considerations
- Prerequisite
- Procedure

### **Overview**

This tutorial describes how to upgrade an existing Ops Manager installation on Microsoft Windows. Ops Manager supports direct upgrades from version 1.6 and later.

## Considerations

**Backup Database** There are data migrations that touch the various backup data stores that make up the *Backup Data Storage*. The data stores must all be online when you upgrade. Any data stores that are no longer in use should be deleted through the Ops Manager UI before upgrading.

**Backup Daemon** Beginning with Ops Manager 2.0, there is no separate Backup Daemon package. The Ops Manager package also installs the Backup Daemon. When started, the Ops Manager package automatically starts two services: the Ops Manager Application and the Backup Daemon. You choose on which servers to “activate” the Backup Daemon. The daemon always runs, but it performs no operations unless activated.

After upgrade, a server that runs **only** the Ops Manager Application continues to do so but now also runs a “dormant” Backup Daemon service. The Backup Daemon remains dormant as long as you do not activate it.

A server that runs **only** a Backup Daemon runs Ops Manager with an “activated” Backup Daemon and a “dormant” Ops Manager Application. The Ops Manager Application remains dormant as long as you do not direct HTTP traffic to it.

**Backup HTTP Service** Beginning with Ops Manager 2.0, there is no Backup HTTP Service on port 8081. Any Backup Agents that are managed by Automation will be automatically updated to use the new port, 8080. For Backup Agents that were installed **manually**, you must edit the agent’s configuration file, as described in the procedure below. You must have access to the servers running any manually installed Backup Agents.

**Warning:** You must configure the new port for any **manually installed** Backup Agents, or the agents will have no access to Ops Manager.

**Agent Updates** Do not update the agents before upgrade. If you use Automation, Ops Manager prompts you to update the agents after you upgrade. Follow the prompts to update the agents through the Ops Manager UI. Do *not* update the agents manually.

**conf-mms.properties** Beginning in 2.0, Ops Manager stores global configuration settings in the Ops Manager Application Database and stores only local settings in the Ops Manager server’s `conf-mms.properties` file. The upgrade procedure uses the existing `conf-mms.properties` file to connect to the Ops Manager Application Database before replacing the existing file with the new, smaller 2.0 file.

**Restore properties** The following properties no longer apply and are replaced by settings specified when initiating a restore:

- `mms.backup.restore.linkExpirationHours`
- `mms.backup.restore.linkUnlimitedUses`

**mms.conf** If you have modified the `mms.conf` file in your current installation (which is not typical), back up the file. You must use the new `mms.conf` file installed by the upgrade. You can redo any modifications once the upgrade is complete.

## Prerequisite

**Hardware and Software Requirements** Your servers must meet the *Ops Manager Hardware and Software Requirements*.

**Warning:** Failure to configure servers according to the *Ops Manager Hardware and Software Requirements*, including the requirement to read the [MongoDB Production Notes](#), can lead to production failure.

**Administrator Privileges** You must have administrator privileges on the servers on which you perform the upgrade.

**Download Link** 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>.

## Procedure

There are no supported downgrade paths for Ops Manager.

---

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

---

To upgrade Ops Manager:

**Step 1: Take a full backup of the Ops Manager backing databases.**

**Step 2: Shut down Ops Manager.** To shutdown Ops Manager:

1. Click the *Start* button.
2. Click *Administrative Tools*.
3. Click *Services*.
4. Right-click the *MongoDB Ops Manager HTTP Service* and select *Stop*.
5. Right-click the *MongoDB Ops Manager Backup Daemon Service* and select *Stop*.
6. Right-click the *MongoDB Ops Manager Backup HTTP Service* and select *Stop*. (Pre-2.0 only)

**Step 3: Uninstall the Backup Daemon package. (Pre-2.0 only)** Do the following on **every server** where the Backup Daemon is installed:

1. Create a backup copy of each `conf-daemon.properties` file to preserve the existing configuration.
2. Uninstall the Backup Daemon package.

---

**Important:** If you are prompted to restart the Windows server after uninstalling Ops Manager, restart the server before installing the new version of Ops Manager.

---

**Step 4: Download the latest version of the Ops Manager package.**

1. In a browser, go to <http://www.mongodb.com> and click on the **Download** button.
2. Complete the form.
3. On the *MongoDB Download Center* page, click on the *Ops Manager* tab.
4. Select *Windows Server 2008 R2+* from the *Platforms* drop-down menu.
5. Select *MSI* from the *Packages* drop-down menu.
6. Click *Download*.

---

**Note:** The downloaded package is named `mongodb-mms-<version>.msi`, where `<version>` is the version number.

---

---

**Step 5: Install the Ops Manager MSI package on each server being used for Ops Manager.**

---

**Important:** On servers where there was only a Backup Daemon service, this will be a new install of Ops Manager package.

---

To install:

1. Double click the MSI package.
2. Follow the instructions in the *Setup Wizard*.
3. During setup, the *Configuration/Log Folder* step prompts you to specify a folder where the configuration and log files will be stored.

The installation restricts access to the folder to users with the *Administrator* access privileges only.

**Step 6: Copy the connection string to the new Ops Manager instances.**

1. Copy the `mongo.mongoUri` string from the `conf-mms.properties` file of the server that was upgraded to the current version of Ops Manager.
2. Replace the `mongo.mongoUri` string in each `conf-mms.properties` file in each servers which previously only hosted a Backup Daemon.

---

**Note:** The former Backup Daemon only servers used a different config file, but the upgrade to Ops Manager on those servers created a `conf-mms.properties` file. For these new instances to properly connect to the databases, this string must be replaced.

---

**Step 7: Start Ops Manager on every server.**

1. To start the service, click the *Start* button.
2. Click *Administrative Tools*.
3. Click *Services*.
4. In the *Services* list, right-click the *MongoDB Ops Manager HTTP Service* and select *Start*.

**Step 8: Log into Ops Manager as a user with the Global Owner role.** Ops Manager displays a setup wizard for entering the Ops Manager configuration in the Application Database.

**Step 9: Complete the Ops Manager configuration setup.** The setup wizard walks you through several pages to configure the Application Database. Refer to your existing `conf-mms.properties` file as needed.

**See also:**

[Ops Manager Configuration](#) for the mapping of `conf-mms.properties` parameters to setup wizard fields.

---

**Important:** Make sure you click *Save* on each tab, even if you do not make any changes.

---

**Step 10: Restart all Ops Manager services.**

**Step 11: For any Backup Agents that were installed manually, update their configurations to use port 8080.** Beginning with Ops Manager 2.0, Backup Agents access Ops Manager through port 8080. The Automation Agent automatically updates the ports on managed Backup Agents. For any Backup Agents that were installed **manually**, you must edit the agent's *configuration file* and set the *mothership* property to access Ops Manager on port 8080.

## 3 Create or Import a MongoDB Deployment

**Getting Started** Set up your first MongoDB deployment.

**Provision Servers** Provision servers for MongoDB deployments.

**Add Existing Processes to Monitoring** Add existing MongoDB processes to Ops Manager Monitoring.

**Add Monitored Processes to Automation** Add a monitored MongoDB deployment to be managed through Ops Manager Automation.

**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 Getting Started with Deployments

#### On this page

- [Get Started with Ops Manager](#)

Ops Manager is a web application that lets you create, manage, monitor and back up MongoDB deployments. The web application runs on the Ops Manager server, and software agents run on the servers running MongoDB.

You can use Ops Manager to create new deployments and manage existing ones. To monitor existing deployments, you need to download Ops Manager agents to the servers in the MongoDB deployment.

Ops Manager walks you through adding existing MongoDB deployments to monitor. The choices you make during setup do not limit your choices for the future.

#### Get Started with Ops Manager

**Step 1: Ask your Ops Manager administrator for your Ops Manager URL and whether you have a pre-existing Ops Manager group.**

If you do not have a pre-existing group, you will create one upon registering an account.

**Step 2: Open the Ops Manager URL in a browser.**

**Step 3: Click the *Register* link to create an account.**

Ops Manager walks you through registration and setup of your first deployment.

## 3.2 Provision Servers

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

Monitoring provides deployment metrics, visualization, and alerting on key database and hardware indicators. Automation provides all Monitoring functionality and lets you deploy, configure, and update your MongoDB processes directly from Ops Manager.

**Provision Servers for Automation** Add servers on which to deploy managed MongoDB deployments.

**Provision Servers for Monitoring** Add servers for monitored MongoDB deployments. Monitored deployments do not include automated management.

### Provision Servers for Automation

#### On this page

- [Overview](#)
- [Prerequisites](#)
- [Procedure](#)
- [Next Steps](#)

#### Overview

Ops Manager can automate operations for the MongoDB processes running on your servers. Ops Manager can both discover existing processes and deploy new ones.

Ops Manager Automation relies on an Automation Agent, which must be installed on every server that runs a monitored MongoDB deployment. The Automation Agents periodically poll Ops Manager to determine the goal configuration, deploy changes as needed, and report deployment status back to Ops Manager.

When you provision servers for Automation they are also provisioned for Monitoring. To provision servers for Automation, install the Automation Agent on each server.

#### Prerequisites

Before you can provision servers for automation you must meet the following prerequisites.

**Server Hardware** Each server must meet the following requirements.

- 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.

**Server Networking Access** The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server's FQDN by issuing `hostname -f` in a shell connected to the server. 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.

**Installing to a Server that Already Runs MongoDB** If you install the Automation Agent to a server that is already running a MongoDB process, the agent must have:

- Permission to stop the MongoDB process. The Automation Agent will restart the process using the agent's own set of MongoDB binaries. If you had installed MongoDB with a package manager, use the same package manager to install the Automation Agent. This gives the agent the same owner as MongoDB.
- Read and Write permissions on the MongoDB data directory and log directory.
- Permission to stop, start, and update any existing Monitoring and Backup Agents.

**Installing to a Server Before Installing MongoDB** If you deploy the Automation Agent to a server that does not have MongoDB installed, ensure the user that owns the Automation Agent has Read and Write permissions on the MongoDB data and log directories you plan to use.

**MongoDB Enterprise Dependencies** If you will run *MongoDB Enterprise*, you must manually install a set of dependencies to each server **before installing MongoDB**. The MongoDB manual provides the appropriate command to install the dependencies. See the link for the server's operating system:

- *Red Hat*
- *Ubuntu*
- *Debian*
- *SUSE*
- *Amazon AMI*

## Procedure

Install the Automation Agent on 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](#).

On Linux servers, if you installed MongoDB with a package manager, use the same package manager to install the Automation Agent. If you installed MongoDB without a package manager, use an archive to install the Automation Agent.

**Step 1: In Ops Manager, select the *Settings* tab and then select *Agents*.**

**Step 2: Under *Automation*, click your operating system and follow the instructions to install and run the agent.** For more information, see also [Install the Automation Agent](#).

## Next Steps

Once you have installed the agent to all your servers, you can deploy your first *replica set*, *cluster*, or *standalone*.

## Provision Servers for Monitoring

## On this page

- [Overview](#)
- [Prerequisites](#)
- [Procedure](#)

### Overview

To add your existing MongoDB deployments to Ops Manager [Monitoring](#) you must install a Monitoring Agent on one of the servers.

### Prerequisites

**Server Networking Access** The servers that host the MongoDB deployment must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server's FQDN by issuing `hostname -f` in a shell connected to the server.

**MongoDB Enterprise Dependencies** If you will run [MongoDB Enterprise](#), you must manually install a set of dependencies to each server **before installing MongoDB**. The MongoDB manual provides the appropriate command to install the dependencies. See the link for the server's operating system:

- [Red Hat](#)
- [Ubuntu](#)
- [Debian](#)
- [SUSE](#)
- [Amazon AMI](#)

### Procedure

Install the Monitoring Agent on one of the servers. See [Install Monitoring Agent](#).

## 3.3 Add Existing MongoDB Processes to Ops Manager

## On this page

- [Overview](#)
- [Considerations](#)
- [Add MongoDB Processes](#)

### Overview

Ops Manager provides a wizard for adding your existing MongoDB deployments to monitoring and management. The wizard prompts you to install an Automation Agent if none exists, and then prompts you to identify the [cluster](#), [replica set](#), or [standalone](#) to add. You can choose to add the deployment to [monitoring](#) or to both [monitoring](#) and [automation](#).

## Considerations

### Unique Names

Deployment items must have unique names within the group.

---

**Important:** Replica set, sharded cluster, and shard names within the same group must be unique. Failure to have unique names for the deployment items will result in broken backup snapshots.

---

### 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*.

### Authentication

If your deployment requires authentication, you must provide the necessary credentials when adding the deployment to Ops Manager. For information on configuring authentication, see [Configure MongoDB Authentication and Authorization](#).

### Automation and Updated Security Settings

Adding to automation may affect the security settings of the Ops Manager group or the MongoDB process.

- **Enables Ops Manager Group Security Setting.** If the MongoDB process requires authentication but the Ops Manager group does not have authentication settings enabled, upon successful addition of the MongoDB process to automation, the group's security settings will have the security settings of the newly imported deployment.

---

**Note:** The import process only enables the Ops Manager group's security setting if the group's security setting is currently not enabled. If the group's security setting is currently enabled, the import process does not disable the group's security setting or change its enabled authentication mechanism.

---

- **Updates MongoDB Users.** If the imported MongoDB process already has mms-backup-agent and mms-monitoring-agent users in the admin database, and the group's authentication settings are already enabled or will become enabled by the import process, the roles assigned to mms-backup-agent and mms-monitoring-agent will be overridden with the roles designated by the group.
- **Updates Ops Manager Group's MongoDB Users.** Regardless of the group's security setting, if the MongoDB process to import contains users, the import process will add these users to the group and apply the updated list of users to *all* processes in the group's deployment. During the import process, you can remove the users from importing into the group while allowing them to remain in an unmanaged state in the database. Only import the users you want managed since once imported, users cannot be "forgotten".
- **Updates Ops Manager Group's MongoDB User Roles.** If the MongoDB process contains user-defined roles, the import process will add these roles to the group. You can only remove these roles after the import process completes. That is, you can only remove roles from the group and all its managed processes as a whole.

---

**Note:** Custom roles are fully managed by Ops Manager, and the Automation agent will remove custom roles manually added to a database.

---

- 
- **Applies to All Deployments in Group.** The group's updated security settings apply to *all* deployments in the group and will restart all deployments in the group with the new setting, including the imported process. All processes will use the Ops Manager automation keyfile upon restart.

If the existing deployment or deployments in the group require a different security profile from the imported process, create a new group to import the MongoDB process.

## Add MongoDB Processes

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click *Add* and select *Existing MongoDB Deployment*.

**Step 3:** Follow the prompts to add the deployment.

## 3.4 Add Monitored Processes to Automation

### On this page

- Overview
- Considerations
- Prerequisites
- Procedure

### Overview

Ops Manager Automation lets you deploy, reconfigure, and upgrade your MongoDB databases directly from the Ops Manager console.

If Ops Manager is already monitoring your MongoDB processes, you can add them to Automation using this procedure. If Ops Manager is not monitoring your MongoDB processes, see instead [Add Existing MongoDB Processes to Ops Manager](#), which adds the processes to both Automation and Monitoring.

Automation relies on the Automation Agent, which you install on each server that hosts a process to be added to automated management. The Automation Agents regularly poll Ops Manager to determine goal configuration and deploy changes as needed. An Automation Agent must run as the same user and in the same group as the MongoDB process it will manage.

### Considerations

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](#).

If the MongoDB process or the Ops Manager group requires authentication, the addition of the MongoDB process to automation can update the security settings. See [Automation and Updated Security Settings](#).

## Prerequisites

### Ops Manager is Monitoring the Processes

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.

The Automation Agent must have:

- Permission to stop the MongoDB processes. The Automation Agent will restart the processes using the agent's own set of MongoDB binaries. If you had installed MongoDB with a package manager, use the same package manager to install the Automation Agent. This gives the agent the same owner as MongoDB.
- Read and Write permissions on the MongoDB data directories and log directories.

### The Process UID and GID must Match the Automation Agent

The user (UID) and group (GID) of the MongoDB process must match that of the Automation Agent. For example, if your Automation Agent runs as the “mongod” user in the “mongod” group, the MongoDB process must also run as the “mongod” user in the “mongod” group.

### Server Networking Access

The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server’s FQDN by issuing `hostname -f` in a shell connected to the server. 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.

## Procedure

### Add Monitored Processes to Automation

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Select the first of the two *Processes* icons.

**Step 3:** Click the ellipsis icon for the replica set, sharded cluster, or standalone and select *Properties*

**Step 4:** Click the *Add Automation* button.

**Step 5:** Follow the prompts to add the replica set, cluster, or standalone to Automation.

## 3.5 Deploy a Replica Set

### On this page

- Overview
- Unique Names for Deployment Items
- Prerequisites
- Procedure

### 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.

### Unique Names for Deployment Items

Use a unique name for the replica set.

---

**Important:** Replica set, sharded cluster, and shard names within the same group must be unique. Failure to have unique names for the deployment items will result in broken backup snapshots.

---

### Prerequisites

You must [provision servers](#) onto which to deploy, and Ops Manager must have access to the servers.

---

**Important:** If you will run [MongoDB Enterprise](#) and provision your own Linux servers, then you must manually install a set of dependencies to each server **before installing MongoDB**. The MongoDB manual provides the appropriate command to install the dependencies. See the link for the server's operating system:

- [Red Hat](#)
  - [Ubuntu](#)
  - [Debian](#)
  - [SUSE](#)
  - [Amazon AMI](#)
-

## Procedure

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the *Add* button and then select *New Replica Set*.

**Step 3:** Configure the replica set.

Enter information as required and click *Apply*.

The following table provides information for certain fields:

<i>Auth Schema</i>	Specifies the schema for storing user data. MongoDB 3.0 uses a different schema for user data than previous versions. For compatibility information, see the <a href="#">MongoDB Release Notes</a> .
<i>Version</i>	Specifies the servers to which Ops Manager deploys MongoDB. To let Ops Manager select from any of your provisioned servers, enter a period ("."). To select a specific set of servers, enter their common prefix. To use your local machine, enter the machine name.
<i>Eligible Server</i>	If you deploy a sharded cluster on MongoDB 3.2 or higher, the config servers deploy as a replica set. This field specifies the name for the replica set.
<i>RegExp</i>	Configures replica set members. By default, each member is a voting member that bears data. You can configure a member as an <a href="#">arbiter</a> , <a href="#">hidden</a> , <a href="#">delayed</a> , or <a href="#">having a certain priority in an election</a> .
<i>Config Server</i>	Creates a MongoDB index. For details, see <a href="#">Create Indexes</a> .
<i>Replica Set Name</i>	Configures additional runtime options. For option descriptions, see <a href="#">Advanced Options for MongoDB Deployments</a> .
<i>Member Options</i>	If you run MongoDB 3.0 or higher, you can choose a storage engine in <a href="#">Advanced Options</a> by adding the <i>engine</i> option. For information on storage engines, see <a href="#">Storage</a> in the MongoDB manual.
<i>Index Configuration</i>	
<i>Advanced Options</i>	

**Step 4: Optional. Make changes before you deploy.**

If needed, you can reconfigure processes and change the topology.

### To modify settings for a MongoDB process:

1. Click either of the *Processes* icons.
2. On the line listing the process, click the wrench icon.
3. Make changes as desired and click *Apply*.

### To move a process to a different server:

1. Click the first of the two *Servers* icons to display the topology.

**Note:** The colored bars on the right of each server indicate the replica set or sharded cluster to which the server belongs.

2. Drag and drop the process to a different server.

**Step 5: Click *Review & Deploy* to review your changes.**

**Step 6: Review and approve your changes.**

Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 3.6 Deploy a Sharded Cluster

### On this page

- Overview
- Unique Names for Deployment Items
- Prerequisites
- Procedure

### 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.

### Unique Names for Deployment Items

Use unique names for the new cluster and its shards.

---

**Important:** Replica set, sharded cluster, and shard names within the same group must be unique. Failure to have unique names for the deployment items will result in broken backup snapshots.

---

### Prerequisites

You must [provision servers](#) onto which to deploy, and Ops Manager must have access to the servers.

---

**Important:** If you will run [MongoDB Enterprise](#) and provision your own Linux servers, then you must manually install a set of dependencies to each server **before installing MongoDB**. The MongoDB manual provides the appropriate command to install the dependencies. See the link for the server's operating system:

- [Red Hat](#)
  - [Ubuntu](#)
  - [Debian](#)
  - [SUSE](#)
  - [Amazon AMI](#)
-

## Procedure

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the *Add* button and select *New Cluster*.

**Step 3:** Configure the sharded cluster.

Enter information as required and click *Apply*.

The following table provides information for certain fields:

<i>Auth Schema</i>	Specifies the schema for storing user data. MongoDB 3.0 uses a different schema for user data than previous versions. For compatibility information, see the <a href="#">MongoDB Release Notes</a> .
<i>Version</i>	Specifies the servers to which Ops Manager deploys MongoDB. To let Ops Manager select from any of your provisioned servers, enter a period ("."). To select a specific set of servers, enter their common prefix. To use your local machine, enter the machine name.
<i>Eligible Server</i>	If you deploy a sharded cluster on MongoDB 3.2 or higher, the config servers deploy as a replica set. This field specifies the name for the replica set.
<i>RegExp</i>	Configures replica set members. By default, each member is a voting member that bears data. You can configure a member as an <a href="#">arbiter</a> , <a href="#">hidden</a> , <a href="#">delayed</a> , or <a href="#">having a certain priority in an election</a> .
<i>Config Server</i>	Creates a MongoDB index. For details, see <a href="#">Create Indexes</a> .
<i>Replica Set Name</i>	Configures additional runtime options. For option descriptions, see <a href="#">Advanced Options for MongoDB Deployments</a> .
<i>Member Options</i>	If you run MongoDB 3.0 or higher, you can choose a storage engine in <a href="#">Advanced Options</a> by adding the <i>engine</i> option. For information on storage engines, see <a href="#">Storage</a> in the MongoDB manual.
<i>Index Configuration</i>	
<i>Advanced Options</i>	

**Step 4: Optional. Make changes before you deploy.**

If needed, you can reconfigure processes and change the topology.

**To modify settings for a MongoDB process:**

1. Click either of the *Processes* icons.
2. On the line listing the process, click the wrench icon.
3. Make changes as desired and click *Apply*.

**To move a process to a different server:**

1. Click the first of the two *Servers* icons to display the topology.

---

**Note:** The colored bars on the right of each server indicate the replica set or sharded cluster to which the server belongs.

---

2. Drag and drop the process to a different server.

**Step 5: Click *Review & Deploy* to review your changes.**

**Step 6: Review and approve your changes.**

Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 3.7 Deploy a Standalone MongoDB Instance

### On this page

- Overview
- Prerequisites
- Procedure

### 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 [provision a server](#) to which to deploy. For testing purposes, you can use your localhost.

### Procedure

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Click the *Add* button and select *New Standalone*.**

**Step 3: Configure the standalone MongoDB instance.**

Enter information as required and click *Apply*.

The *Auth Schema Version* field determines the schema for storing user data. MongoDB 3.0 uses a different schema for user data than previous versions. For compatibility information, see the [MongoDB Release Notes](#).

If you run MongoDB 3.0 or higher, you can choose a storage engine by selecting *Advanced Options* and adding the *engine* option. For information on storage engines, see [Storage](#) in the MongoDB manual.

For descriptions of *Advanced Options*, see [Advanced Options for MongoDB Deployments](#).

**Step 4: Click *Review & Deploy* to review your changes.**

**Step 5: Review and approve your changes.**

Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 3.8 Connect to a MongoDB Process

### On this page

- Overview
- Firewall Rules
- Procedures

### Overview

To connect to a MongoDB, retrieve the hostname and port information from Ops Manager and then use a MongoDB client, such as the `mongo` shell or a [MongoDB driver](#), to connect. To connect to a [\*cluster\*](#), retrieve the hostname and port for the `mongos` process. To connect to a [\*replica set\*](#) or [\*standalone process\*](#), retrieve the hostname and port for the `mongod` processes.

### 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. For information on firewalls on servers running MongoDB, see the firewall information in the [Network Security](#) document in the MongoDB manual.

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

You can retrieve a shell command for connecting to your MongoDB instance, or you can retrieve the host and port number of a specific MongoDB process and create your own command for connecting using either shell or [MongoDB driver](#).

#### Get a Shell Command to Connect to a MongoDB Instance

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Click the first of the two *Processes* icons.**

**Step 3: On the line listing the cluster, replica set, or process, click the ellipsis icon and select *Connect to this instance*.** Ops Manager provides a `mongo` shell command that you can use to connect to the MongoDB process.

## Get the Host and Port for a MongoDB Process

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** If the process is part of a sharded cluster, click the filter button for the type of process. Click one of the following:

<i>Shards</i>	Displays the mongod processes that host your data.
<i>Configs</i>	Displays the mongod processes that run as <i>config servers</i> to store a sharded cluster's metadata.
<i>Mongos</i>	Displays the mongos processes that route data in a sharded cluster.

**Step 4:** On the line listing the process, click the ellipsis icon and click *Performance Metrics*. 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

Get the host and port using the *above procedure*. From a shell, run `mongo` and specify the host and port. For example:

```
mongo --username <user> --password <pass> --host <host> --port <port>
```

## Connect to a Deployment Using a MongoDB Driver

Get the host and port using the *above procedure*. See your driver's instructions for creating a connection string that specifies the hostname and port.

For sharded clusters, you specify the hostname and port of the `mongos` instance. For a replica set, you specify a seed list of all hosts in the replica set. Your driver will automatically connect to the *primary*. For example:

```
mongodb://[<username>:<password>@]hostname0:<port>[,hostname1:<port1>][,hostname2:<port2>][...][,hos
```

# 4 Manage Deployments

**Edit a Deployment's Configuration** Edit the configuration of a MongoDB deployment.

**Create Indexes** Create indexes for your databases.

**Calculate Suggested Indexes** Retrieve suggested indexes for improving query performance.

**Edit a Replica Set** Add hosts to, remove hosts from, or modify the configuration of hosts in a managed replica set.

**Migrate a Replica Set Member** Migrate replica sets to new underlying systems by adding members to the set and decommissioning existing members.

**Convert Config Servers to a Replica Set** Convert the config servers to run as a replica set.

**MongoDB Processes** Start, stop, shut down, and remove MongoDB processes managed by Ops Manager.

**Move or Add an Agent** Migrate a backup and monitoring agents to different servers.

**MongoDB Versions** Configure available MongoDB versions, and upgrade or downgrade a deployment's version.

## 4.1 Edit a Deployment's Configuration

### On this page

- Overview
- Considerations
- Procedure

### Overview

You can modify a deployment's configuration and topology, including its MongoDB versions, storage engines, and numbers of hosts or shards. You can make modifications at all levels of a deployment's topology from the top-level, such as a *sharded cluster* or *replica set*, to lower levels, such as a replica set within a sharded cluster or an individual process within a replica set. You can also modify *standalone* processes.

### Considerations

#### MongoDB Version

To choose which versions of MongoDB are available to Ops Manager, see [Configure Available MongoDB Versions](#).

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.
- [MongoDB Compatibility](#)

Plan the version change during a predefined maintenance window.

Before applying the change to a production environment, change the MongoDB version on a staging environment that reproduces your production environment. This can help avoid discovering compatibility issues that may result in downtime for your production deployment.

If you *downgrade* to an earlier version of MongoDB and your MongoDB configuration file includes options that are not part of the earlier MongoDB version, you must perform the downgrade in two phases. First, remove the configuration settings that are specific to the newer MongoDB version, and deploy those changes. Then, update the MongoDB version and deploy that change.

For example, if you are running MongoDB version 3.0 with the *engine* option set to `mmapv1`, and you wish to downgrade to MongoDB 2.6, you must first remove the *engine* option as MongoDB 2.6 does not include that option.

#### Storage Engine

If you run or upgrade to MongoDB 3.0 or higher and modify the MongoDB storage engine, Ops Manager shuts down and restarts the MongoDB process. For a multi-member replica set, Ops Manager performs a rolling *initial sync* of each member.

Ops Manager creates backup directories during the migration from one storage engine to the other as long as the server has adequate disk space. Ops Manager *does not* delete the backup directories once the migration is complete. The backup directories are safe to keep and will not affect any future storage engine conversion, but they do take up disc

space. If desired, you can delete them. The backup directories are located in the `mongod`'s data directory. For example, if the data directory was `/data/process`, the backup would be `/data/process.bak.UNIQUENAME`. The `UNIQUENAME` is a random string that Ops Manager generates.

Before you can change the storage engine for a standalone instance or single-member replica set, you must give the Automation Agent write access to the MongoDB data directory's *parent* directory. The agent creates a temporary backup of the data in parent directory when updating the storage engine.

You cannot change the storage engine on a *config server*. For more information on storage engines and the available options, see [Storage](#) in the MongoDB manual.

## Fixed Properties

Certain properties of a deployment cannot be changed, including data paths, log paths, ports, and the server assignment for each MongoDB process.

## Deployment Topology

You can make modifications at all levels of a deployment's topology, including child processes. If you edit a child process, any future related edits to the parent might no longer apply to the child. For example, if you turn off journaling for a replica set member and then later change the journal commit interval for the replica set, the change will not apply to the member.

You can also modify the topology itself. To do so use this tutorial or one of the more specific tutorials available in this section of the manual, such as [Migrate a Replica Set Member to a New Server](#).

## Group-Level Modifications

Some modifications that affect a deployment occur at the group level. The following changes affect every MongoDB process in the group. For these changes, use the specified tutorials:

- To enable SSL for the deployment, see [Enable SSL for a Deployment](#).
- To enable authentication for the deployment, see [Enable Authentication for an Ops Manager Group](#).
- To add or modify MongoDB users and roles for the deployment, see [Manage MongoDB Users and Roles](#).

## Multiple Modifications

You can combine multiple modifications into one deployment. For example, you could make all the following modifications before clicking the *Review Changes* button:

- Add the latest stable version of MongoDB to the [Version Manager](#).
- Enable SSL for the deployment's MongoDB processes.
- Add a new sharded cluster running the latest stable version of MongoDB from above.

When you click *Review Changes*, the review displays all the changes on one screen for you to confirm before deploying.

## Procedure

To edit the deployment's configuration:

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Click the first of the two *Processes* icons.**

**Step 3: On the line listing the deployment item, click the wrench icon.**

A deployment item can be a *sharded cluster*, a *replica set*, a member of a sharded cluster or replica set, or a *standalone*.

**Step 4: Modify deployment settings.**

Make changes as appropriate and click *Apply*. Ops Manager locks fields that you cannot change. To add shards to a cluster or members to a replica set, increase the number of shards or members.

The following table provides information for certain fields:

<i>Auth Schema</i>	Specifies the schema for storing user data. MongoDB 3.0 uses a different schema for user data than previous versions. For compatibility information, see the <a href="#">MongoDB Release Notes</a> .
<i>Version</i>	
<i>Eligible Server</i>	Specifies the servers to which Ops Manager deploys MongoDB. To let Ops Manager select from any of your provisioned servers, enter a period ("."). To select a specific set of servers, enter their common prefix. To use your local machine, enter the machine name.
<i>RegExp</i>	
<i>Config Server</i>	If you deploy a sharded cluster on MongoDB 3.2 or higher, the config servers deploy as a replica set. This field specifies the name for the replica set.
<i>Replica Set Name</i>	
<i>Member Options</i>	Configures replica set members. By default, each member is a voting member that bears data. You can configure a member as an <a href="#">arbiter</a> , <a href="#">hidden</a> , <a href="#">delayed</a> , or <a href="#">having a certain priority in an election</a> .
<i>Index Configuration</i>	Creates a MongoDB index. For details, see <a href="#">Create Indexes</a> .
<i>Advanced Options</i>	Configures additional runtime options. For option descriptions, see <a href="#">Advanced Options for MongoDB Deployments</a> . If you run MongoDB 3.0 or higher, you can choose a storage engine in <a href="#">Advanced Options</a> by adding the <i>engine</i> option. For information on storage engines, see <a href="#">Storage</a> in the MongoDB manual.

**Step 5: Confirm any changes to topology.**

If you have added processes to a sharded cluster or replica set, select the first *Servers* icon to view where Ops Manager will deploy the processes. If you wish to move a process to a different server, drag and drop it.

**Step 6: Click *Review & Deploy* to review your changes.**

**Step 7: Review and approve your changes.**

Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 4.2 Create Indexes

## On this page

- [Overview](#)
- [Procedures](#)

## Overview

Ops Manager Automation lets you build MongoDB indexes without downtime for your deployment. The Automation Agent will sequentially take each individual node in a replica set or cluster offline to build the indexes. The agent builds the indexes without increasing load on your deployment or blocking read or write operations. You can create any type of MongoDB index. For available types, see [Index Types](#) in the MongoDB manual.

**Warning:** The Automation Agent does not currently support automated removal of an index from a collection. To remove an index that is managed by the Automation Agent, you must first remove the index from Automation and then manually remove the index from the collection. See the [Remove an Index](#) procedures below.

## Procedures

### Create an Index

**Warning:** To include the new index in any backups of the MongoDB instances, you must [resync backups](#) after performing this procedure.

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** On the line listing the deployment item, click the wrench icon. A deployment item can be a *sharded cluster* or *replica set*.

**Step 4:** Expand the *Index Configuration* section. If the *Index Configuration* section is not visible, collapse the open sections above it, such as *Cluster Configuration* or *Replica Set Configuration*. Or just scroll down to the *Index Configuration* section.

**Step 5:** Click *Add* and select the type of index. Select one of the following:

<i>Index</i>	An index on one or more fields. You can create indexes of any of the <a href="#">types described in the MongoDB manual</a> .
<i>TTL Index</i>	A single-field index that MongoDB can use to automatically remove documents from a database that stores transient information, such as logs. For more information, see the <a href="#">TTL Indexes</a> in the MongoDB manual.

**Step 6: Configure the index.** Enter the following information, according to index type, and click *Add*:

<i>Database and Collection Field</i>	The index applies to the specified collection in the specified database.  The field to index. For a regular index you can specify multiple fields using the <i>add another</i> link. The order of the fields is important. See <a href="#">Compound Indexes</a> in the MongoDB manual for more information.
<i>Value Index only</i>	For a <b>TTL index</b> you must specify a data field. TTL indexes expire documents after the specified number of seconds has passed since the indexed field value. Specifies either the sort order or index type. For more information on sort order, see the <a href="#">Index Introduction</a> and <a href="#">Compound Indexes</a> sections of the MongoDB manual. For more information on index types, see the following in the MongoDB manual: <ul style="list-style-type: none"><li>• <a href="#">Text Indexes</a></li><li>• <a href="#">Hashed Indexes</a></li><li>• <a href="#">2dsphere Indexes</a></li><li>• <a href="#">2d Indexes</a></li><li>• <a href="#">geoHaystack Indexes</a>, for the <i>Geospatial</i> selection option</li></ul>
<i>Expiration TTL Index only</i>	The number of seconds to wait after field value before removing the document. The expiration time is the field value plus the specified number of seconds.
<i>Unique Index only</i>	A unique index rejects documents that contain a duplicate value for the indexed field. For more information, see the <a href="#">Unique Indexes</a> in the MongoDB manual.
<i>Sparse Index only</i>	Sparse indexes only contain entries for documents that have the indexed field, even if the index field contains a null value. The index skips over any document that is missing the indexed field. For more information, see the <a href="#">Sparse Indexes</a> in the MongoDB manual.
<i>2d min/max Index only</i>	Stores the coordinates for a <i>2d</i> index.
<i>bucketSize Index only</i>	Stores the size of the buckets for a geoHaystack index. For more information on bucket size, see <a href="#">Create a Haystack Index</a> in the MongoDB manual.

**Step 7: Click *Apply*.**

**Step 8: Click *Review & Deploy*.**

**Step 9: Click *Confirm & Deploy*.**

#### Remove an Index

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** On the line listing the deployment item, click the wrench icon. A deployment item can be a *sharded cluster* or *replica set*.

**Step 4:** Expand the *Index Configuration* section. If the *Index Configuration* section is not visible, collapse the open sections above it, such as *Cluster Configuration* or *Replica Set Configuration*. Or just scroll down to the *Index Configuration* section.

**Step 5:** For the selected index, click *Remove*

**Step 6:** Click *Apply*.

**Step 7:** Click *Review & Deploy*.

**Step 8:** Click *Confirm & Deploy*.

**Step 9:** Manually remove the index, as described in Remove Indexes in the MongoDB manual.

## 4.3 Calculate Suggested Indexes

### On this page

- Overview
- Prerequisites
- Procedure

### Overview

Ops Manager can analyze the data on query patterns collected by the database profiler and suggest a set of indexes that could improve query performance. Ops Manager scores each suggested index on its expected benefit.

### Prerequisites

For Ops Manager to suggest indexes, the following must be true:

- You must enable database profiling for the MongoDB process. See [Profile Databases](#).
- The profiler must have data. If profiling is enabled, but no profiling data has yet been collected, Ops Manager cannot suggest indexes.

### Procedure

To edit the deployment's configuration:

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Click the first of the two *Processes* icons.**

**Step 3: If the process is part of a sharded cluster, click the filter button for the type of process.**

Click one of the following:

<i>Shards</i>	Displays the mongod processes that host your data.
<i>Configs</i>	Displays the mongod processes that run as <i>config servers</i> to store a sharded cluster's metadata.
<i>Mongos</i>	Displays the mongos processes that route data in a sharded cluster.

**Step 4: On the line listing the process, click the ellipsis icon and click *Performance Metrics*.**

**Step 5: Click the *Profiler* tab above the charts.**

**Step 6: Click the *calculate suggested indexes* link above the chart.**

The *calculate suggested indexes* link will not appear if the *Prerequisites* have not been met.

**Step 7: Copy the indexes you want to create and add them to the MongoDB process.**

For instructions on adding an index to a MongoDB process, see either:

- [Index Creation](#) in the MongoDB manual, or
- [Create Indexes with Automation](#) (if available on your account).

## 4.4 Edit a Replica Set

### On this page

- [Overview](#)
- [Procedures](#)
- [Additional Information](#)

### 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: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: On the line listing the replica set, click the wrench icon.** If you do not see the replica set, click the first of the two *Processes* icons.

**Step 3: Add and configure the new member.** Add the member by increasing the number of members in the *MongoDs Per Replica Set* field.

You can configure the member as a normal replica set member, as a *hidden*, *delayed*, or priority-0 secondary, or as an *arbiter*.

**Step 4: Click *Apply*.**

**Step 5: Click *Review & Deploy* to review your changes.**

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

### 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
- Change a member's votes.

To reconfigure a member as an *arbiter*, see [Replace a Member with an Arbiter](#).

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: On the line listing the replica set, click the wrench icon.** If you do not see the replica set, click the first of the two *Processes* icons.

**Step 3: In the *Member Options* box, configure each member as needed.** You can configure the member as a normal replica set member, as a *hidden*, *delayed*, or priority-0 secondary, or as an *arbiter*.

**Step 4: Click *Apply*.**

**Step 5: Click *Review & Deploy* to review your changes.**

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Replace a Member with an Arbiter

You cannot directly reconfigure a member as an arbiter. Instead, you must add a new member to the replica set as an arbiter. Then you must shut down an existing secondary.

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** Click the replica set's wrench icon.

**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* to review your changes.

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

**Step 7: Remove the secondary.** When the deployment completes, click the ellipsis icon for secondary to be removed, and then select *Remove from Replica Set*.

**Step 8: Click *Review Changes*.**

**Step 9: Click *Confirm & Deploy*.**

**Step 10: Remove the standalone instance. (Optional)** Upon completion, Ops Manager removes the member from the replica set. The instance continues to run as a *standalone*. To shut down the standalone, see [Shut Down a MongoDB Process](#).

## 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 a standalone instance.

When removing members, you must keep a majority of voting members active with respect to the original number of voting members. Otherwise your primary will step down and your replica set become read-only. You can remove multiple members at once **only if doing so leaves a majority**. For more information on voting, see [Replica Set Elections](#) and [Replica Set High Availability](#) in the MongoDB Manual.

Removing members might affect the ability of the replica set to acknowledge writes, depending on the level of *write concern* you use. For more information, see [Write Concern](#) in the MongoDB manual.

To remove a member:

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** Click the ellipsis icon for the member to be removed and select *Remove from Replica Set*.

**Step 4:** Click *Remove* to confirm.

**Step 5:** Click *Review & Deploy* to review your changes.

**Step 6:** Review and approve your changes. Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

**Step 7: Remove the standalone instance. (Optional)** Upon completion, Ops Manager removes the member from the replica set. The instance continues to run as a *standalone*. To shut down the standalone, see *Shut Down a MongoDB Process*.

## Additional Information

For more information on replica set configuration options, see, [Replica Set Configuration](#) in the MongoDB manual.

## 4.5 Migrate a Replica Set Member to a New Server

### On this page

- Overview
- Considerations
- Procedure

### 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 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

When migrating multiple members, you must keep a majority of voting members active with respect to the original number of voting members. Otherwise your primary will step down and your replica set become read-only. You can remove multiple members at once **only if doing so leaves a majority**. For more information on voting, see [Replica Set High Availability](#) and [Replica Set Elections](#) in the MongoDB Manual.

Removing members during migration might affect the ability of the replica set to acknowledge writes, depending on the level of *write concern* you use. For more information, see [Write Concern](#) in the MongoDB manual.

## Procedure

Perform this procedure separately for each member of a replica set to migrate.

### Step 1: Provision the new server.

See [Provision Servers](#).

### Step 2: Click the *Deployment* tab, then click the *Deployment* page.

### Step 3: On the line listing the replica set, click the wrench icon.

If you do not see the replica set, click the topology icon (the first *Processes* icon).

### Step 4: Add a member to the replica set.

In the *MongoDs Per Replica Set* field, increase the number of members by 1, and then click *Apply*.

### Step 5: Verify the server to which Ops Manager will deploy the new member.

Click the first *Servers* icon to view the server to which Ops Manager will deploy the new member. If Ops Manager has not chosen the server you intended, drag the new replica set member to the server to which to deploy it.

### Step 6: Click *Review & Deploy* to review your changes.

### Step 7: Review and approve your changes.

Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

#### **Step 8: Verify that the new member has synchronized.**

On the *Deployment* page, select one of the *Processes* icons 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.**

Click the member's ellipsis icon and select *Remove from Replica Set*. Click *Review Changes* and then click *Confirm & Deploy*.

#### **Step 10: Shut down the old member.**

Click the member's ellipsis icon and select *Shutdown*. Click *Review Changes* and then click *Confirm & Deploy*.

#### **Step 11: Optionally, remove the old member.**

To remove the member from Ops Manager management, see [Remove a Process from Management or Monitoring](#).

## **4.6 Convert Config Servers to a Replica Set**

### **On this page**

- [Overview](#)
- [Prerequisites](#)
- [Procedure](#)

### **Overview**

#### **Feature Availability**

- For deployments running MongoDB 3.2.4 or Later.
- Requires Ops Manager version 2.0.3 or greater.

In a sharded cluster, config servers store the metadata for the sharded cluster. For a sharded cluster to be fully operational, the config servers must be available and the metadata must be consistent across the config servers.

Beginning in MongoDB 3.2, *config servers*, that run the WiredTiger storage engine, can be deployed as replica sets. Running config servers as replica sets improves consistency and increases availability for the config servers.

**Warning:** If you have backup enabled and if you convert a sharded cluster's config servers to a replica set, then Ops Manager Automation can no longer use snapshots created before the conversion to restore the sharded cluster. Automation can only use snapshots created after the conversion.

For new sharded cluster deployments using MongoDB 3.2, Ops Manager deploys config servers as a replica set.

For existing sharded cluster deployments that are not running config servers as replica sets, you can convert the config servers to replica sets.

You can manage config server replica sets just as you would any other replica set.

## Prerequisites

- **Ops Manager version 2.0.3 or greater.** Ops Manager must be version 2.0.3 or greater.
- **MongoDB 3.2.4 or Later** To convert the config servers to a replica set, the sharded cluster must be running MongoDB 3.2.4 or later versions. To upgrade the MongoDB version, see [Change the Version of MongoDB](#).
- **Three Config Servers to Convert** The sharded cluster must be running exactly three config servers to convert. After the conversion of the config servers to a replica set, you can add additional members to the config server replica set.
- **Write Permission for Parent Directory of dbPath** The Automation Agent must have write permission on the **parent** directory of the dbPath of each config server.

## Procedure

**Note:** During the conversion of the config servers to a replica set,

- If the config servers use the MMAPv1 *storage engine*, Ops Manager changes the storage engine to WiredTiger.
- Ops Manager creates temporary config servers, and automation eventually deletes these config servers after the conversion. However, if your Monitoring Agent should detect them at any point during the conversion, they will appear as unreachable hosts in your deployments list. After the conversion, you can safely delete them from the list. See [Remove a Process from Management or Monitoring](#).

**Step 1: If you have not upgraded your sharded cluster to MongoDB 3.2.4 or later, do so now.**

See [Change the Version of MongoDB](#).

**Step 2: Click Deployment, then click the Processes tab, and then the Topology view.**

**Step 3: For the sharded cluster, under the Members column, click 3 CONFIGS.**

**Step 4: For the sharded cluster, click the ellipsis icon and select Convert Config Servers to Replica Sets.**

**Step 5: Enter a name for the new replica set.**

**Step 6: Enter the ports for the temporary config servers created during the conversion and click Upgrade.**

During conversion, the process creates temporary config servers. These temporary config servers must be network accessible from all nodes in the cluster. During the procedure, specify distinct available ports for these temporary config servers.

**Step 7: Click Review & Deploy.**

**Step 8: Click Confirm & Deploy.**

## 4.7 MongoDB Processes

**Shut Down a Process** Shut down MongoDB processes managed by Ops Manager.

**Restart a Process** Restart MongoDB processes managed by Ops Manager.

**Suspend Management of a Process** Temporarily suspend Automation's control over a process to allow manual maintenance.

**Remove a Process from Management or Monitoring** Remove MongoDB processes from Ops Manager management, monitoring, or both.

**Start Processes with Init Scripts** For an Automation Agent installed with an `rpm` or `deb` package, Ops Manager provides a tool that creates scripts to run your processes if you choose to stop using Automation.

## Shut Down a MongoDB Process

### On this page

- Overview
- Procedure

### Overview

Ops Manager supports shutting down individual `mongod` and `mongos` processes, as well as *replica sets* and *sharded clusters*. When you shut down a process, cluster, or replica set, Ops Manager continues to manage it, even though it is not running.

You can later *restart* your processes, or, if you no longer want Ops Manager to manage them, you can *remove* them.

### Procedure

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** On the line listing the cluster, replica set, or process, click the ellipsis icon and select *Shutdown*.

**Step 4:** Click *Shutdown* to confirm.

**Step 5:** Click *Review & Deploy* to review your changes.

**Step 6:** Review and approve your changes. Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Restart a MongoDB Process

### On this page

- Overview
- Considerations
- Procedure

### 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:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** On the line listing the cluster, replica set, or process, click the ellipsis icon and select *Startup*.

**Step 4:** Click *Startup* to confirm.

**Step 5:** Click *Review & Deploy* to review your changes.

**Step 6:** Review and approve your changes. Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Suspend or Resume Automation for a Process

### On this page

- Overview
- Procedures

## Overview

You can suspend Automation's control over a MongoDB process so that you can shut down the process for manual maintenance, without Automation starting the process back up again. Automation ignores the process until you return control.

When you resume Automation for a process, Ops Manager applies any changes that occurred while Automation was suspended.

If you wish to permanently remove a process from automation, see: [Remove a Process from Management or Monitoring](#).

## Procedures

### Suspend Automation for a Process

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** Click the ellipsis icon for the process and click *Pause Automation*.

**Step 4:** Click *Pause Automation* again to confirm.

**Step 5:** Click *Review & Deploy*.

**Step 6:** Click *Confirm & Deploy*.

### Resume Automation for a Process

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** Click the ellipsis icon for the process and click *Resume Automation*.

**Step 4:** Click *Resume Automation* again to confirm.

**Step 5:** Click *Review & Deploy* to review your changes.

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Remove a Process from Management or Monitoring

### On this page

- Overview
- Procedures

### Overview

You can remove a process from management, monitoring, or both. Removing a process does **not** shut down the process. You can remove individual processes or an entire *cluster* or *replica set*.

**Managed Processes** Removing a process from management (i.e., Automation) means you can no longer control it directly through Ops Manager. If you are removing a managed process and want to be able to easily stop and start the process after removal, you can *create scripts for stopping and starting the process*.

If you do **not** want to leave a managed process running after removal, *shut down the process* before removing it.

As an alternative to removing a process from automated management, you can temporarily *suspend* management of the process. This is useful if you need to temporarily shut down the process, for example for maintenance, and do not want Ops Manager to automatically start the process back up until you are ready.

**Monitored Processes** Removing a process from monitoring means Ops Manager no longer displays its status or tracks its metrics.

As an alternative to removing a process from monitoring, you can instead disable its alerts, which allows you to continue to view the process in the *Deployment* page but turns off notifications about its status. To use this option, see *Manage a Process's Alerts*.

### Procedures

#### Remove a Process from Management

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** On the line listing the cluster, replica set, or process, click the ellipsis icon and select the option to remove it.

**Step 4:** Select whether to also stop monitoring the process. Select one of the following and click *Remove*. If prompted for an authentication code, enter the code.

<i>Unmanage this item but continue to monitor</i>	Removes the process from management only. You can no longer control the process through Ops Manager, but Ops Manager continues to display its status and track its metrics.
<i>Completely remove</i>	Removes the processes from both management and monitoring. Ops Manager no longer displays the process.

**Step 5: Click *Review & Deploy* to review your changes.**

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

**Remove a Process from Monitoring** For processes that are monitored but not managed, do the following to remove the processes from monitoring:

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: On the line listing the process, click the ellipsis icon and select the option to remove it.**

**Step 3: Click the *Remove* button.** If prompted for an authentication code, enter the code.

## Start MongoDB Processes with Init Scripts

### On this page

- [Overview](#)
- [The “Make Init Scripts” Tool](#)
- [Procedures](#)

### Overview

Ops Manager provides a tool to create init scripts that will run your MongoDB processes if you should decide to stop managing them through Automation. The tool creates scripts that can start your processes automatically upon boot up or that you can use to start the processes manually.

The tool is available when you install the Automation Agent using an `rpm` or `deb` package.

### The “Make Init Scripts” Tool

When you install the Automation Agent using an `rpm` or `deb` package, the agent includes the following “Make Init Scripts” tool:

```
/opt/mongodb-mms-automation/bin/mongodb-mms-automation-make-init-scripts
```

The tool creates an init script for each `mongod` or `mongos` process on the server and has the following options:

Option	Description
-boot	Configures the init scripts to start the MongoDB processes on system boot. By default, the tool creates the scripts with this option disabled.
-cluster <string>	Specify the absolute path and filename of the local copy of the <i>automation configuration</i> <b>only if</b> the local copy does not use the default path and name, which are: <code>/var/lib/mongodb-mms-automation/mms-cluster-config-backup.json</code> The tool references the local copy of the configuration file to determine the desired state of the MongoDB processes.
-d <string>, -distribution <string> -h, -help	Specifies the Linux distribution. By default, the tool auto-detects the distribution. If this fails, specify your distribution as either <code>debian</code> or <code>redhat</code> .
	Describes the tool's options.

## Procedures

**Remove Processes from Automation and Create Init Scripts** This procedure creates scripts for stopping and starting MongoDB processes and then removes the processes from Automation.

Perform this procedure on each server that runs processes you want to remove from Automation. If you are removing a replica set, perform this on each replica set member separately, which allows you to remove the set from Automation without downtime.

**Step 1: Create the init scripts.** To create the init scripts, run the `mongodb-mms-automation-make-init-scripts` tool with superuser access. It is recommended that you use the `-boot` option so that you configure the scripts to start the MongoDB processes on system boot. Otherwise, you will be responsible to manually start each script.

To run the tool with superuser access and with the `-boot` option, issue:

```
sudo /opt/mongodb-mms-automation/bin/mongodb-mms-automation-make-init-scripts -boot
```

The tool places the init scripts in the `/etc/init.d` directory and names each one using the following form:

```
(mongod|mongos)-<process-name>
```

`<process-name>` is the name given to the `mongod` or `mongos` process in the cluster configuration.

**Step 2: Shut down each process.** In the Ops Manager *Deployment* tab, click the ellipsis icon for the process and select the option to shutdown. Deploy the changes. For detailed steps, see [Shut Down a MongoDB Process](#).

**Step 3: Remove each process from Automation.** In the Ops Manager *Deployment* tab, click the ellipsis icon for the process and select the option to remove it. Deploy the changes. For detailed steps, see [Remove a Process from Management](#).

**Step 4: Uninstall the Automation Agent.** If you installed the agent with an `rpm` package, issue following:

```
sudo rpm -e mongodb-mms-automation-agent-manager
```

If you installed the agent with an `deb` package, issue following:

```
sudo apt-get remove mongodb-mms-automation-agent-manager
```

**Step 5: Start each MongoDB process using its init script.** Issue the following for each process:

```
sudo /etc/init.d/<script-name> start
```

**Start or Stop a MongoDB Process using the Init Script** If you chose to run the “Make Init Scripts” tool without the –boot, then you must stop and start your MongoDB processes manually.

To start a MongoDB process using the init script, issue

```
sudo /etc/init.d/<script-name> start
```

To stop a MongoDB process using its init script, issue:

```
sudo /etc/init.d/<script-name> stop
```

## 4.8 Move or Add a Monitoring or Backup Agent

### On this page

- [Overview](#)
- [Procedures](#)

### 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: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Select the *Servers* tile view.** The *Servers* tile 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 ellipsis icon and select to install that type of agent.**

**Step 4:** On the server from which to remove the agent, click the ellipsis icon and remove the agent.

**Step 5:** Click *Review & Deploy* to review your changes.

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

#### 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:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Servers* icons.

**Step 3:** On the server to which to add an additional agent, click the ellipsis icon and select the agent to add.

**Step 4:** Click *Review & Deploy* to review your changes.

**Step 5: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 4.9 MongoDB Versions

**Change MongoDB Version** Upgrade or downgrade MongoDB deployments managed by Ops Manager.

**Configure Available MongoDB Versions** Configure the versions that are available for your Automation Agents to download.

### Change the Version of MongoDB

#### On this page

- Overview
- Considerations
- Procedure

## Overview

For MongoDB deployments managed by Ops Manager, 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*.

[Configure Available MongoDB Versions](#) describes how to choose which versions of MongoDB are available to Ops Manager.

If your deployment is not managed by Ops Manager, you will need to manually change the version of MongoDB. The MongoDB Manual provides upgrade tutorials with each release. For example, see: [Upgrade MongoDB to 3.0](#) for upgrading to MongoDB 3.0 from an earlier version.

## 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.
- [MongoDB Compatibility](#)

Plan the version change during a predefined maintenance window.

Before applying the change to a production environment, change the MongoDB version on a staging environment that reproduces your production environment. This can help avoid discovering compatibility issues that may result in downtime for your production deployment.

If you *downgrade* to an earlier version of MongoDB and your MongoDB configuration file includes options that are not part of the earlier MongoDB version, you must perform the downgrade in two phases. First, remove the configuration settings that are specific to the newer MongoDB version, and deploy those changes. Then, update the MongoDB version and deploy that change.

For example, if you are running MongoDB version 3.0 with the *engine* option set to `mmapv1`, and you wish to downgrade to MongoDB 2.6, you must first remove the *engine* option as MongoDB 2.6 does not include that option.

## Procedure

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Click the first of the two *Processes* icons.**

**Step 3: On the line listing the deployment item, click the wrench icon.**

**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 your changes.**

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Configure Available MongoDB Versions

### On this page

- [Overview](#)
- [Version Manager](#)
- [Procedure](#)

### 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 do not download available versions unless the version is in use by a MongoDB process on your server.

### Version Manager

The *Version Manager* lists the available MongoDB versions. Access the *Version Manager* from the *Deployment* tab.

### 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* to review your changes.

**Step 4: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 5 Monitoring and Alerts

[\*\*View Diagnostics\*\*](#) View diagnostics for processes, clusters, and replica sets.

[\*\*View Database Performance\*\*](#) Collect profile data for the host.

[\*\*View Logs\*\*](#) View host and agent logs.

[\*\*Manage Alerts\*\*](#) View, acknowledge, and unacknowledge alerts.

[\*\*Manage Alert Configurations\*\*](#) Create and manage alert configurations for a group.

**Alert Conditions** Identifies all available alert triggers and conditions.

**Global Alerts** Create and manage alert configurations for multiple groups at once.

**System Alerts** Describes internal health checks that monitor the health of Ops Manager itself.

## 5.1 View Diagnostics

### On this page

- [Overview](#)
- [Procedure](#)

### Overview

Ops Manager provides charts for analyzing the statistics collected by the Monitoring Agent for each process, replica set, and cluster.

### Procedure

To view diagnostics:

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Click the first of the two *Processes* icons.**

**Step 3: Click the ellipses icon for the process, cluster, or replica set and select *Performance Metrics*.**

You can alternatively hover your mouse over the process, cluster, or replica set and click *view metrics*.

**Step 4: Select the information to display.**

The *Data Size* field at the top of the page measures the data size on disk. See the explanation of `dataSize` on the [dbStats](#) page in the MongoDB manual.

The following table describes how to select what to display.

Task	Action
Select which components to display.	For a cluster: select whether to display <i>shards</i> , <i>mongos's</i> , or <i>config servers</i> using the buttons above the chart. For a shard, select whether to display <i>primaries</i> , <i>secondaries</i> , or both using the buttons below the chart. Select whether to display individual components using the checkmarks in the table below the chart. To isolate a few components from a large number, select the <i>None</i> button and then select the checkmarks for the components to display. For a replica set: select which members to display using the <i>P</i> and <i>S</i> icons above the charts. Hover the mouse pointer over an icon to display member information. To move up within a replica set or cluster, use the breadcrumb at the top of the page.
Select which charts to display.	For a cluster, select a chart from the <i>Chart</i> drop-down list. Ops Manager graphs the data for each component individually. To average or sum the data, click the <i>Averaged</i> or <i>Sum</i> buttons. For a replica set or process, select one or more charts from the <i>Add Chart</i> drop-down list.
Select data granularity and zoom.	Select the <i>Granularity</i> of the data displayed. The selected granularity option determines the available <i>Zoom</i> options. Hold the mouse button and drag the pointer over a portion of the chart. All charts will zoom to the same level. To reset the zoom, double-click the chart.
Expand an area of the chart.	Hover the mouse pointer over a chart. Click the <i>i</i> icon next to the chart name. Move the mouse pointer over a point on the chart. The data in the table below changes as you move the pointer. Click the two-way arrow at the top right of the chart. Hover the mouse pointer over the chart, click and hold the grabber in the upper left corner, and then drag the chart to the new position.
Display chart controls.	Click the curved arrow at the top right of the chart and select <i>Chart Permalink</i> .
View a chart's description.	Click the curved arrow at the top right of the chart and select <i>Email Chart</i> .
Display point-in-time statistics.	Select the <i>ellipsis</i> icon and select either <i>PDF</i> or <i>PNG</i> . Select the <i>ellipsis</i> icon and select either <i>Grid</i> or <i>Row</i> . Hover your mouse over the <i>clock</i> icon at the top of the page.
Pop-out an expanded view of the chart.	Hover the mouse pointer over the cluster name. Click the <i>pencil</i> icon and enter the new name.
Move a chart on the page.	Solid <b>vertical</b> bars on a chart indicate server events: <ul style="list-style-type: none"><li>• <i>Red</i> indicates a server restart.</li><li>• <i>Purple</i> indicates the server is now a primary.</li><li>• <i>Yellow</i> indicates the server is now a secondary.</li></ul>
Create a link to the chart.	
Email the chart.	
Export the charts.	
Display charts in a grid or row.	
View the times of the last and next snapshots.	
Change the name of the cluster.	
View sever events.	

## 5.2 Profile Databases

## On this page

- [Overview](#)
- [Considerations](#)
- [Procedures](#)

## Overview

Ops Manager can collect and display data from the MongoDB [database profilers](#) running on your mongod instances. An instance must have its profiler enabled for Ops Manager to collect data. The profiler provides statistics about database performance and operations. Ops Manager displays collected data in the **Profiler** section of an instance's metrics page.

To collect profiling data for a given mongod instance, you must enable profiling on the instance and enable Ops Manager to collect the profiling data from it. To enable both at the same time, see the procedure on this page to [Enable Profiling and the Collection of Profiling Data](#).

To enable Ops Manager collection of profiling data when the profiler is already enabled on a process, see the procedure on this page to [Enable Collection of Profiling Data](#). To enable or disable Ops Manager collection of profiling data for all processes in the group, see [Group Settings](#).

**Please read the considerations below before enabling profiling.**

## Considerations

Before enabling profiling, be aware of the following.

### Security

Profile data can include sensitive information, including the content of database queries. Ensure that exposing this data to Ops Manager is consistent with your information security practices.

### Resources

The profiler can consume resources which may adversely affect MongoDB performance. Consider the implications before enabling profiling.

Ops Manager samples up to 20 entries or 4MB of data, whichever comes first. The 4MB limit is reached only if you have increased the default size the `system.profile` collection (which by default is 1MB) *and* if you have large profiler documents.

The agent sends only the most recent 20 entries from the last minute.

If you use a Monitoring Agent version **earlier than version 3.8.0.230**, and if the agent authenticates as a user containing the `clusterAdmin` role, the agent might leak cursors if you enable profiling. A leaked cursor will automatically time-out on the server after 10 minutes.

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, Ops Manager 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.

## Time to Propagate

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.

## Profiled Operations

The MongoDB profiler stores data in the <database>.system.profile collections. When collecting data from the profiler, Ops Manager ignores data about operations on the <database>.system.profile collections themselves, such as Monitoring Agent queries of the <database>.system.profile collections.

## Procedures

### Enable Profiling and the Collection of Profiling Data

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the ellipses icon for the process and select *Performance Metrics*. Alternatively, you can hover your mouse over the process and click *view metrics*.

**Step 3:** Click the *Profiler* tab above the charts.

**Step 4:** Toggle the *Profiling* button to *On*.

**Step 5: Define slow operations.** By default, profiling collects data for operations that take longer than 100 milliseconds. The threshold for slow operations applies to all databases on the process. To change this threshold, click the number of milliseconds and enter a new number.

**Step 6:** Click *Review & Deploy* to review your changes.

**Step 7: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

### Enable Collection of Profiling Data

This procedure enables Ops Manager to collect profiling data from a mongod instance. You must separately enable the instance's profiler, either upon deployment or through the `setProfilingLevel` command. If using Automation, you can enable profiling upon deployment by setting the `profile` option under *Advanced Options*.

To enable collection of profiling data from an instance's active profiler:

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the ellipses icon for the process and select *Monitoring Settings*.

**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.

## 5.3 View Logs

### On this page

- Overview
- MongoDB Real-Time Logs
- MongoDB On-Disk Logs
- Agent Logs

### 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: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Click the first of the two *Processes* icons.**

**Step 3: If the process is part of a sharded cluster, click the filter button for the type of process.** Click one of the following:

<code>Shards</code>	Displays the <code>mongod</code> processes that host your data.
<code>Configs</code>	Displays the <code>mongod</code> processes that run as <i>config servers</i> to store a sharded cluster's metadata.
<code>Mongos</code>	Displays the <code>mongos</code> processes that route data in a sharded cluster.

**Step 4: On the line listing the process, click the ellipsis icon and click *Performance Metrics*.**

**Step 5: Click the *Logs* tab.** The tab displays log information. If the tab instead displays the *Collect Logs For Host* option, toggle the option to *On* and refresh the page.

**Step 6: Refresh the browser window to view updated entries.**

#### **Enable or Disable Log Collection for a Deployment**

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Select the list view by clicking the second of the two *Processes* icons.

**Step 3:** On the line for any process, click the ellipsis icon and select *Monitoring Settings*.

**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 *Settings* 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 ellipsis icon for a process and select *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 *Settings* 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 *Settings* tab and then *Agents* page.

**Step 2:** Scroll down to the *Agent Log Settings* section.

**Step 3: Edit the log settings.** Click the *pencil* icon to edit an agent's log settings. You can edit the following settings. When you are done, click *Confirm*:

Name	Type	Description
<i>Linux Log Path</i>	string	The path to which the agent writes its logs.
<i>Rotate Logs</i>	boolean	Specifies whether Ops Manager should rotate the logs for the agent.
<i>Size Threshold (MB)</i>	number	Max size in MB for an individual log file before rotation.
<i>Time Threshold (Hours)</i>	integer	Max time in hours for an individual log file before rotation.
<i>Max Uncompressed Files</i>	integer	<i>Optional</i> Max number of total log files to leave uncompressed, including the current log file.
<i>Max Percent of Disk</i>	number	<i>Optional</i> Max percent of the total disk space all log files can occupy before deletion.

**Step 4:** Return to the *Deployment* page

**Step 5:** Click *Review & Deploy* to review your changes.

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 5.4 Manage Alerts

### On this page

- [Overview](#)
- [Procedures](#)

### Overview

Ops Manager sends an alert notification when a specified *alert condition* occurs, such as an unresponsive host or an outdated agent. You can view your alerts through the *Activity* tab, as well as through the alert notification.

When a condition triggers an alert, you receive the alert at regular intervals until the alert resolves or Ops Manager cancels it. You can acknowledge an alert for a period of time, but if the alert condition persists, you will again receive notifications once the acknowledgment period ends.

You can temporarily suspend alerts on a resource by creating an alert *maintenance window*. For example, you can create a maintenance window that suspends host alerts while you shut down hosts for maintenance.

Ops Manager administrators define alert configurations on a *per-group* or *multi-group* basis.

### Resolved Alerts

Alerts resolve when the alert condition no longer applies. For example, if a replica set's *primary* goes down, Ops Manager issues an alert that the replica set does not have a primary. When a new primary is elected, the alert condition no longer applies, and the alert will resolve. Ops Manager sends a notification of the alert's resolution.

### Cancelled Alerts

Ops Manager cancels an alert if the alert configuration that triggered the alert is deleted, disabled, or edited, or if the open alert becomes invalid. Some examples of an alert becoming invalid are:

- There is an open "Host Down" alert, and then you delete the target host.
- There is an open "Replication Lag" alert, and the target host becomes the primary.
- There is an open "Replica Set Has No Primary" alert for a replica set whose name is "rs0," and the target replica set is renamed to "rs1."

When an alert is canceled, Ops Manager does **not** send a notification and does not record an entry in the *Activity* tab.

### Procedures

#### View Alerts

**Step 1: Select the *Activity* tab.**

Step 2: Use the fields above the list to filter which alerts to display.	
To filter by:	Do this:
Open or closed Alert type Date	Select the appropriate button above the list. To view both open and closed alerts, click <i>All Activity</i> . Closed alerts are those that users have closed or that no longer meet the alert condition. Type the <i>alert type</i> in the <i>Types</i> box and click <i>Filter</i> . Select the dates in the <i>From</i> and <i>To</i> fields and click <i>Filter</i> .

### Download the Activity Feed

You can download the activity feed as a CSV file with comma-separated values. You can filter the events before downloading. Ops Manager limits the number of events returned to 10,000.

**Step 1:** Select the *Activity* tab.

**Step 2:** To specify events from a particular time period, enter the dates and click *Filter*.

**Step 3:** Click the ellipsis icon and select *Download Activity Feed*.

### Acknowledge an Alert

When you acknowledge the alert, Ops Manager sends no further notifications to the alert's distribution list until the acknowledgement period has passed or until you resolve the alert. 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.

If you configure an alert with PagerDuty, a third-party incident management service, you can only acknowledge the alert on your PagerDuty dashboard.

**Step 1:** Select the *Activity* tab.

**Step 2:** Select *Open Alerts*.

### Step 3: Acknowledge alerts

- For a single alert, on the line item for the alert, click *Acknowledge*.
- For multiple, but not all, alerts:
  - Check the checkbox to the left of each alert to acknowledge.
  - Click *Acknowledge* above the table.
- For all alerts, check the checkbox in the table header to select every alert.
  - It is also then possible to uncheck a few checkboxes to select fewer than all alerts.

**Step 4: Select the time period for which to acknowledge the alert.** Click the time frame for which you no longer wish to receive alerts.

Ops Manager will send no further alert messages for the period of time you select.

**Step 5: If all Alerts were checked, select All Visible or All Open Alerts. (Optional)** If all alerts are checked, then another set of radio buttons appear:

- Clicking *Acknowledge all visible checked alerts* will acknowledge all alerts that were loaded onto the page.
- Clicking *Acknowledge all open alerts* will acknowledge all alerts: checked, unchecked, visible on the current page and those that have not been loaded on the current page.

**Step 6: Click *Acknowledge*.**

#### Unacknowledge an Alert

You can undo an acknowledgment and again receive notifications if the alert condition still applies.

**Step 1: Select the *Activity* tab.**

**Step 2: Select *Open Alerts*.**

**Step 3: On the line item for the alert, click *Unacknowledge*.**

**Step 4: Click *Confirm*.** If the alert condition continues to exist, Ops Manager will resend alerts.

#### Manage a Process's Alerts

You can turn off alerts for a given process. This might be useful, for example, if you want to temporarily disable the process but do not want it hidden from monitoring. Use the following procedure both to turn alerts off or on.

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: On the line listing the process, click the ellipsis icon and select *Monitoring Settings*.**

**Step 3: Select *Alert Status* and then modify the alert settings.**

#### Suspend Alerts by Adding a Maintenance Window

Specify maintenance windows to temporarily turn off alert notifications for a given resource while you perform maintenance. To view maintenance windows, select the *Activity* tab and then select the *Maintenance Windows* filter.

#### Add or Edit a Maintenance Window

**Step 1: Select the *Activity* tab.**

**Step 2:** Click the *ellipsis icon* and select *Alert Settings*.

**Step 3:** Add or edit the maintenance window.

- **Add a maintenance window:** Click the *Add* button and select *New Maintenance Window*.
- **Edit a maintenance window:** Click the *Maintenance Windows* filter. Click the window's *ellipsis icon* and select *Edit*.

**Step 4:** Select the target components for which to suspend alerts. Note that selecting the *Host* target selects both *HOST* and *HOST\_METRIC* alert configurations returned through the *alertConfigs endpoint*.

**Step 5:** Select the time period for which to suspend alerts.

**Step 6:** Enter an optional description for the maintenance window.

**Step 7:** Click *Save*.

#### Delete a Maintenance Window

**Step 1:** Select the *Activity* tab.

**Step 2:** Click the *ellipsis icon* and select *Alert Settings*.

**Step 3:** Click the *Maintenance Windows* filter.

**Step 4:** For the window to delete, click the *ellipsis icon* and select *Delete*.

**Step 5:** Click *Confirm*.

## 5.5 Manage Alert Configurations

### On this page

- Overview
- Considerations
- Procedures

### Overview

An alert configuration defines the *conditions* that trigger an alert and the alert's notification methods. This tutorial describes how to create and manage the alert configurations for a specified group. To create and manage global alert configurations, see *Global Alerts*.

## **Default Alert Configurations**

Ops Manager creates the following alert configurations for a group automatically upon creation of the group:

- [Users awaiting approval to join group](#)
- [is exposed to the public Internet](#)
- [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](#)

## **Default Settings for Notification Options**

Group administrators can configure default settings for the following notification options:

- PagerDuty
- Flowdock
- HipChat
- Slack
- Webhook

To configure default settings for one of the above notification options, select the *Settings* tab and then select the *Group Settings* page. Locate the notification option and enter the default values.

Ops Manager will fill in the default values automatically when a user selects that option when creating an alert configuration.

## **Considerations**

### **SMS Delivery**

Many factors may affect alert delivery, including do not call lists, caps for messages sent or delivered, delivery time of day, and message caching.

Check with your telephone service contract for the costs associated with receiving text messages.

If you choose SMS, Ops Manager sends alert text messages to all users in the group who have filled in their mobile numbers for their accounts.

### **Alert Intervals**

You can create multiple alert configurations with different frequencies. The minimum frequency for an alert is 5 minutes.

The time between re-notifications increases by the frequency amount every alert cycle up to a maximum of 24 hours. For example, if the frequency amount is 5 minutes, and the alert condition is first triggered at 9am, subsequent alerts occur at 9:05am, 9:15am, 9:30am, etc.

You can set the time to elapse before Ops Manager sends an alert after an alert condition occurs. This helps eliminate false positives.

## Procedures

These procedures apply to alert configurations assigned to a specific group. For global alert configurations, see [Global Alerts](#).

### Create a New Alert Configuration

When you create a new alert configuration you have the option of using an existing configuration as a template.

**Step 1: Select the *Activity* tab.**

**Step 2: Click the *ellipsis icon* and select *Alert Settings*.**

**Step 3: Select whether to use an existing alert as a template.** Do one of the following:

- To use an existing alert configuration as a template, click the configuration's *ellipsis icon* and select *Clone*.
- To create the alert without pre-filled information, click the *Add* button and select *New Alert*.

**Step 4: Select the condition that triggers the alert.** In the *Alert if* section, select the target component. If you select *Host*, select the type of host. Then select the condition and, if applicable, the threshold for the metric. For explanations of alert conditions and metrics, see [Alert Conditions](#).

**Step 5: Apply the alert to specific targets, if applicable.** If the options in the *For* section are available, you can optionally filter the alert to apply only to a subset of the targets by selecting *Hosts where ....*

Of the available filter conditions, *is*, *is not*, *contains*, *does not contain*, *starts with*, and *ends with* use direct string comparison, while *matches* uses regular expressions.

The *For* options are only available for replica sets and hosts.

**Step 6: Select the alert recipients and delivery methods.** In the *Send to* section, configure notifications. To add notifications or recipients, click *Add* and select from the options listed below. To test a notification, click the test link that appears after you configure the notification.

Notification Option	Description
Group ( <i>group or global alerts only</i> )	Sends the alert by email or SMS to the group. If you select <i>SMS</i> , Ops Manager sends the text message to the number configured on each user's <a href="#">Account page</a> . To send only to specific roles, deselect <i>All Roles</i> and select the desired roles.
Ops Manager User	Sends the alert by email or SMS to a specified Ops Manager user. If you select <i>SMS</i> , Ops Manager sends the text message to the number configured on the user's <a href="#">Account page</a> .
SNMP Host	Specify the hostname that will receive the v2c trap on standard port 162. The MIB file for SNMP is available for <a href="#">download here</a> .
Email	Sends the alert to a specified email address.
HipChat	Sends the alert to a HipChat room message stream. Enter the HipChat room name and API token.
Slack	Sends the alert to a Slack channel. Enter the channel name and either an API token or a Bot token. To create an API token, see the <a href="https://api.slack.com/web">https://api.slack.com/web</a> page in your Slack account. For information on Bot users in Slack, see <a href="https://api.slack.com/bot-users">https://api.slack.com/bot-users</a> .
Flowdock	Sends the alert to a Flowdock account. Enter the following:
	<ul style="list-style-type: none"> <li>• <i>Org Name</i>: The Flowdock organization name in lower-case letters. This is the name that appears after <code>www.flowdock.com/app/</code> in the URL string.</li> <li>• <i>Flow Name</i>: The flow name in lower-case letters. The flow name appears after the org name in the URL string: <code>www.flowdock.com/app/orgname/flowname</code>. The flow name also appears in the “flow email address” setting in Flowdock. For example: <code>flowname@example.flowdock.com</code>.</li> <li>• <i>User API Token</i>: Your Flowdock “personal API token” found on the <a href="https://www.flowdock.com/account/tokens">https://www.flowdock.com/account/tokens</a> page of your Flowdock account.</li> </ul>
PagerDuty	Enter only the service key. Define escalation rules and alert assignments in <a href="#">PagerDuty</a> .
Webhook ( <i>group alerts only</i> )	Sends an HTTP POST request to an endpoint for programmatic processing. The request body contains a JSON document that uses the same format as the Public API's <a href="#">Alerts resource</a> . This option is available only if you have configured Webhook settings on the <a href="#">Group Settings</a> page.
Administrators ( <i>global or system alerts only</i> )	Sends the alert to the email address specified in the <i>Admin Email Address</i> field in the Ops Manager <a href="#">configuration options</a> .

**Step 7: Click Save.**

## 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. The minimum frequency for sending an alert is 5 minutes.

**Step 1:** Select the *Activity* tab.

**Step 2:** Click the *ellipsis icon* and select *Alert Settings*.

**Step 3:** On the line listing the alert configuration, click the *ellipsis icon* and select *Edit*.

**Step 4: Select the condition that triggers the alert.** In the *Alert if* section, select the target component. If you select Host, select the type of host. Then select the condition and, if applicable, the threshold for the metric. For explanations of alert conditions and metrics, see [Alert Conditions](#).

**Step 5: Apply the alert to specific targets, if applicable.** If the options in the *For* section are available, you can optionally filter the alert to apply only to a subset of the targets by selecting *Hosts where ....*

Of the available filter conditions, *is*, *is not*, *contains*, *does not contain*, *starts with*, and *ends with* use direct string comparison, while *matches* uses regular expressions.

The *For* options are only available for replica sets and hosts.

**Step 6: Select the alert recipients and delivery methods.** In the *Send to* section, configure notifications. To add notifications or recipients, click *Add* and select from the options listed below. To test a notification, click the test link that appears after you configure the notification.

Notification Option	Description
Group ( <i>group or global alerts only</i> )	Sends the alert by email or SMS to the group. If you select <i>SMS</i> , Ops Manager sends the text message to the number configured on each user's <a href="#">Account page</a> . To send only to specific roles, deselect <i>All Roles</i> and select the desired roles.
Ops Manager User	Sends the alert by email or SMS to a specified Ops Manager user. If you select <i>SMS</i> , Ops Manager sends the text message to the number configured on the user's <a href="#">Account page</a> .
SNMP Host	Specify the hostname that will receive the v2c trap on standard port 162. The MIB file for SNMP is available for <a href="#">download here</a> .
Email	Sends the alert to a specified email address.
HipChat	Sends the alert to a HipChat room message stream. Enter the HipChat room name and API token.
Slack	Sends the alert to a Slack channel. Enter the channel name and either an API token or a Bot token. To create an API token, see the <a href="https://api.slack.com/web">https://api.slack.com/web</a> page in your Slack account. For information on Bot users in Slack, see <a href="https://api.slack.com/bot-users">https://api.slack.com/bot-users</a> .
Flowdock	Sends the alert to a Flowdock account. Enter the following:
	<ul style="list-style-type: none"> <li>• <i>Org Name</i>: The Flowdock organization name in lower-case letters. This is the name that appears after <code>www.flowdock.com/app/</code> in the URL string.</li> <li>• <i>Flow Name</i>: The flow name in lower-case letters. The flow name appears after the org name in the URL string: <code>www.flowdock.com/app/orgname/flowname</code>. The flow name also appears in the “flow email address” setting in Flowdock. For example: <code>flowname@example.flowdock.com</code>.</li> <li>• <i>User API Token</i>: Your Flowdock “personal API token” found on the <a href="https://www.flowdock.com/account/tokens">https://www.flowdock.com/account/tokens</a> page of your Flowdock account.</li> </ul>
PagerDuty	Enter only the service key. Define escalation rules and alert assignments in <a href="#">PagerDuty</a> .
Webhook ( <i>group alerts only</i> )	Sends an HTTP POST request to an endpoint for programmatic processing. The request body contains a JSON document that uses the same format as the Public API's <a href="#">Alerts resource</a> . This option is available only if you have configured Webhook settings on the <a href="#">Group Settings</a> page.
Administrators ( <i>global or system alerts only</i> )	Sends the alert to the email address specified in the <i>Admin Email Address</i> field in the Ops Manager <a href="#">configuration options</a> .

**Step 7: Click Save.**

## Delete an Alert Configuration

If you delete an alert configuration that has open alerts, Ops Manager cancels the open alerts whether or not they have been acknowledged and sends no further notifications.

**Step 1:** Select the *Activity* tab.

**Step 2:** On the line listing the alert configuration, click the ellipsis icon and *Delete*.

**Step 3:** Click *Confirm*.

## Disable or Enable an Alert Configuration

When you disable an alert configuration, Ops Manager cancels active alerts related to the disabled configuration. The configuration remains visible but grayed-out and can be later re-enabled. If you need to disable an alert only for a period of time, you can alternatively *suspend alerts*.

To disable or enable an alert configuration:

**Step 1:** Select the *Activity* tab.

**Step 2:** On the line listing the alert configuration, click the ellipsis icon and select either *Disable* or *Enable*.

## View the History of Changes to an Alert Configuration

**Step 1:** Select the *Activity* tab.

**Step 2:** Click the *ellipsis icon* and select *Alert Settings*.

**Step 3:** On the line listing the alert configuration, click the ellipsis icon and select *History*. Ops Manager displays the history of changes to the alert configuration.

## 5.6 Alert Conditions

### On this page

- Overview
- Host Alerts
- Replica Set Alerts
- Agent Alerts
- Backup Alerts
- User Alerts
- Group Alerts

## Overview

When you create a [group](#) or [global](#) alert configuration, specify the targets and alert conditions described here.

## Host Alerts

When configuring an alert that applies to hosts, you select the host type as well as the alert condition.

### Host Types

For host type, you can apply the alerts to all MongoDB processes or to a specific type of process:

Host Type	Applies To
Any type	All the types described here.
Stan-dalone	Any <a href="#">mongod</a> instance that is <b>not</b> part of a replica set or sharded cluster and that is <b>not</b> used as a config server.
Primary	All replica set <a href="#">primaries</a> .
Secondary	All replica set <a href="#">secondaries</a> .
Mongos	All <a href="#">mongos</a> instances.
Conf	All mongod instances used as <a href="#">config servers</a> .

### Host Alert Conditions

**Status** The following conditions apply to a change in status for a MongoDB process:

#### **is added**

Sends an alert when Ops Manager starts monitoring or managing a mongod or mongos process for the first time.

#### **is removed**

Sends an alert when Ops Manager stops monitoring or managing a mongod or mongos process for the first time.

#### **is added to replica set**

Sends an alert when the specified type of mongod process is added to a [replica set](#).

#### **is removed from replica set**

Sends an alert when the specified type of mongod process is removed from a [replica set](#).

#### **is down**

Sends an alert when Ops Manager does not receive a ping from a host for more than 4 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 4 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](#).

**Asserts** The following alert conditions measure the rate of asserts for a MongoDB process, as collected from the MongoDB `serverStatus` command's `asserts` document. You can view asserts through [deployment metrics](#).

#### **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** The following alert conditions measure the rate of database operations on a MongoDB process since the process last started, as collected from the MongoDB `serverStatus` command's `opcounters` document. You can view opcounters through [deployment metrics](#).

**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** The following alert conditions measure the rate of database operations on MongoDB `secondaries`, as collected from the MongoDB `serverStatus` command's `opcountersRepl` document. You can view these metrics on the *Opcounters - Repl* chart, accessed through [deployment metrics](#).

**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** The following alert conditions measure memory for a MongoDB process, as collected from the MongoDB `serverStatus` command's `mem` document. You can view these metrics on the Ops Manager *Memory* and *Non-Mapped Virtual Memory* charts, accessed through [deployment metrics](#).

**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 [Procedure](#).

**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 [Procedure](#).

**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 [Procedure](#).

**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 measures connections to a MongoDB process, as collected from the MongoDB `serverStatus` command's `connections` document. You can view this metric on the Ops Manager `Connections` chart, accessed through [deployment metrics](#).

**Connections is**

Sends an alert if the number of active connections to the host meets the specified average.

**Queues** The following alert conditions measure operations waiting on locks, as collected from the MongoDB `serverStatus` command's `globalLock` document. You can view these metrics on the Ops Manager `Queues` chart, accessed through [deployment metrics](#).

**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 *Procedure*.

**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** The following alert conditions measure the number of *cursors* for a MongoDB process, as collected from the MongoDB serverStatus command's `metrics.cursor` document. You can view these metrics on the Ops Manager *Cursors* chart, accessed through *deployment metrics*.

**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** The following alert conditions measure throughput for MongoDB process, as collected from the MongoDB serverStatus command's `network` document. You can view these metrics on a host's *Network* chart, accessed through *deployment metrics*.

**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 Oplg** The following alert conditions apply to the MongoDB process's `oplog`. You can view these metrics on the following charts, accessed through *deployment metrics*:

- *Replication Oplog Window*
- *Replication Lag*
- *Replication Headroom*
- *Oplog GB/Hour*

The following alert conditions apply to the oplog:

**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 *Procedure*.

**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 *Procedure*.

**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.

**WiredTiger Storage Engine** The following alert conditions apply to a MongoDB process's `WiredTiger` storage engine, as collected from the MongoDB `serverStatus` command's `wiredTiger.cache` and `wiredTiger.concurrentTransactions` documents.

You can view these metrics on the following charts, accessed through *deployment metrics*:

- *Tickets Available*
- *Cache Activity*
- *Cache Usage*

The following are the alert conditions that apply to WiredTiger:

**Tickets Available: Reads is**

Sends an alert if the number of read tickets available to the WiredTiger storage engine meet the specified threshold.

**Tickets Available: Writes is**

Sends an alert if the number of write tickets available to the WiredTiger storage engine meet the specified threshold.

**Cache: Dirty Bytes is**

Sends an alert when the number of dirty bytes in the WiredTiger cache meets the specified threshold.

**Cache: Used Bytes is**

Sends an alert when the number of used bytes in the WiredTiger cache meets the specified threshold.

**Cache: Bytes Read Into Cache is**

Sends an alert when the number of bytes read into the WiredTiger cache meets the specified threshold.

**Cache: Bytes Written From Cache is**

Sends an alert when the number of bytes written from the WiredTiger cache meets the specified threshold.

## Replica Set Alerts

These alert conditions apply 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**

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**

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 apply 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.

---

### **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 down**

Sends an alert when the 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.

### **Backup Agent is out of date**

Sends an alert when the Backup Agent is not running the latest version of the software.

### **Backup Agent has too many conf call failures**

Available only as a *global alert*.

Sends an alert when the cluster topology as known by monitoring does not match the backup configuration from conf calls made by the Backup Agent. Ops Manager sends the alert after the number of attempts specified in the `maximumFailedConfCalls` setting.

## Backup Alerts

These alert conditions apply to Ops Manager Backup.

### **Oplog Behind**

Sends an alert if the most recent *oplog* data received by Ops Manager is more than 75 minutes old.

To resolve this alert, see *Receive “Oplog Behind” alert*.

### **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, you must resync backup, as described in the procedure *Resync a Backup*.

Also, 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 backed-up replica set.
- The *oplog* for the backed-up replica set was resized or deleted.
- High oplog churn caused the agent to lose the tail of the oplog.

### **Cluster Mongos Is Missing**

Sends an alert if Ops Manager cannot reach a mongos for the cluster.

### **Bind Error**

Available only as a [global alert](#).

Sends an alert if a backup job fails to bind to a Backup Daemon. A job might fail to bind if, for example:

- No *primary* is found for the backed-up replica set. At the time the binding occurred, the Monitoring Agent did not detect a primary. Ensure that the replica set is healthy.
- Not enough space is available on any Backup Daemon.

In both cases, resolve the issue and then [restart the initial sync of the backup](#).

As an alternative, you can manually bind jobs to daemons through the *Admin* interface. See [Jobs Page](#) for more information.

### **Backup has reached a high number of retries**

Available only as a [global alert](#).

Sends an alert if the same task fails repeatedly. This could happen, for example, during maintenance. [Check the corresponding job log](#) for an error message explaining the problem. Contact MongoDB Support if you need help interpreting the error message.

### **Backup is in an unexpected state**

Available only as a [global alert](#).

Sends an alert when something unexpected happened and the Backup state for the replica set is broken. You must resync the backed-up replica set, as described in the [Resync a Backup](#) procedure.

In case of a [Backup is in an unexpected state](#) alert, [check the corresponding job log](#) for an error message explaining the problem. Contact MongoDB Support if you need help interpreting the error message.

### **Late Snapshot**

Available only as a [global alert](#).

Sends an alert if a snapshot has failed to complete before the next snapshot is scheduled to begin. [Check the job log](#) in the Ops Manager Admin interface for any obvious errors.

### **Bad Clustershot Count is**

Sends an alert if Ops Manager fails a consecutive number of times to successfully take a cluster snapshot. Ops Manager sends notification when the number exceeds the number specified in the alert configuration.

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.

### **Sync Slice Has Not Progressed in**

Available only as a [global alert](#).

Sends an alert when an [initial sync](#) has started but then subsequently stalled. A number of issues can cause this, including:

- processes that are down (agents, ingest, backing databases)
- network issues

- incorrect authentication credentials

#### **Job is busy for**

Available only as a *global alert*.

Sends an alert when a backup job has taken longer than the time specified. This could occur if you have an overloaded Backup Daemon or blockstore. *Check the corresponding job log* for error messages. Contact MongoDB Support if you need help interpreting the error message.

## User Alerts

These alert conditions apply 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 Alerts

These alert conditions apply to group membership, group security, and the group's subscription.

#### **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.

#### **Users do not have two factor authentication enabled**

Sends an alert if the group has users who have not set up *two-factor authentication*.

#### **Service suspended due to unpaid invoice(s) more than 30 days old**

Sends an alert if the group is suspended because of non-payment. A suspended group:

- denies users access,
- stops backups,
- terminates stored snapshots 90 days after suspension.

## 5.7 Global Alerts

### On this page

- Overview
- Procedures

### Overview

A global alert applies the same *alert configuration* to multiple groups at once. When an alert condition occurs, Ops Manager sends notification only to the affected group. Ops Manager sends notification at regular intervals until the alert *resolves* or gets *canceled*.

To access global alerts you must have the *Global Monitoring Admin* role or *Global Owner* role.

## Procedures

### View and Manage Global Alerts

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

**Step 2:** Select the *Alerts* tab and then select *Open Alerts* under *Global Alerts*.

Step 3: Select one or more filters above the list and click <i>Filter</i> .	
To filter by:	Do this:
Acknowledgement Group	Select the appropriate option in the <i>All States</i> drop-down list.
Alert type (open alerts only)	Type the group's name in the <i>Groups</i> box.
Date	Type the <i>alert type</i> in the <i>Types</i> box. The box autocompletes an alert type only if there is an open alert of that type.
	Select the dates in the <i>From</i> and <i>To</i> fields.

**Step 4: Acknowledge or unacknowledge an alert.** To acknowledge an alert, click *Acknowledge* on the line for the alert, select the time period for the acknowledgement, and click *Acknowledge*. Ops Manager sends no further notifications for the selected period.

To “undo” an acknowledgment and again receive notifications if the alert condition still applies, click *Unacknowledge* on the line for the alert and click *Confirm*.

### Configure a Global Alert

**Step 1: Click the *Admin* link in the top right corner of Ops Manager.** Ops Manager displays the *Admin* link only if you have administrative privileges.

**Step 2: Select the *Alerts* tab and then the *Global Alert Settings* page.**

**Step 3: Click the *Add Alert* button.** Ops Manager displays the alert configuration options.

**Step 4: Select whether the alert applies to all groups or specific groups.** If you select *These groups*, type the first few letters of each group you want to add and then select it from the drop-down list.

**Step 5: Select the condition that triggers the alert.** In the *Alert if* section, first select the target component. If you select *Host*, select the type of host as well.

Second, select the condition and, if applicable, the condition threshold. For explanations of alert conditions, see *Alert Conditions*.

**Step 6: If applicable, apply the alert to specific targets.** If the options in the *For* section are available, you can optionally filter the alert to apply only to a subset of the targets.

**Step 7: Select the alert recipients and delivery methods.** In the *Send to* section, configure notifications. To add notifications or recipients, click *Add* and select from the options listed below. To test a notification, click the test link that appears after you configure the notification.

Notification Option	Description
Group ( <i>group or global alerts only</i> )	Sends the alert by email or SMS to the group. If you select <i>SMS</i> , Ops Manager sends the text message to the number configured on each user's <a href="#">Account page</a> . To send only to specific roles, deselect <i>All Roles</i> and select the desired roles.
Ops Manager User	Sends the alert by email or SMS to a specified Ops Manager user. If you select <i>SMS</i> , Ops Manager sends the text message to the number configured on the user's <a href="#">Account page</a> .
SNMP Host	Specify the hostname that will receive the v2c trap on standard port 162. The MIB file for SNMP is available for <a href="#">download here</a> .
Email	Sends the alert to a specified email address.
HipChat	Sends the alert to a HipChat room message stream. Enter the HipChat room name and API token.
Slack	Sends the alert to a Slack channel. Enter the channel name and either an API token or a Bot token. To create an API token, see the <a href="https://api.slack.com/web">https://api.slack.com/web</a> page in your Slack account. For information on Bot users in Slack, see <a href="https://api.slack.com/bot-users">https://api.slack.com/bot-users</a> .
Flowdock	Sends the alert to a Flowdock account. Enter the following:
	<ul style="list-style-type: none"> <li>• <i>Org Name</i>: The Flowdock organization name in lower-case letters. This is the name that appears after <code>www.flowdock.com/app/</code> in the URL string.</li> <li>• <i>Flow Name</i>: The flow name in lower-case letters. The flow name appears after the org name in the URL string: <code>www.flowdock.com/app/orgname/flowname</code>. The flow name also appears in the “flow email address” setting in Flowdock. For example: <code>flowname@example.flowdock.com</code>.</li> <li>• <i>User API Token</i>: Your Flowdock “personal API token” found on the <a href="https://www.flowdock.com/account/tokens">https://www.flowdock.com/account/tokens</a> page of your Flowdock account.</li> </ul>
PagerDuty	Enter only the service key. Define escalation rules and alert assignments in <a href="#">PagerDuty</a> .
Webhook ( <i>group alerts only</i> )	Sends an HTTP POST request to an endpoint for programmatic processing. The request body contains a JSON document that uses the same format as the Public API's <a href="#">Alerts resource</a> . This option is available only if you have configured Webhook settings on the <a href="#">Group Settings</a> page.
Administrators ( <i>global or system alerts only</i> )	Sends the alert to the email address specified in the <i>Admin Email Address</i> field in the Ops Manager <a href="#">configuration options</a> .

**Step 8: Click Save.**

## 5.8 System Alerts

### On this page

- Overview
- System Alerts
- Procedures

### Overview

System alerts are internal health checks that monitor the health of Ops Manager itself, including the health of backing databases, Backup Daemons, and backed-up deployments. Ops Manager runs health checks every 5 minutes.

To view system alerts, click the *Admin* link at the top of Ops Manager, select the *Alerts* tab, and then select the *Open Alerts* link under *System Alerts*.

Users with the *Global Owner* or *Global Monitoring Admin* roles can *modify notification settings* or *disable* an alert.

### System Alerts

Ops Manager provides the following system alerts:

#### **System detects backup oplog TTL was resized**

Sends an alert if the Backup Daemon has fallen so far behind in applying *oplog* entries that Ops Manager has extended the period of time it will store the entries. By default, Ops Manager stores oplog entries in the *Oplog Store* for 24 hours. If the Daemon has not yet applied an entry an hour before its expiration, Ops Manager extends the storage period by three hours. If the entry again reaches an hour from expiration, Ops Manager continues to extend the storage period, up to seven days. The system sets the storage period in the Daemon's `mms.backup.restore.snapshotPITEExpirationHours` setting.

If you receive this alert, check that your Backup Daemon is running and that it has sufficiently performant hardware to apply oplog entries in a timely manner.

#### **System detects backing database startup warnings**

Sends an alert if the MongoDB process hosting a *Backing Database* contains `startupWarnings` in its log files. Check the logs on the server running the MongoDB process.

#### **System detects an unhealthy database backing the system**

Sends an alert if Ops Manager cannot connect to a backing database and run the ping command.

#### **System detects backup daemon is down**

Sends an alert if the Backup Daemon has not pinged Ops Manager for more than 15 minutes.

#### **System detects backup daemon has low free head space**

Sends an alert if the disk partition on which the local copy of a backed-up replica set is stored has less than 10 GB of free space remaining.

The Ops Manager *Daemons Page* displays head space used for each daemon. The `mms.alerts.LowHeadFreeSpace.minimumHeadFreeSpaceGB` setting controls the alert threshold, which has a default value of 10 GB.

#### **System detects backup was not moved successfully**

Sends an alert if you attempt to move a job to a new Backup Daemon but the move failed. The job will continue to run in its original location. For more information on moving jobs, see *Jobs Page*.

## Procedures

### Modify Notification Settings for a System Alert

**Step 1: Click the *Admin* link in the top right corner of Ops Manager.** Ops Manager displays the *Admin* link only if you have administrative privileges.

**Step 2: Select the *Alerts* tab and then the *System Alert Settings* page.**

**Step 3: On the line for the alert, click the ellipsis icon and select *Edit*.**

**Step 4: Select the alert recipients and delivery methods.** In the *Send to* section, configure notifications. To add notifications or recipients, click *Add* and select from the options listed below. To test a notification, click the test link that appears after you configure the notification and ensure that the service you are testing receives the message.

Notification Option	Description
Group ( <i>group or global alerts only</i> )	Sends the alert by email or SMS to the group. If you select <i>SMS</i> , Ops Manager sends the text message to the number configured on each user's <a href="#">Account page</a> . To send only to specific roles, deselect <i>All Roles</i> and select the desired roles.
Ops Manager User	Sends the alert by email or SMS to a specified Ops Manager user. If you select <i>SMS</i> , Ops Manager sends the text message to the number configured on the user's <a href="#">Account page</a> .
SNMP Host	Specify the hostname that will receive the v2c trap on standard port 162. The MIB file for SNMP is available for <a href="#">download here</a> .
Email	Sends the alert to a specified email address.
HipChat	Sends the alert to a HipChat room message stream. Enter the HipChat room name and API token.
Slack	Sends the alert to a Slack channel. Enter the channel name and either an API token or a Bot token. To create an API token, see the <a href="https://api.slack.com/web">https://api.slack.com/web</a> page in your Slack account. For information on Bot users in Slack, see <a href="https://api.slack.com/bot-users">https://api.slack.com/bot-users</a> .
Flowdock	Sends the alert to a Flowdock account. Enter the following:
	<ul style="list-style-type: none"> <li>• <i>Org Name</i>: The Flowdock organization name in lower-case letters. This is the name that appears after <code>www.flowdock.com/app/</code> in the URL string.</li> <li>• <i>Flow Name</i>: The flow name in lower-case letters. The flow name appears after the org name in the URL string: <code>www.flowdock.com/app/orgname/flowname</code>. The flow name also appears in the “flow email address” setting in Flowdock. For example: <code>flowname@example.flowdock.com</code>.</li> <li>• <i>User API Token</i>: Your Flowdock “personal API token” found on the <a href="https://www.flowdock.com/account/tokens">https://www.flowdock.com/account/tokens</a> page of your Flowdock account.</li> </ul>
PagerDuty	Enter only the service key. Define escalation rules and alert assignments in <a href="#">PagerDuty</a> .
Webhook ( <i>group alerts only</i> )	Sends an HTTP POST request to an endpoint for programmatic processing. The request body contains a JSON document that uses the same format as the Public API's <a href="#">Alerts resource</a> . This option is available only if you have configured Webhook settings on the <a href="#">Group Settings</a> page.
Administrators ( <i>global or system alerts only</i> )	Sends the alert to the email address specified in the <i>Admin Email Address</i> field in the Ops Manager <a href="#">configuration options</a> .

**Step 5: Click Save.**

## Disable a System Alert

**Step 1: Click the *Admin* link in the top right corner of Ops Manager.** Ops Manager displays the *Admin* link only if you have administrative privileges.

**Step 2: Select the *Alerts* tab and then the *System Alert Settings* page.**

**Step 3: Disable the alert.** On the line for the system alert that you want to disable, click the ellipsis icon and select *Disable*.

# 6 Back Up and Restore Deployments

***Back Up Deployments*** Describes how Backup works and provides instructions for backing up a deployment.

***Manage Backups*** Configure a deployment's backup settings; stop, start, and resync backups.

***Restore Deployments*** Describes how restores work and provides instructions for restoring a backup.

## 6.1 Back up MongoDB Deployments

This section describes how Backup works and how to create a backup.

***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.

***Back up a Deployment*** Activate the Backup feature for a cluster or replica set.

### Backup Flows

#### On this page

- [Introduction](#)
- [Snapshot Storage](#)
- [Backup States](#)
- [Initial Sync](#)
- [Routine Operation](#)
- [Restores](#)
- [Grooms](#)

#### Introduction

Ops Manager backups, once started, are an ongoing and continuous process. Data is continually backed up as long as the backup remains synchronized with the database. This process works like the process used to replicate data in a *replica set* to secondary. The Backup process:

1. Performs an *initial sync* to back up all of the data in its current state.
2. Takes scheduled snapshots to keep the database backup current.

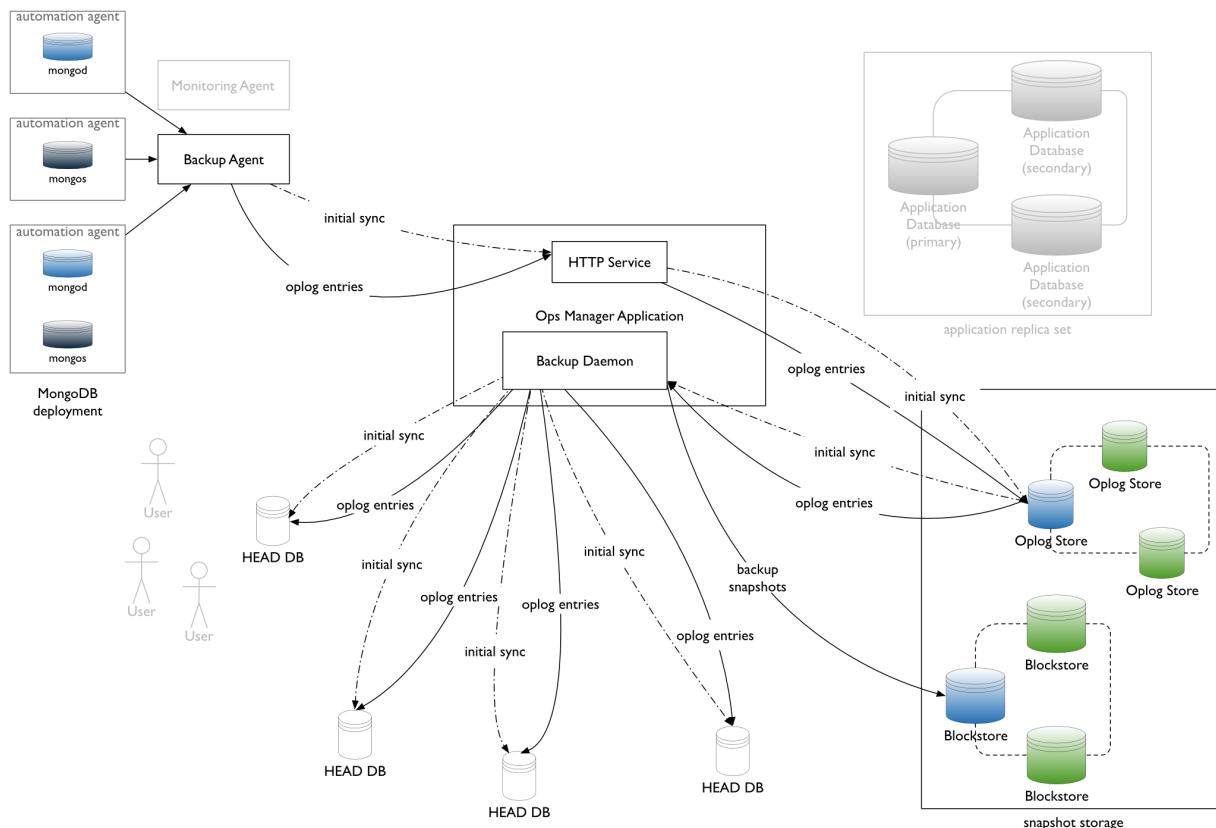
3. Constantly monitors the *oplog* and adds its operations to the latest backup to keep its copy of the data current.

The backup process works in this manner using either database-based (blockstore) or file-based (file system) backup storage.

## Snapshot Storage

**Snapshots** The backup process takes a snapshot of the data directory of the backed-up deployment as often as specified by the *snapshot schedule*. This snapshot is transferred to snapshot storage.

For a sharded cluster, the backup process takes a snapshot of each shard and of the config servers. The backup process can also use *checkpoints* to permit restores at moments between snapshots. You must first *enable checkpoints*.

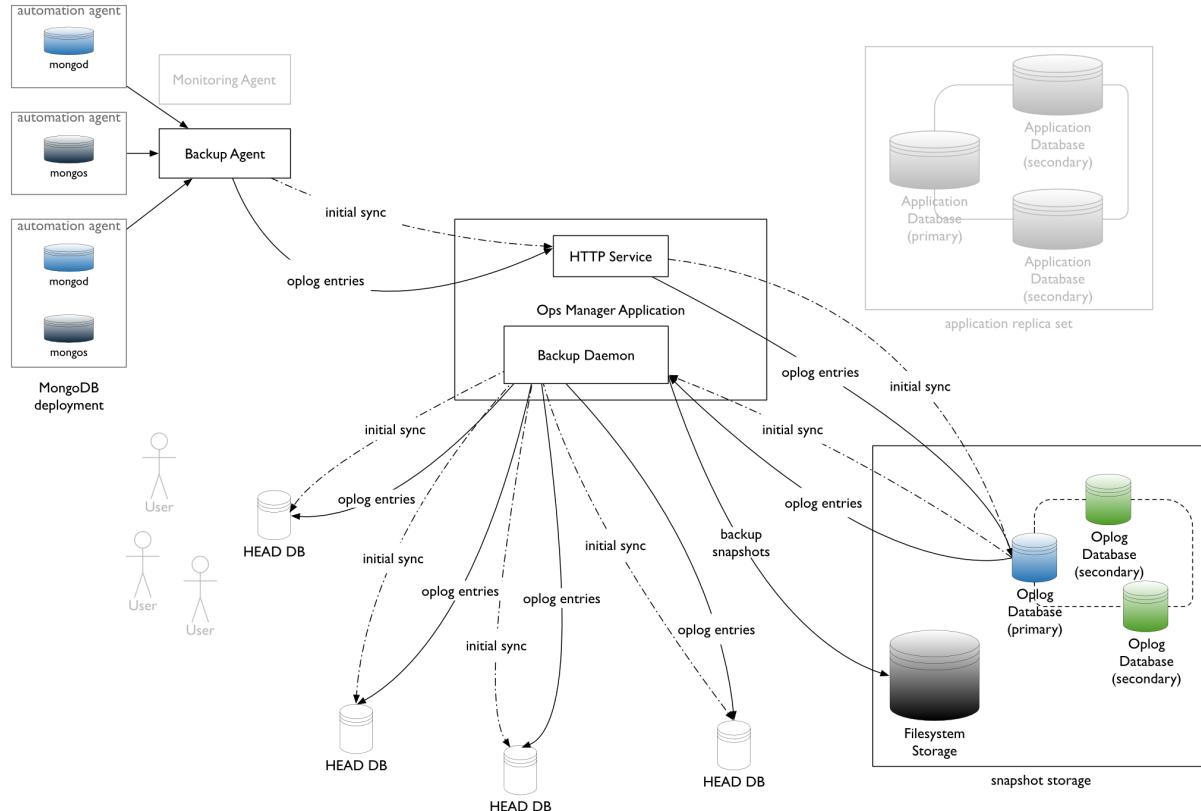


## Blockstores

**File Systems** File System storage gives you full control over your data, allowing you to reduce the number of MongoDB instances you need to run and enabling you to take advantage of existing storage infrastructure. With filesystem storage, you can access the backed-up data files using standard filesystem utilities, rather than having to retrieve them through the Ops Manager interface.

With filesystem storage, you can select any mountable filesystem to which Ops Manager writes the snapshots.

If you use file system snapshot storage and use multiple Ops Manager instances (including those activated as Backup Daemons), all Ops Manager instances must have the same view of the file system snapshot storage. You can achieve this through a Network File System (NFS) application or something similar.



**Important:** If the Ops Manager instances do not share the same view of file system snapshot storage, Backup restores will not be possible and Ops Manager will not be able to remove expired snapshots.

**Operation Logs** When a new Blockstore is added, that Blockstore integrates an *oplog store* into its database.

When a new file system store is created, you are prompted to create an *oplog store*.

Once an oplog has been created for a snapshot store, adding any additional snapshot stores does not require another *oplog*.

## Backup States

A backup job defines how much and how often data is backed up. There are three states for a backup job: **active**, **inactive** or **stopped**.

State	Retain Old Snapshots	Create New Snapshots	Apply Oplogs
Active	Yes	Yes	Yes
Stopped	Yes	No	No
Inactive	No	No	No

Once backups are active for a Group, they run without further intervention until they are stopped or terminated. The operator can change the state of a backup in the following ways:

Initial State	Desired State	Method
Active	<i>Active after Initial Sync</i>	Click <i>Start</i> .
Active	<i>Stopped</i>	Click <i>Stop</i> .
Stopped	<i>Active after Initial Sync</i>	Click <i>Restart</i> .
Stopped	<i>Inactive</i>	Click <i>Terminate</i> .
<b>Warning:</b> <i>Terminate</i> deletes all retained backups.		

**Important:** You may receive a [Resync Required](#) alert for your backup jobs. This may require you to [Resync a Backup](#). This is not a different state, but a triggering of a new [Initial Sync](#). The backup job, once Initial Sync completes, becomes *Active* again.

---

## Initial Sync

Initial Sync is an interim state between *Inactive* and *Active*. Its activities are exposed to the operator to keep them informed that the process is in progress.

**Transfer of Data and Oplog Entries** When you start a backup, the Backup Agent streams your deployment's existing data to Ops Manager in batches of documents, totaling roughly 10 MB each, called **slices**. Slices contain only the data as it existed when you started the backup.

While transferring the data, the Backup Agent tails the [oplog](#) and streams the oplog updates to Ops Manager. Ops Manager then stores the oplog entries in the oplog database for later processing offline.

**Building the Backup** When Ops Manager has received all of the slices, it creates a local database on its server and inserts the documents that were captured as slices during the initial sync. Ops Manager then applies the oplog entries from the oplog store.

Ops Manager then validates the data. If there are missing documents, Ops Manager queries the deployment for the documents and inserts them. A missing document could occur because of an update that caused a document to move during the initial sync.

Once Ops Manager validates the accuracy of the data directory, it has completed the initial sync process and proceeds to routine operation.

**Initial Sync Phases** Backups go through a series of phases during backup. The first backup is called the initial synchronization (`initialSync`). If you click the *Backup* button, you are presented with the list of backup jobs in progress. During `initialSync`, they proceed through these phases in the following order:

Phase	Description	Measure of Progress
starting	Waiting phase until Ops Manager receives sync slices.	None
transferring	Shows progress one namespace at a time.	<ul style="list-style-type: none"> <li>A ratio of ingested documents to total documents.</li> <li>A percentage of documents ingested.</li> </ul>
building	<p>Ops Manager has started building the deployment HEAD database.</p> <p><b>Note:</b> As of Backup Agent 4.0 and Ops Manager 2.0, this happens in the background during the transferring phase, but the UI shows the transferring phase until all slices are received.</p>	The percentage of the entire HEAD database that has been built.
applying oplogs	Ops Manager has completed building the HEAD database. Ops Manager is applying the oplog entries to catch up with the customer's oplog at the point that the previous phase completed.	The percentage of slices currently applied.
fetching missing documents	Ops Manager queries the customers deployment for documents missed during the building and applying oplogs phases.	The percentage of slices currently applied.
creating indexes	Ops Manager creates all the indexes of the customer's deployment.	<ul style="list-style-type: none"> <li>The index currently being processed.</li> <li>A percentage of total indexes processed so far.</li> </ul>
complete	The backup initialSync job has completed.	

## Routine Operation

The Backup Agent tails the deployment's oplog and routinely batches and transfers new oplog entries to the HTTP Service, which stores them in the oplog store. The Backup process applies all newly-received oplog entries in batches to its local copy of the backed-up deployment.

## Restores

When a user requests a snapshot, a backup process retrieves the data from the snapshot storage and delivers it to the requested destination.

### See also:

[Restore Overview](#) for an overview of the restore process.

## Grooms

Groom jobs perform periodic “garbage collection” on the snapshot storage to reclaim space from deleted snapshots. A scheduling process determines when grooms are necessary.

## Backup Preparations

### On this page

- [Overview](#)
- [Snapshot Frequency and Retention Policy](#)
- [Namespaces Filter](#)
- [Storage Engine](#)
- [Resyncing Production Deployments](#)
- [Checkpoints](#)
- [Snapshots when Agent Cannot Stop Balancer](#)
- [Snapshots when Agent Cannot Contact a mongod](#)

### 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 a Snapshot](#).

### Namespaces Filter

The namespaces filter lets you specify which databases and collections to back up. You create either a *Blacklist* of those to exclude or a *Whitelist* of those to include. You make your selections when starting a backup and can later edit them as needed. If you change the filter in a way that adds data to your backup, a resync is required.

Use the blacklist to prevent backup of 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. Using a blacklist is often preferable to using a whitelist as a whitelist requires you to intentionally opt in to every namespace you want backed up.

---

**Important:** If you use the namespaces filter, your backup's restore data will not include the `seedSecondary` script. The `seedSecondary` script provides an alternative to initial sync on new or restored replica set members. For more information, see [Seed a New Secondary from Backup Restore](#).

---

## 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. The WiredTiger encryption option is not available for storing backups. 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.

---

**Important:** WiredTiger's encryption option is not available for backups. You can store backups using the WiredTiger storage engine, but you cannot enable the encryption option. If you restore from a backup, you restore unencrypted files.

---

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 initial 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.

You may also want to resync your backup after:

- A reduction in data size, such that the size on disk of Ops Manager's copy of the data is also reduced. This scenario also includes if you:
  - Have a [TTL index](#) in place, which periodically deletes documents.
  - [Drop a collection](#) (MMAPv1 only).

- Run a *sharded cluster*, and there have been a lot of chunks moved off a particular shard.
- A switch in storage engines, if you want Ops Manager to provide snapshots in the new storage engine format.
- A manual build of an index on a replica set in a rolling fashion (as per [Build Indexes on Replica Sets](#) in the MongoDB manual).

## Checkpoints

For *sharded clusters*, checkpoints provide additional restore points between snapshots. With checkpoints enabled, Ops Manager creates restoration points at configurable intervals of every 15, 30 or 60 minutes between snapshots. To enable checkpoints, see [enable checkpoints](#).

To create a checkpoint, Ops Manager 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 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` process, 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.

## Back up a Deployment

### On this page

- [Overview](#)
- [Unique Names for Deployment Items](#)
- [Prerequisites](#)
- [Procedure](#)

### Overview

You can choose to back up all databases and collections on the deployment or specific ones.

---

**Important:** Only sharded clusters or replica sets can be backed up. To back up a standalone `mongod` process, you must first [convert it to a single-member replica set](#).

---

## Unique Names for Deployment Items

Ensure your deployment items have unique names before creating backups.

---

**Important:** Replica set, sharded cluster, and shard names within the same group must be unique. Failure to have unique names for the deployment items will result in broken backup snapshots.

---

## Prerequisites

- For a Replica Set:
  - Ops Manager must be monitoring the deployment.
  - It must run MongoDB version 2.2.0 or later.
  - It must have an active *primary* node.
- For a Sharded Cluster:
  - Ops Manager must be monitoring the deployment, including at least one `mongos` in the cluster.
  - It must run MongoDB version 2.4.3 or later.
  - All *config servers* must be running. The config server mongod processes must be started with either the `--configsvr` command line option or the `{ "clusterRole": "configsvr" }` setting in the mongod configuration file.
  - The balancing round must be able to complete in less than one hour.
- The MongoDB version and Ops Manager version must meet the *compatibility requirements*.
- Decide how to back up the data and what data to back up. See the *Backup Preparations*.

## Procedure

**Step 1: Click the *Backup* tab.** If you have not yet enabled Ops Manager Backup, click *Begin Setup* and complete the wizard. This results in a completed backup setup, so you can skip the rest of this procedure.

**Step 2: Start backing up the process.** From the list of processes, navigate to the *Status* column for the process you want to back up and click *Start*.

**Step 3: In the *Start Backup* sidebar, configure the backup source and storage engine.** Select values from these two drop-down menus:

Menu	Possible and Default Values
Sync source	This can be <i>any</i> secondary (Ops Manager chooses) or any specific secondary or the primary node. The default value ( <code>any secondary</code> ) is preferred because it minimizes performance impact on the primary.
Storage Engine	This can be either <i>MongoDB Memory Mapped Files</i> or <i>WiredTiger</i> , but the default value is the same storage engine as the primary node of the database being backed up. See the considerations in <i>Storage Engines</i> .

**Step 4: If the deployment is not under Automation and requires authentication, specify the authentication mechanism and credentials.** Specify the following, as appropriate:

<i>Auth Mechanism</i>	The <a href="#">authentication mechanism</a> the host uses. The options are: <ul style="list-style-type: none"><li>• <a href="#">Username/Password</a></li><li>• <a href="#">Kerberos</a> (Enterprise Only)</li><li>• <a href="#">LDAP</a> (Enterprise Only)</li><li>• <a href="#">X.509 Client Certificate</a></li></ul>
<i>DB Username</i>	For Username/Password or LDAP authentication, the username used to authenticate the Backup Agent to the MongoDB deployment. See <a href="#">Configure Backup Agent for MONGODB-CR</a> or <a href="#">Configure Backup Agent for LDAP Authentication</a> .
<i>DB Password</i>	For Username/Password or LDAP authentication, the password used to authenticate the Backup Agent to the MongoDB deployment.
<i>Allows SSL for connections</i>	If checked, the Backup Agent uses SSL to connect to MongoDB. See <a href="#">Configure Backup Agent for SSL</a> .

**Step 5: To filter which namespaces get backed up, click the triangle to the right of *Advanced Settings*.** To exclude databases and collections from this backup:

1. Click *Blacklist*.
2. Enter the first database and collection in the text box. For collections, enter the full namespace: <database>.<collection>.
3. To exclude additional databases or collections, click the *Add another* link then repeat the previous step.

To include only certain databases and collections for this backup:

1. Click *Whitelist*.
2. Enter the first database and collection in the text box. For collections, enter the full namespace: <database>.<collection>.
3. To include additional databases or collections, click the *Add another* link then repeat the previous step.

**Step 6: Click *Start*.**

## 6.2 Manage Backups

[\*\*Edit a Backup's Settings\*\*](#) Modify a backup's schedule, storage engines, and namespaces filter.

[\*\*Stop, Restart, or Terminate a Backup\*\*](#) Stop, restart, or terminate a deployment's backups.

[\*\*View a Backup's Snapshots\*\*](#) View a deployment's available snapshots.

[\*\*Delete a Snapshot\*\*](#) Manually remove unneeded stored snapshots from Ops Manager.

[\*\*Resync a Backup\*\*](#) If your Backup oplog has fallen too far behind your deployment to catch up, you must resync Backup.

[\*\*Generate a Key Pair for SCP Restores\*\*](#) Generate a key pair for SCP restores

[\*\*Disable the Backup Service\*\*](#) Disable Ops Manager Backup.

## Edit a Backup's Settings

### On this page

- Overview
- Procedure

### Overview

You can modify a backup's *schedule*, *namespaces filter*, 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** Cluster *checkpoints* provide restore points in between scheduled snapshots. You can use checkpoints to create custom snapshots at points in time between regular snapshots.

**Step 1: Select the *Backup* tab and then *Overview* filter.**

**Step 2: Select the backup's ellipsis icon and select *Edit Snapshot Schedule*.** On the line listing the process, click the ellipsis icon and click *Edit Snapshot Schedule*.

**Step 3: Enable cluster checkpoints.** Select *Create cluster checkpoint every* and set the interval. Then click *Submit*.

### Edit Snapshot Schedule and Retention Policy

**Step 1: Click the *Backup* tab.**

**Step 2: On the row for the process, click the backup's ellipsis icon, then click *Edit Snapshot Schedule*.**

**Step 3: Configure the snapshot settings.**

1. Enter the following information as needed.

**Note:** See *Snapshot Frequency and Retention Policy* for default and maximum retention values as well as how these settings affect Ops Manager.

<p><i>Take snapshots every ... hours and save for ... days</i></p> <p><i>Create cluster checkpoint every ... minutes (Sharded Clusters only)</i></p> <p><i>Store daily snapshots for</i></p> <p><i>Store weekly snapshots for</i></p> <p><i>Store monthly snapshots for</i></p> <p><i>Allow point-in-time restores going back</i></p>	<p>Sets how frequently, in hours, Ops Manager takes a base snapshot of the deployment and the number of days Ops Manager retains base snapshots.</p> <p>Sets how frequently, in minutes, Ops Manager creates a <i>checkpoint</i> in between snapshots of a sharded cluster. Checkpoints provide restore points that you can use to create custom <i>point-in-time</i> snapshots.</p> <p>See <a href="#">Checkpoints</a>.</p> <p>Sets the number of days that Ops Manager retains daily snapshots.</p> <p>Sets the number of weeks that Ops Manager retains weekly snapshots.</p> <p>Sets the number of months that Ops Manager retains monthly snapshots.</p> <p>Sets the number of days that Ops Manager retains oplogs alongside snapshots.</p> <p>See <a href="#">Restore Overview</a> for how snapshots and point-in-time restores work.</p>
---	--

2. Click *Submit*.

## Edit Security Credentials

**Step 1:** Select the *Backup* tab and then the *Overview* filter.

**Step 2:** Select the backup's ellipsis icon and select *Edit Credentials* On the line listing the process, click the ellipsis icon and click *Edit Credentials*.

**Step 3:** Provide the authentication information. Enter the following information as needed and click *Submit*.

<i>Auth Mechanism</i>	The <a href="#">authentication mechanism</a> the host uses. The options are: <ul style="list-style-type: none"><li>• <a href="#">Username/Password</a></li><li>• <a href="#">Kerberos</a> (Enterprise Only)</li><li>• <a href="#">LDAP</a> (Enterprise Only)</li><li>• <a href="#">X.509 Client Certificate</a></li></ul>
<i>DB Username</i>	For Username/Password or LDAP authentication, the username used to authenticate the Backup Agent to the MongoDB deployment. See <a href="#">Configure Backup Agent for MONGODB-CR</a> or <a href="#">Configure Backup Agent for LDAP Authentication</a> .
<i>DB Password</i>	For Username/Password or LDAP authentication, the password used to authenticate the Backup Agent to the MongoDB deployment.
<i>Allows SSL for connections</i>	If checked, the Backup Agent uses SSL to connect to MongoDB. See <a href="#">Configure Backup Agent for SSL</a> .

**Edit the Namespaces Filter** The [Namespaces Filter](#) specifies which databases and collections Ops Manager backs up.

**Step 1:** Select the *Backup* tab and then the *Overview* filter.

**Step 2:** Select the backup's ellipsis icon and select *Edit Namespaces Filter*.

**Step 3: Add or remove namespaces and click *Submit*.**

### Modify the Storage Engine Used for Backups

**Step 1:** Select the *Backup* tab and then the *Overview* filter.

**Step 2: Select the backup's ellipsis icon and select *Edit Storage Engine*** On the line listing the process, click the ellipsis icon and click *Edit Storage Engine*.

**Step 3: Select the storage engine.** Select the storage engine. See: *Storage Engine* for more about choosing an appropriate storage engine for your backup.

**Step 4: 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 5: Click *Submit*.**

### Stop, Restart, or Terminate a Backup

#### On this page

- Overview
- Stop Backup for a Deployment
- Restart Backup for a Deployment
- Terminate a Deployment's Backups

#### Overview

When you stop backups for a replica set or sharded cluster Ops Manager stops taking new snapshots but retains existing snapshots until their listed expiration date.

If you later restart backups for replica set or cluster, Ops Manager might perform an *initial sync*, depending on how much time has elapsed.

If you **terminate** a backup, Ops Manager immediately **deletes all the backup's snapshots**.

#### Stop Backup for a Deployment

**Step 1:** Select the *Backup* tab and then *Overview* page.

**Step 2:** On the line listing the process, click the ellipsis icon and click *Stop*.

**Step 3: Click the *Stop* button.** If prompted for an authentication code, enter the code and click *Verify*. Click *Stop* again.

## **Restart Backup for a Deployment**

**Step 1:** Select the *Backup* tab and then *Overview* page.

**Step 2:** On the line listing the process, click the ellipsis icon and click *Restart*.

**Step 3:** Select a *Sync source* and click *Restart*.

## **Terminate a Deployment's Backups**

**Warning:** If you terminate a backup, Ops Manager immediately deletes the backup's snapshots.

**Step 1:** Select the *Backup* tab and then *Overview* page.

**Step 2:** Click the backup's ellipsis icon and click *Stop*.

**Step 3:** Click the *Stop* button. If prompted for an authentication code, enter the code and click *Verify*. Click *Stop* again.

**Step 4:** Once the backup has stopped, click the backup's ellipsis icon and click *Terminate*.

**Warning:** If you terminate a backup, Ops Manager immediately deletes the backup's snapshots.

**Step 5:** Click the *Terminate* button.

## **View a Backup's Snapshots**

### **On this page**

- [Overview](#)
- [Procedure](#)

### **Overview**

By default, Ops Manager takes a base snapshot of a backed-up deployment every 6 hours and retains snapshots as described in [Snapshot Frequency and Retention Policy](#). Administrators can change the frequency and retention settings.

You can view all available snapshots, as described here.

### **Procedure**

**Step 1:** Select the *Backup* tab and then *Overview* page.

**Step 2: On the line listing the process, click the ellipsis icon and click *View All Snapshots*.**

## Delete a Snapshot

### On this page

- [Overview](#)
- [Procedure](#)

### Overview

To delete snapshots for replica sets and sharded clusters, use the Ops Manager console to find then select a backup snapshot to delete.

---

**Important:** Do not delete a snapshot if you might need it for a point-in-time restore. Point-in-time restores apply oplog entries to the last snapshot taken before the specified point.

---

### Procedure

**Step 1: Select the *Backup* tab and then *Overview* page.**

**Step 2: On the line listing the process, click the ellipsis icon and click *View All Snapshots*.**

**Step 3: Select backup file to delete.** On the list of snapshots, click the *Delete* link to the right of a snapshot.

**Step 4: Confirm deletion.** Click the OK button on the *Delete Snapshot* interface to confirm deletion of the backup.

## Resync a Backup

### On this page

- [Overview](#)
- [Considerations](#)
- [Procedure](#)

### Overview

When a backup becomes out of sync with the MongoDB deployment, Ops Manager produces a [Resync Required](#) alert. If you receive this alert, you must resync the backup for the specified MongoDB instance.

The following scenarios trigger a Resync Required alert:

**The Oplog has rolled over** This is the most common case for the [Resync Required](#) alert. It 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.

---

**Important:** Ops Manager does not attempt to automatically recover from conditions that caused the [Resync Required](#) alert. This alert means there is not enough data to complete a restore. There is no way to automatically recover from not having enough data from the snapshots and the oplog. Resyncing the backup is the best option.

---

## 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.
- 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.

### See also:

For more information on the Backup oplog, see the [Backup FAQs](#).

## Procedure

**Step 1: Select the *Backup* tab and then *Overview* page.**

**Step 2: On the line listing the process, click the ellipsis icon and click *Resync*.**

**Step 3: Click the *Resync* button.** If prompted for an authentication code, enter the code and click *Verify*. Click *Resync* again.

## Generate a Key Pair for SCP Restores

### On this page

- [Overview](#)
- [Procedure](#)

## Overview

When you restore a snapshot, you can choose to access the files through a download link, or have Ops Manager copy the restore files via SCP to your server. To use SCP, you must generate a key pair that Ops Manager can use to transmit the restore.

---

**Note:** Windows machines do not come with SCP and require additional setup outside the scope of this manual.

---

## Procedure

**Step 1:** Select the *Settings* 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. Ops Manager 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.

---

## Disable the Backup Service

### On this page

- [Overview](#)
- [Procedure](#)

## Overview

Disabling Ops Manager Backup immediately deletes all snapshots. Later, if you want re-enable Backup, Ops Manager behaves as if your deployments had never been backed up and requires an *initial sync*.

## Procedure

**Step 1: Stop and then terminate each deployment enrolled in Backup.** In Ops Manager, click the *Backup* tab. For each deployment enrolled in Backup:

1. Click the backup's ellipsis icon and click *Stop*.
2. Click the *Stop* button. If prompted for an authentication code, enter the code.

1. Once the backup has stopped, click the backup's ellipsis icon and click *Terminate*.
2. Click the *Terminate* button.

**Step 2: Stop all Backup Agents.** See [Start or Stop the Backup Agent](#).

## 6.3 Restore MongoDB Deployments

Use these procedures to restore a MongoDB deployment using Backup artifacts.

[Restore Overview](#) Overview of the different restore types.

[Restore Flows](#) Overview of how the different restore types operate internally.

[Restore Sharded Cluster](#) Restore a sharded cluster from a stored snapshot or custom point-in-time snapshot based on a cluster checkpoint.

[Restore Replica Set](#) Restore a replica set from a stored snapshot or custom point-in-time snapshot.

[Restore a Single Database](#) Restore only a portion of a backup to a new mongod process.

[Seed a New Secondary from a Backup](#) Use Ops Manager Backup to seed a new secondary in an existing replica set.

### Restore Overview

#### On this page

- [Introduction](#)
- [Automated Restore](#)
- [Manual Restore](#)

#### Introduction

When you restore a deployment from a backup, you select a restore point and Ops Manager provides you with one or more restore files. If you use Ops Manager automation, Ops Manager can automatically restore the files for you. Otherwise, you manually restore the files by copying them yourself to the servers you choose.

You can restore a single MongoDB process or an entire *replica set* or *sharded cluster*. You select whether to restore from an existing snapshot or a restore point between snapshots. For the latter, Ops Manager builds a custom snapshot based on the restore point you choose. Installing from a custom snapshot takes longer as Ops Manager must build the snapshot by applying the [oplog](#) to the previous regular snapshot. For sharded clusters, you must [enable checkpoints](#) before you can restore to a point between snapshots.

You can restore to either new hardware or existing hardware. If you use existing hardware and perform the restore manually, create a new path for the MongoDB data directory. Do not use existing data directories.

#### Automated Restore

If you choose to have Ops Manager automatically restore your backup, Ops Manager clears out all existing data for the chosen servers and replaces them with the data from your backup. For an automated restore, only the system restoring and receiving the backup data needs to have Automation Agents installed.

**Important:** Under certain conditions, an automated restore fails because either:

1. The backup and destination databases' storage engines do not match.
2. The destination database uses settings that are not currently preserved in the backup itself.

If the backup and destination database storage engines do not match, mongod cannot start once the backup is restored. At this time, you can either:

1. Not restore this snapshot.
2. Change the storage engine of the destination sharded cluster or replica set to match the configuration of the snapshot.

If the destination database uses any of the following settings, an automated restore fails:

- `storage.directoryPerDB`
- `security.enableEncryption` (WiredTiger only)
- `storage.wiredTiger.engineConfig.directoryForIndexes` (WiredTiger only)

## Manual Restore

For a manual restore, Ops Manager provides each restore file as `.tar.gz` file containing a complete copy of the process's data directory. For a replica set, Ops Manager provides a single `.tar.gz` file that you copy to each replica set member. For a sharded cluster, Ops Manager provides one `.tar.gz` file for the *config servers* and separate `.tar.gz` files for each *shard*.

Ops Manager delivers the files through either HTTP or SCP. You select whether Ops Manager provides a download link for the restore files or uses SCP to copy the files directly to your system. The SCP method requires you to *generate a key pair* ahead of time but provides faster delivery. Windows machines do not come with SCP and require additional setup outside the scope of this manual.

For SCP delivery, you can choose to receive individual data files instead of `tar.gz` files. Ops Manager transmits the data files directly to the target directory. The destination server must have sufficient space for the data files.

If you choose to receive `tar.gz` files, you must extract each file before reconstructing your data. This is generally faster than the data-files option but requires temporary space on the server hosting the Backup Daemon and sufficient space on the destination server for both the archive file and the extracted contents.

## Restore Flows

### On this page

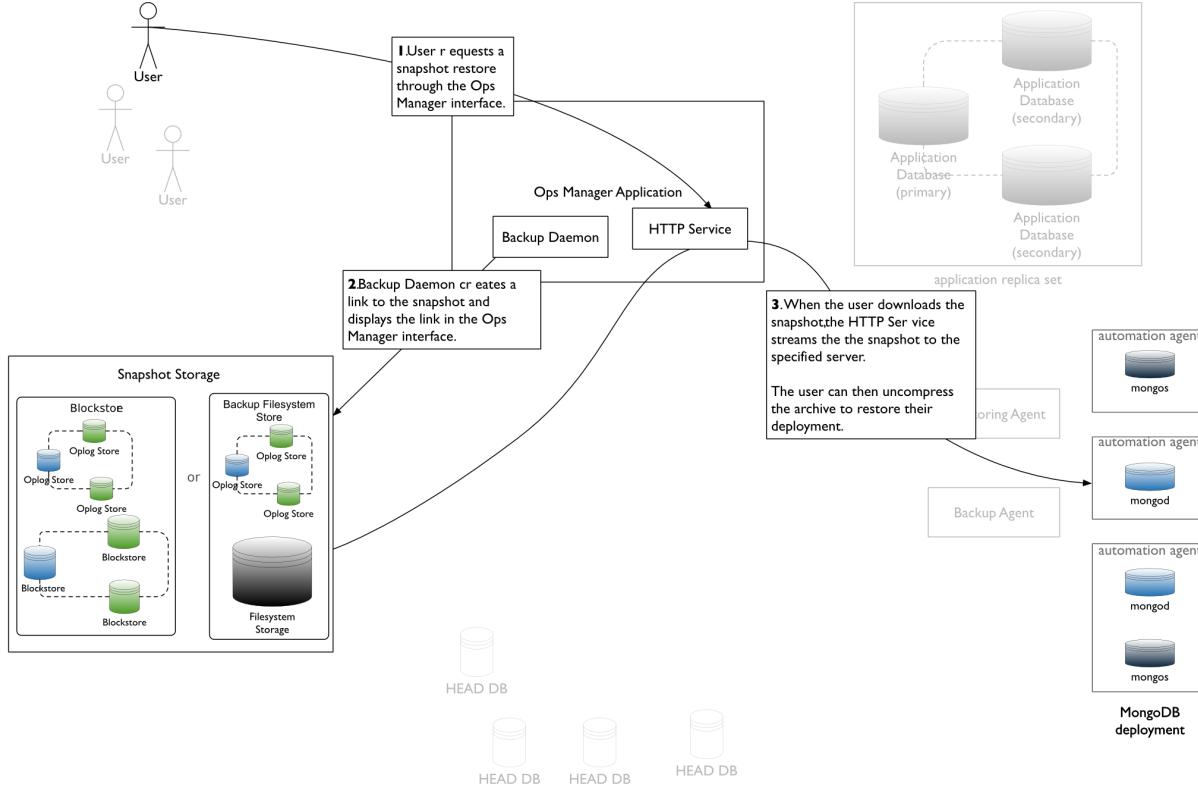
- [Introduction](#)
- [HTTP Restore](#)
- [Archive SCP Restore](#)
- [Individual Files SCP Restore](#)

### Introduction

When you request a restore, the Backup Daemon prepares the snapshot you will receive. If you choose to restore manually, you either download the files from the HTTP Service or the Backup Daemon securely copies the files to the destination server.

The following sections describe the flows for manual restores of scheduled snapshots and custom snapshots according to delivery option.

## HTTP Restore



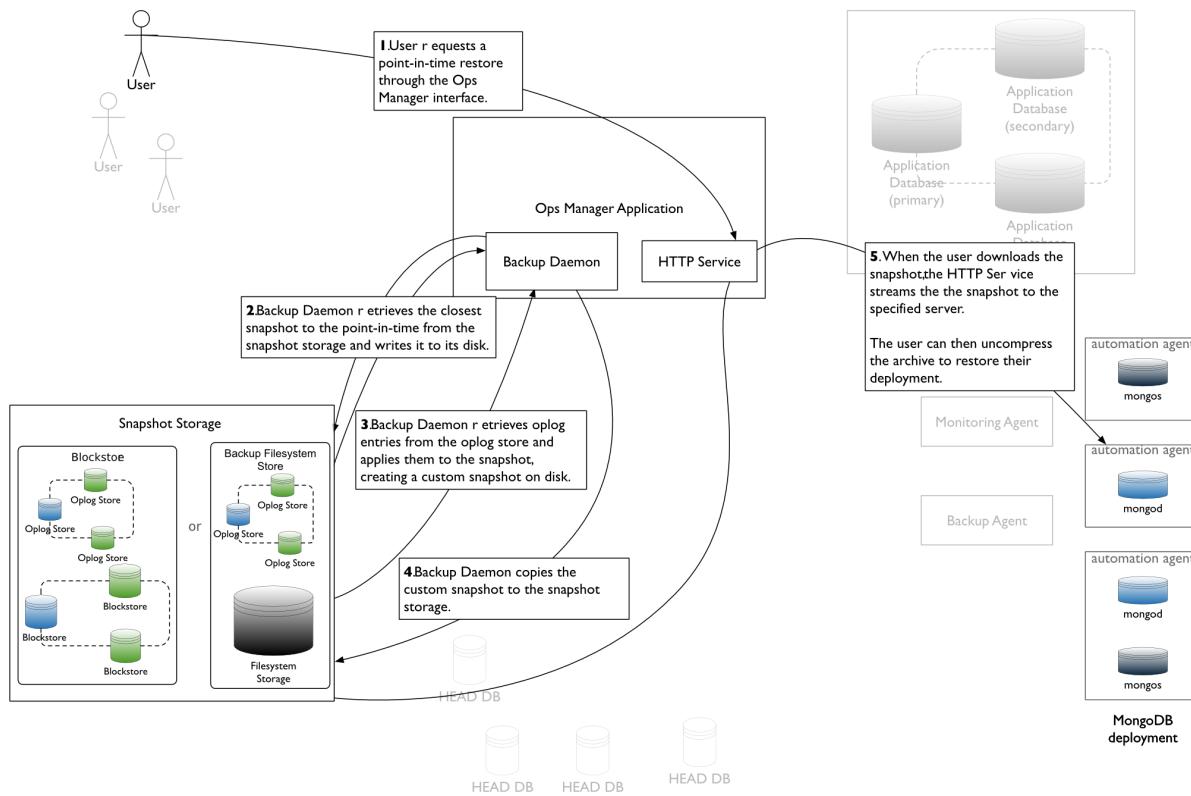
**Snapshot** With the HTTP PULL snapshot restore, the Backup Daemon creates a link to the appropriate snapshot in the snapshot storage. When the user clicks the download link, they download the snapshot from the HTTP Service, which streams the file out of the blockstore database or filesystem store.

This restore method has the advantage of taking up **no** space on the server hosting the Backup Daemon: the file passes directly from the snapshot storage 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 *oplog store* and applies them, creating a custom snapshot from that point in time. The Daemon then writes the snapshot back to the snapshot storage. Finally, when the user clicks the download link, the user downloads the snapshot from the HTTP Service, which streams the file out of the blockstore database or filesystem storage.

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 retrieves the snapshot from the snapshot storage 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 requires that you have adequate space on the server hosting the Backup Daemon for the snapshot files and archive.

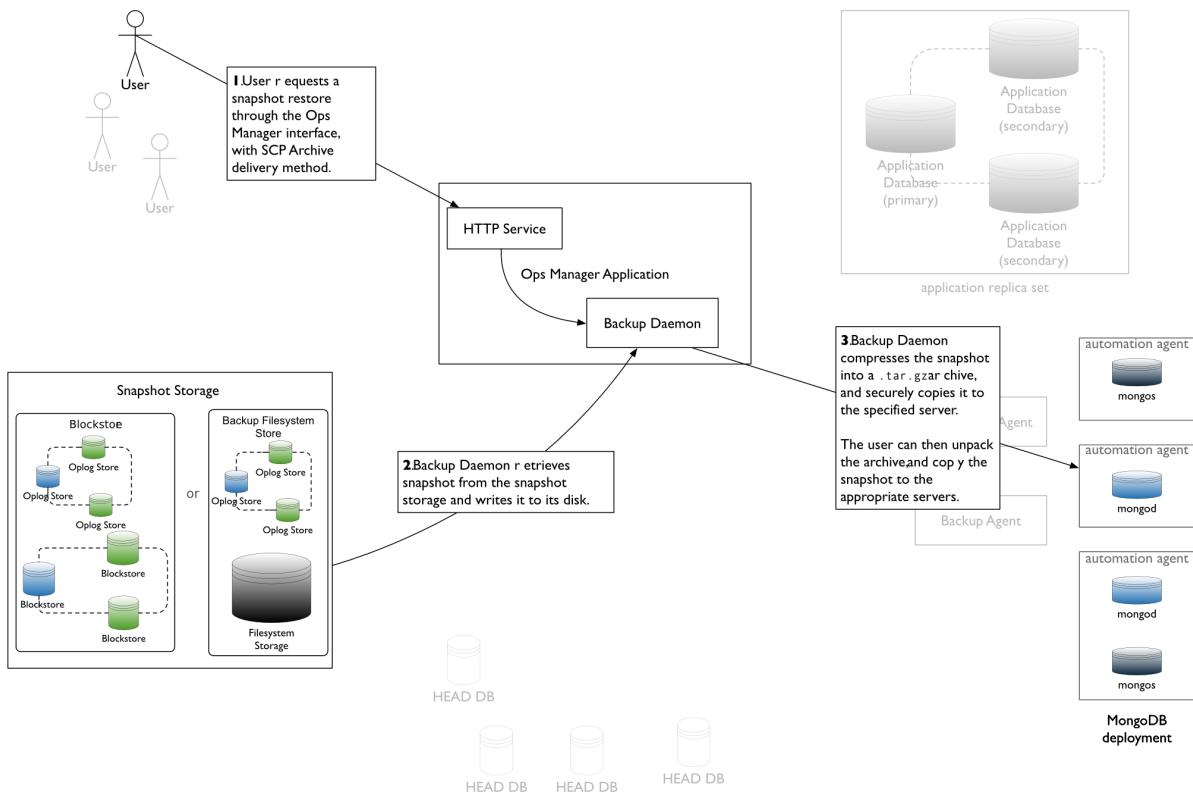
**Point-In-Time** The point-in-time restore with SCP archive delivery follows the same pattern as the snapshot restore, but with added steps for applying the oplog.

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 `oplog store` and applies them, creating a custom snapshot from that point in time. The Daemon then combines and compresses the snapshot into a `.tar.gz` archive and securely copies the archive to the destination server.

This restore method requires 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 retrieves the snapshot from the snapshot storage and securely copies the data files to the target directory on the destination server.



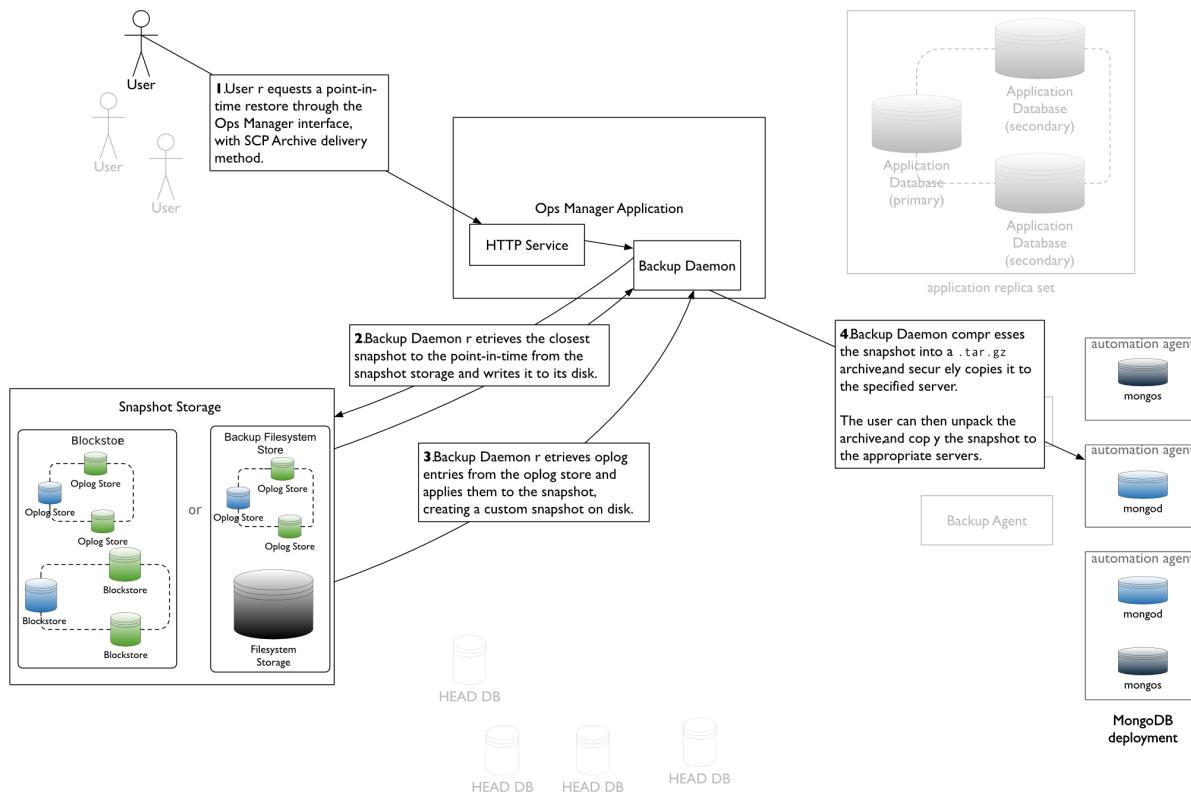
This restore method has the advantage of taking up no space on the server hosting the Backup Daemon: the file passes directly from the snapshot storage to the destination server. The destination server requires only sufficient space for the uncompressed data files. The data *is* compressed during transmission.

**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 *oplog store* and applies them, creating a custom snapshot from that point in time. The Backup Daemon then securely copies the data files to the target directory on the destination server.

This restore method requires 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
- Considerations
- Prerequisites
- Automatic Restore
- Manual Restore

## Overview

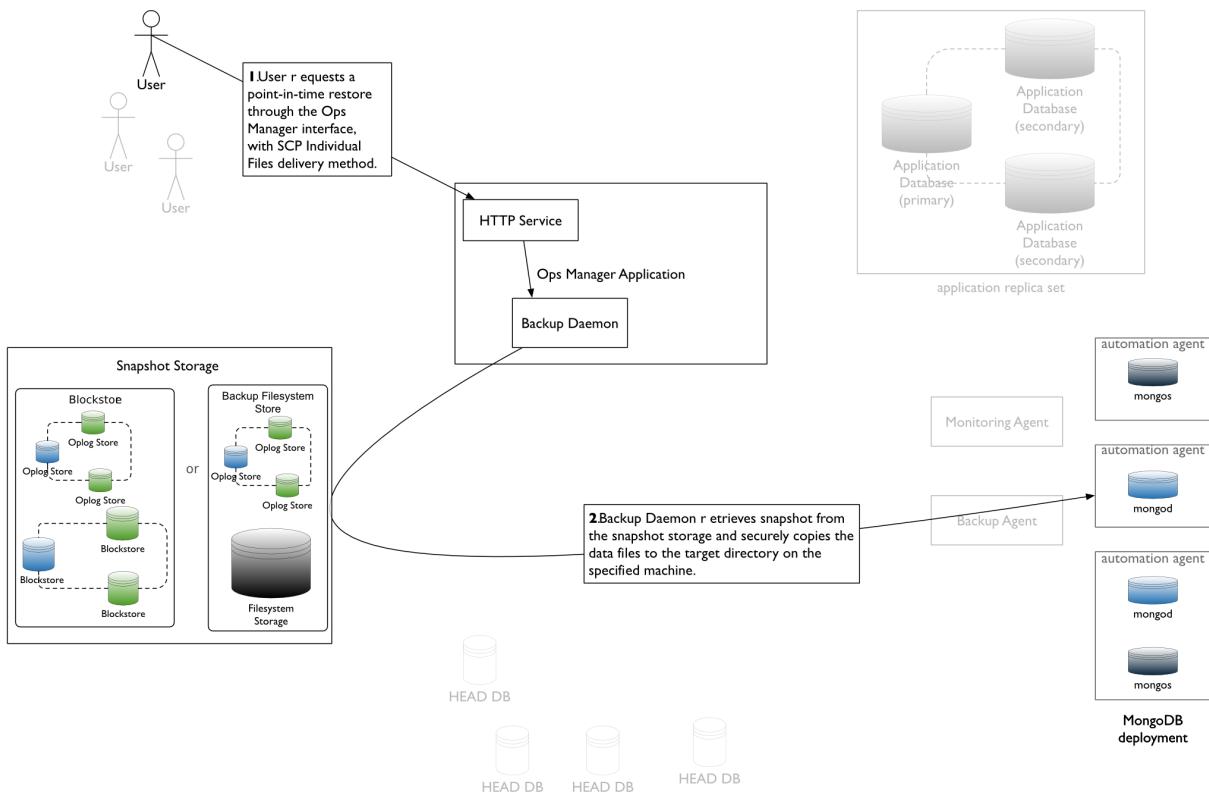
When you restore a cluster from a backup, Ops Manager provides you with a restore files for the selected restore point. For an overview of the restore process, please see [Restore Overview](#).

## Considerations

The BSON specification changed the default subtype for the BSON binary datatype (BinData) from 2 to 0. Some binary data stored in a snapshot may be BinData subtype 2. The Backup Agent automatically detects and converts snapshot data in BinData subtype 2 to BinData subtype 0. If your application code expects BinData subtype 2, you must update your application code to work with BinData subtype 0.

### See also:

The notes on the [BSON specification](#) explain the particular specifics of this change.



The backup restore file now includes a metadata file, `restoreInfo.txt`. This file captures the options the database used when the snapshot was taken. The database must be run with the listed options after it has been restored.

#### See also:

The *Considerations* section of [Seed a New Secondary from Backup Restore](#) explains the use and contents of the `restoreInfo.txt` metadata file.

#### Prerequisites

**Client Requests During Restoration** You must ensure that the MongoDB deployment does not receive client requests during restoration. You must either:

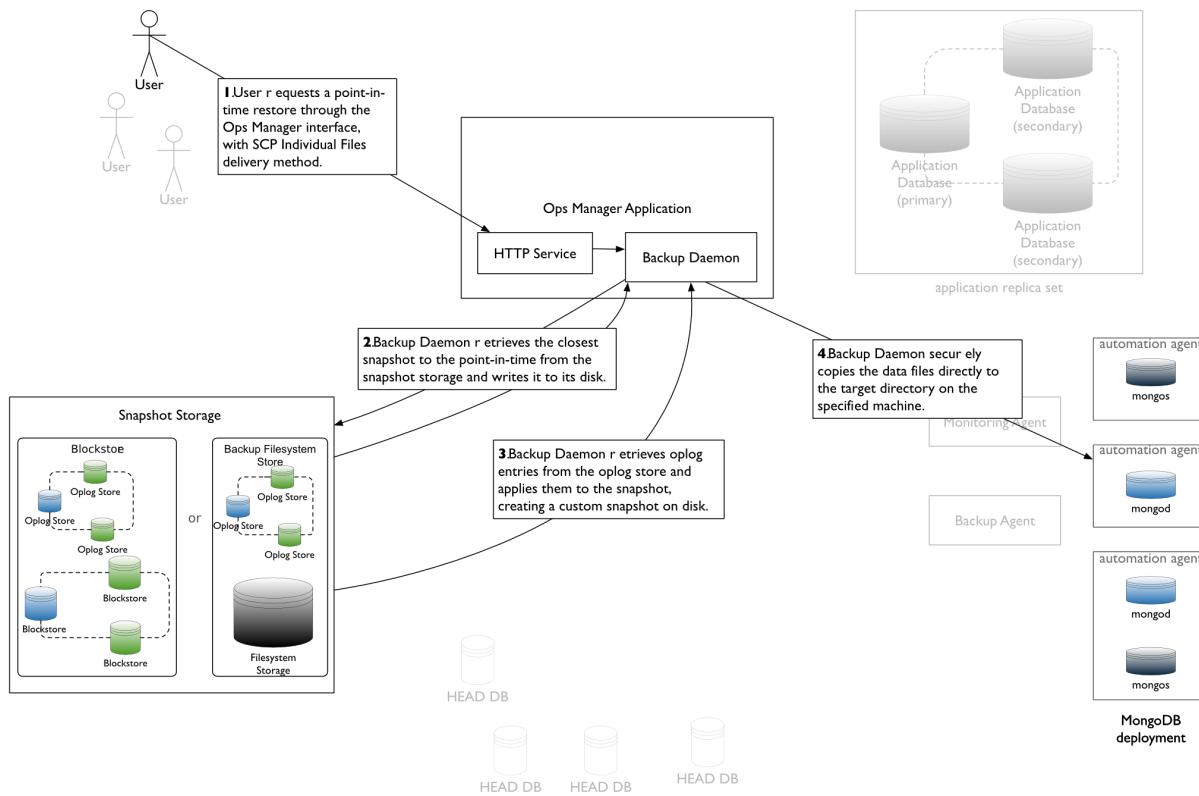
- restore to new systems with new hostnames and reconfigure your application code once the new deployment is running, *or*
- ensure that the MongoDB deployment will *not* receive client requests while you restore data.

**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](#).

**Secure Copy (scp) Delivery** If you intend to deliver using `scp`, you must first [generate a key pair](#).

---

**Note:** Windows machines do not come with `scp` and require additional installation and configuration.




---

**Important:** When copying files individually, the `MaxStartups` value in `sshd_config` should be increased to:  
 $(4 \times (\text{number of shards} + \text{number of config servers})) + 10$

`scp` is performed in parallel and, by default, `sshd` installations use a small number of concurrent connections. Changing this setting allows `scp` to support sufficient connections to complete the restore.

---

### Example

For a sharded cluster with 7 shards and 3 config servers, change `MaxStartups` to 50:

```
MaxStartups 50
```

---

### Automatic Restore

To have Ops Manager automatically restore the backup, perform the *select the snapshot* procedure.

### Manual Restore

To restore the backup manually, perform the following:

1. *Select and Retrieve the Snapshot.*
2. *Restore Primary Replica Set Member for Each Shard.*

3. *Restore All Secondary Replica Set Members for Each Shard.*
4. *Restore Each Config Server.*
5. *Start the MongoDB Sharded Cluster.*

### Select and Retrieve the Snapshot

**Step 1:** Select the *Backup* tab and then *Overview* page.

**Step 2:** Click the deployment's ellipsis icon and select *Restore*.

**Step 3: Select the restore point.** Select the restore point, enter information as needed, and then click *Next*:

<i>Snapshot Point In Time</i>	Restores from a scheduled snapshot. Select the snapshot from which to restore. Creates a custom snapshot based on a <i>checkpoint</i> . Select a <i>Date</i> and <i>Time</i> and click <i>Next</i> .
-------------------------------	---

<b>Step 4:</b> <i>Select whether to have Ops Manager restore the snapshot.</i>	
<i>Yes</i>	Ops Manager restores the snapshot for you. If you select this option, Ops Manager prompts to choose the cluster to which to restore. You can select an existing cluster or create a new one. Follow the prompts and <b>skip the rest of this procedure</b> . Also skip the other procedures on this page. Ops Manager provides you with the restore files, and you perform the restore manually.
<i>No, I'll do it myself</i>	

**Step 5: Select how to receive the restore files.** Select the restore method, format, and destination. Enter information as needed, and then click *Finalize Request*:

<i>Pull Via Secure HTTP</i>	Create a direct download link. Select the following options and click <i>Finalize Request</i> . Skip the rest of this procedure. <ul style="list-style-type: none"> <li>• <i>Pull Restore Usage Limit</i>: Select whether the link can be re-used or used just once. If you select <i>No Limit</i>, the link is re-usable until it expires.</li> <li>• <i>Restore Link Expiration (in hours)</i>: Select the number of hours until the link expires.</li> </ul>
<i>Push Via Secure Copy</i>	Direct Ops Manager to copy the restore files to your server via SCP. To use this option you must have an existing key pair that Ops Manager can use to transmit the files. See <a href="#">Generate a Key Pair for SCP Restores</a> . Windows machines do not come with SCP and require additional setup outside the scope of this manual.
<i>Format</i>	Sets the format of the restore files: <ul style="list-style-type: none"> <li>• <i>Individual DB Files</i>: Transmits MongoDB data files produced by Ops Manager directly to the target directory. The data is compressed during transmission.</li> <li>• <i>Archive (tar.gz)</i>: Delivers database files in a single <code>tar.gz</code> file that you must extract before reconstructing databases. With <i>Archive (tar.gz)</i> delivery, you need sufficient space on the destination server for the archive <i>and</i> the extracted files.</li> </ul>
<i>SCP Host</i>	The hostname of the server to receive the files.
<i>SCP Port</i>	The port of the server to receive the files.
<i>SCP User</i>	The username used to access to the server.
<i>Auth Method</i>	Select whether to use a username and password or an SSH certificate to authenticate to the server.
<i>Password</i>	The user password used to access to the server.
<i>Passphrase</i>	The SSH passphrase used to access to the server.
<i>Target Directory</i>	The absolute path to the directory on the server to which to copy the restore files.

**Step 6: Retrieve the snapshot.** If you selected *Pull Via Secure HTTP*, Ops Manager creates links to the snapshot that by default are available for an hour and can be used just once. To download the snapshot files, select the *Backup* tab and then *Restore History* page. When the restore job completes, a *download* link appears for each *shard* and for one of the *config servers*. Click each link to download the files and copy each to its server. For a shard, copy the file to every member of the shard's *replica set*.

If you selected *Push Via Secure Copy*, Ops Manager copies the files to the server directory you specified. To verify that the files are complete, see the section on how to validate an SCP restore. For each shard, copy its restore file to every member of the shard's *replica set*.

**Restore Primary Replica Set Member for Each Shard** For all shards, restore the primary replica set member. You must have a copy of the snapshot on the server that host the primary replica set member.

**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 a MongoDB Process](#).

- **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 process 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 <backupRestoreName>.tar.gz
mv <backupRestoreName> /data
```

**Step 3: Start the primary with the new dbpath.** For example:

```
mongod --dbpath <pathToData> --replSet <replicaSetName> --logpath <pathToData>/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 a MongoDB Process](#).

- **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 /<pathToData> --logpath /<pathToData>/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 script on the primary.** Use the appropriate script for your operating system:

UNIX-based	<code>seedSecondary.sh</code>
Windows	<code>seedSecondary.bat</code>

This script recreates the oplog collection and seeds it with the timestamp of the snapshot's creation. This allows the secondary to come back up to time without requiring a full initial sync. Ops Manager customizes this script for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command, where:

<code>&lt;mongodPort&gt;</code>	The port of the mongod process
<code>&lt;oplogSizeInGigabytes&gt;</code>	The size of the replica set's oplog
<code>&lt;replicaSetName&gt;</code>	The name of the replica set
<code>&lt;primaryHost:primaryPort&gt;</code>	The hostname:port combination for the replica set's primary

- For **UNIX-based** systems:

```
./seedSecondary.sh <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>
./seedSecondary.sh 27018 2 rs-1 primaryHost.example.com:27017
```

- For **Windows-based** systems:

```
.\\seedSecondary.bat <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>
.\\seedSecondary.bat 27018 2 rs-1 primaryHost.example.com:27017
```

**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 a MongoDB Process](#).

- **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 /<pathToData> --replSet <replicaSetName>
```

**Restore All Secondary Replica Set Members for Each Shard** After you have restored the primary for a shard you can restore all secondary replica set members. You must have a copy of the snapshot on all servers that host a secondary replica set member.

## 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 process 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 <backupRestoreName>.tar.gz  
mv <backupRestoreName> /data
```

## Step 3: Start the secondary as a standalone, *without the --replicaSet 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 a MongoDB Process](#).

- **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 /<pathToData> --logpath /<pathToData>/mongodb.log --fork
```

## Step 4: Run the seedSecondary.sh script on the secondary. Use the appropriate script for your operating system:

UNIX-based	seedSecondary.sh
Windows	seedSecondary.bat

This script recreates the oplog collection and seeds it with the timestamp of the snapshot's creation. This allows the secondary to come back up to time without requiring a full initial sync. Ops Manager customizes this script for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command, where:

<mongodPort>	The port of the mongod process
<oplogSizeInGigabytes>	The size of the replica set's oplog
<replicaSetName>	The name of the replica set
<primaryHost:primaryPort>	The hostname:port combination for the replica set's primary

- For **UNIX-based** systems:

```
./seedSecondary.sh <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>  
./seedSecondary.sh 27018 2 rs-1 primaryHost.example.com:27017
```

- For **Windows-based** systems:

```
.\seedSecondary.bat <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>  
.\\seedSecondary.bat 27018 2 rs-1 primaryHost.example.com:27017
```

**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 a MongoDB Process](#).

- **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 <pathToData> --replSet <replicaSetName>
```

**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.

Use this procedure only for manual restores.

**Step 1: Restore the snapshot to the config server.** Extract the data files to the location where the config server's `mongod` process 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` process** Start the cluster's `mongos` bound to your new config servers.

## Restore a Replica Set from a Backup

### On this page

- [Overview](#)
- [Considerations](#)
- [Prerequisites](#)
- [Procedures](#)

### Overview

When you restore a replica set from backup, Ops Manager provides you with a restore file for the selected restore point. For an overview of the restore process, please see [Restore Overview](#).

### Considerations

The [BSON specification](#) changed the default subtype for the BSON binary datatype (`BinData`) from 2 to 0. Some binary data stored in a snapshot may be `BinData` subtype 2. The Backup Agent automatically detects and converts snapshot data in `BinData` subtype 2 to `BinData` subtype 0. If your application code expects `BinData` subtype 2, you must update your application code to work with `BinData` subtype 0.

#### See also:

The notes on the [BSON specification](#) explain the particular specifics of this change.

### Prerequisites

**Oplog Size** To seed each replica set member, you will use the `seedSecondary.sh` or `seedSecondary.bat` script included in the backup restore file. When you run the script, you will provide the replica set's oplog size, in gigabytes.

#### See also:

See the **Check the Size of the Oplog** section of the [Troubleshoot Replica Sets](#) if you do not have the size.

**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.

**Secure Copy (scp) Delivery** If you intend to delivery using `scp`, you must *generate a key pair* before attempting to deliver the files. Windows machines do not come with `scp` and require additional installation and configuration.

## Procedures

To have Ops Manager automatically restore the backup, perform the *select the snapshot* procedure.

To restore the backup manually, perform all the following procedures sequentially. If you restore to existing hardware, use a different data directory than used previously.

### See also:

See [Restore a Replica Set from a Backup](#) for alternative approaches to restoring replica sets.

### Select the Snapshot

**Step 1: Select the *Backup* tab and then *Overview* page.**

**Step 2: Click the deployment's ellipsis icon and select *Restore*.**

**Step 3: Select the restore point.** Select the restore point, enter information as needed, and then click *Next*:

<p><i>Snapshot</i></p> <p><i>Point In Time</i></p> <p><i>Oplog Timestamp</i></p>	<p>Restores from a <i>stored snapshot</i>. Select the snapshot from which to restore.</p> <p>Creates a custom snapshot based on the time you choose. Ops Manager includes all operations up to but not including the selected time. For example, if you select 12:00, the last operation in the restore is 11:59:59 or earlier.</p> <p>Select a <i>Date and Time</i> and click <i>Next</i>.</p> <p>Creates a custom snapshot based on the timestamp of an entry in the <i>oplog</i>, as specified by the entry's <code>ts</code> field. Ops Manager includes all operations up to <b>and including</b> the time of the timestamp. An entry's <code>ts</code> field is a <b>BSON</b> timestamp and has two components: the timestamp and the increment.</p> <p>Specify the following:</p> <ul style="list-style-type: none"> <li><i>Timestamp</i>: The value in seconds since the Unix epoch.</li> <li><i>Increment</i>: An incrementing ordinal for operations within a given second.</li> </ul>
--	--

Step 4: Select whether to have Ops Manager restore the snapshot.	
<i>Yes</i>	Ops Manager restores the snapshot for you. If you select this option, Ops Manager prompts to choose the replica set to which to restore. You can select an existing replica set or create a new one. Follow the prompts and <b>skip the rest of this procedure</b> . Also skip the other procedures on this page.
<i>No, I'll do it myself</i>	Ops Manager provides you with the restore files for you to perform the restore manually.

**Step 5: Select how to receive the restore files.** Select the restore method, format, and destination. Enter information as needed, and then click *Finalize Request*:

<i>Pull Via Secure HTTP</i>	Create a direct download link. Select the following options and click <i>Finalize Request</i> . Skip the rest of this procedure. <ul style="list-style-type: none"> <li>• <i>Pull Restore Usage Limit</i>: Select whether the link can be re-used or used just once. If you select <i>No Limit</i>, the link is re-usable until it expires.</li> <li>• <i>Restore Link Expiration (in hours)</i>: Select the number of hours until the link expires.</li> </ul>
<i>Push Via Secure Copy</i>	Direct Ops Manager to copy the restore files to your server via SCP. To use this option you must have an existing key pair that Ops Manager can use to transmit the files. See <a href="#">Generate a Key Pair for SCP Restores</a> . Windows machines do not come with SCP and require additional setup outside the scope of this manual.
<i>Format</i>	Sets the format of the restore files: <ul style="list-style-type: none"> <li>• <i>Individual DB Files</i>: Transmits MongoDB data files produced by Ops Manager directly to the target directory. The data <i>is</i> compressed during transmission.</li> <li>• <i>Archive (tar.gz)</i>: Delivers database files in a single <code>tar.gz</code> file that you must extract before reconstructing databases. With <i>Archive (tar.gz)</i> delivery, you need sufficient space on the destination server for the archive <i>and</i> the extracted files.</li> </ul>
<i>SCP Host</i>	The hostname of the server to receive the files.
<i>SCP Port</i>	The port of the server to receive the files.
<i>SCP User</i>	The username used to access to the server.
<i>Auth Method</i>	Select whether to use a username and password or an SSH certificate to authenticate to the server.
<i>Password</i>	The user password used to access to the server.
<i>Passphrase</i>	The SSH passphrase used to access to the server.
<i>Target Directory</i>	The absolute path to the directory on the server to which to copy the restore files.

**Step 6: Retrieve the snapshot.** If you selected *Pull Via Secure HTTP*, Ops Manager creates a link to the snapshot that by default is available for an hour and can be used just once. To download the snapshot, select the *Backup* tab and then *Restore History* page. When the restore job completes, select the *download* link next to the snapshot.

If you selected *Push Via Secure Copy*, the files are copied to the server directory you specified. 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. Use this procedure only for manual restores.

**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 a MongoDB Process](#).

- **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 process 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 <backupRestoreName>.tar.gz
mv <backupRestoreName> /data
```

**Step 3: Start the primary with the new dbpath.** For example:

```
mongod --dbpath <pathToData> --replSet <replicaSetName> --logpath <pathToData>/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 a MongoDB Process](#).

- **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 /<pathToData> --logpath /<pathToData>/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 script on the primary.** Use the appropriate script for your operating system:

UNIX-based	<code>seedSecondary.sh</code>
Windows	<code>seedSecondary.bat</code>

This script recreates the oplog collection and seeds it with the timestamp of the snapshot's creation. This allows the secondary to come back up to time without requiring a full initial sync. Ops Manager customizes this script for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command, where:

<code>&lt;mongodPort&gt;</code>	The port of the mongod process
<code>&lt;oplogSizeInGigabytes&gt;</code>	The size of the replica set's oplog
<code>&lt;replicaSetName&gt;</code>	The name of the replica set
<code>&lt;primaryHost:primaryPort&gt;</code>	The hostname:port combination for the replica set's primary

- For **UNIX-based** systems:

```
./seedSecondary.sh <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>
./seedSecondary.sh 27018 2 rs-1 primaryHost.example.com:27017
```

- For **Windows-based** systems:

```
.\\seedSecondary.bat <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>
.\\seedSecondary.bat 27018 2 rs-1 primaryHost.example.com:27017
```

**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 a MongoDB Process](#).

- **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 /<pathToData> --replSet <replicaSetName>
```

**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. Use this procedure only for manual restores.

**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 process 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 <backupRestoreName>.tar.gz  
mv <backupRestoreName> /data
```

**Step 3: Start the secondary as a standalone, without the --replicaSet 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 a MongoDB Process](#).

- **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 /<pathToData> --logpath /<pathToData>/mongodb.log --fork
```

**Step 4: Run the seedSecondary.sh script on the secondary.** Use the appropriate script for your operating system:

UNIX-based	seedSecondary.sh
Windows	seedSecondary.bat

This script recreates the oplog collection and seeds it with the timestamp of the snapshot's creation. This allows the secondary to come back up to time without requiring a full initial sync. Ops Manager customizes this script for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command, where:

<mongodPort>	The port of the mongod process
<oplogSizeInGigabytes>	The size of the replica set's oplog
<replicaSetName>	The name of the replica set
<primaryHost:primaryPort>	The hostname:port combination for the replica set's primary

- For **UNIX-based** systems:

```
./seedSecondary.sh <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>  
./seedSecondary.sh 27018 2 rs-1 primaryHost.example.com:27017
```

- For **Windows-based** systems:

```
.\\seedSecondary.bat <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>
.\\seedSecondary.bat 27018 2 rs-1 primaryHost.example.com:27017
```

### 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 a MongoDB Process](#).

- **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 /<pathToData> --replSet <replicaSetName>
```

### 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 a Single Database

### On this page

- [Overview](#)
- [Considerations](#)
- [Select and Download a Snapshot](#)
- [Restore the Database](#)

### Overview

Backup snapshots contain a complete copy of your `mongod` data directory. There may be occasions when you want to restore only part of the data in a snapshot. To restore a single database instead of all databases in the snapshot, load the snapshot first to a temporary `mongod` instance and then export from this instance the single database. Use the export of the single database to restore to the target deployment.

**Note:** As of MongoDB 3.0, `mongodump` and `mongorestore` no longer support the `--dbpath` option. These tools now must connect to a running `mongod` instance.

## Considerations

Before you attempt a restore, ensure your target storage has sufficient space for the restore files and the restored database, plus additional space for dataset growth. Use `db.stats()` to find the current database size.

### Select and Download a Snapshot

**Step 1:** Select the *Backup* tab and then *Overview* page.

**Step 2:** Click the deployment's ellipsis icon and select *Restore*.

**Step 3:** Select the restore point. Select the restore point, enter information as needed, and then click *Next*:

<i>Snapshot</i>	Restores from a <i>stored snapshot</i> . Select the snapshot from which to restore.
<i>Point In Time</i>	Creates a custom snapshot based on the time you choose. Ops Manager includes all operations up to but not including the selected time. For example, if you select 12:00, the last operation in the restore is 11:59:59 or earlier. Select a <i>Date</i> and <i>Time</i> and click <i>Next</i> .
<i>Oplog Timestamp</i>	Creates a custom snapshot based on the timestamp of an entry in the <i>oplog</i> , as specified by the entry's <code>ts</code> field. Ops Manager includes all operations up to <b>and including</b> the time of the timestamp. An entry's <code>ts</code> field is a <b>BSON</b> timestamp and has two components: the timestamp and the increment. Specify the following: <ul style="list-style-type: none"><li>• <i>Timestamp</i>: The value in seconds since the Unix epoch.</li><li>• <i>Increment</i>: An incrementing ordinal for operations within a given second.</li></ul>

**Step 4:** Select how to receive the restore files. Select the restore method, format, and destination. Enter information as needed, and then click *Finalize Request*:

<i>Pull Via Secure HTTP</i>	Create a direct download link. Select the following options and click <i>Finalize Request</i> . Skip the rest of this procedure. <ul style="list-style-type: none"> <li>• <i>Pull Restore Usage Limit</i>: Select whether the link can be re-used or used just once. If you select <i>No Limit</i>, the link is re-usable until it expires.</li> <li>• <i>Restore Link Expiration (in hours)</i>: Select the number of hours until the link expires.</li> </ul>
<i>Push Via Secure Copy</i>	Direct Ops Manager to copy the restore files to your server via SCP. To use this option you must have an existing key pair that Ops Manager can use to transmit the files. See <a href="#">Generate a Key Pair for SCP Restores</a> . Windows machines do not come with SCP and require additional setup outside the scope of this manual.
<i>Format</i>	Sets the format of the restore files: <ul style="list-style-type: none"> <li>• <i>Individual DB Files</i>: Transmits MongoDB data files produced by Ops Manager directly to the target directory. The data is compressed during transmission.</li> <li>• <i>Archive (tar.gz)</i>: Delivers database files in a single <code>tar.gz</code> file that you must extract before reconstructing databases. With <i>Archive (tar.gz)</i> delivery, you need sufficient space on the destination server for the archive and the extracted files.</li> </ul>
<i>SCP Host</i>	The hostname of the server to receive the files.
<i>SCP Port</i>	The port of the server to receive the files.
<i>SCP User</i>	The username used to access to the server.
<i>Auth Method</i>	Select whether to use a username and password or an SSH certificate to authenticate to the server.
<i>Password</i>	The user password used to access to the server.
<i>Passphrase</i>	The SSH passphrase used to access to the server.
<i>Target Directory</i>	The absolute path to the directory on the server to which to copy the restore files.

**Step 5: Retrieve the snapshot.** If you selected *Pull Via Secure HTTP*, Ops Manager creates a link to the snapshot that by default is available for an hour and can be used just once. To download the snapshot, select the *Backup* tab and then *Restore History* page. When the restore job completes, select the *download* link next to the snapshot.

If you selected *Push Via Secure Copy*, the files are copied to the server directory you specified. To verify that the files are complete, see the section on how to validate an SCP restore.

## Restore the Database

The following examples use:

- 27017 as the port for a running MongoDB instance that receives the restored backup.
- 27018 as the port for the temporary MongoDB instance that uses the snapshot to export the single database.

**Step 1: Restore the snapshot data files to the server.** Extract the snapshot archive to a temporary location where a temporary `mongod` process will access the archive contents. Use a different data directory than any other database

running on the server.

For example:

```
tar -xvf <backupRestoreName>.tar.gz  
mv <backupRestoreName> <temp-database-path>
```

**Step 2: Start a new, temporary MongoDB process on a new port using the extracted backup snapshot as the dbpath.** Ensure the user executing the mongod can read, write and execute code in the directory specified with dbpath.

For example:

```
mongod --port 27018 --dbpath <temp-database-path> --logpath <temp-database-path>/mongodb.log --fork
```

**Step 3: Use the mongodump command to export a single database or collection from the temporary running mongod process.** Specify the single database name using --db and, if needed, a --collection for a single collection.

The --out option specifies where the mongodump extracts the target database. Choose an empty directory the user executing mongodump can access.

```
mongodump --port 27018 --db <single-database> --out <new-database-path>
```

It also possible to export only a single collection:

```
mongodump --port 27018 --db <single-database> --collection <collection-name> --out <new-database-path>
```

**Step 4: Use the mongorestore command to import the single database or collection.** Restore the single database to the desired instance using this mongorestore command:

```
mongorestore --port 27017 --db <single-database> <temp-database-path> --drop
```

If you are restoring a single collection, be sure to designate the collection:

```
mongorestore --port 27017 --db <single-database> --collection <collection-name> <temp-database-path>
```

Any existing databases matching the name given for the --db option should be dropped using the --drop option. If you choose not to use the --drop option, the restore may produce errors for any documents with duplicate \_id fields.

**Step 5: Shutdown the temporary MongoDB instance and remove the temporary database.**

1. Start the mongo shell.

```
mongo <port>
```

2. Drop the database and shutdown the process.

```
admin = db.getSiblingDB("admin")  
admin.shutdownServer()  
exit
```

3. Delete the temporary database directory

```
rm -rf <temp-database-path>
```

## Seed a New Secondary from Backup Restore

### On this page

- Overview
- Considerations
- Prerequisites
- Procedure

### Overview

When a new secondary member is added to a replica set, MongoDB will synchronize data to that new secondary as it would with any member of the replica set. This can be very resource or time intensive if the database is large.

As an alternative to the regular synchronization, the new secondary member can first be “seeded” with data from an existing backup. This limits performance impact to the new secondary and allows the active MongoDB process to continue storing and retrieving data.

### 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.

**Important:** The `seedSecondary` script is omitted from the restore files if you have:

- Blacklisted databases or collections or
  - The legacy sync cluster connection configuration servers for a sharded cluster
- Deployments using replica set configuration servers are unaffected.

If you were to have received the script as part of the restore files, then tried to apply it, the secondary would become inconsistent with the other members.

---

### See also:

[Upgrade Config Servers to Replica Set](#).

Seeding a new secondary from a backup restore requires an oplog window on the current primary that spans back to the snapshot’s timestamp.

---

### Prerequisites

To seed a secondary from a backup restore file, you must have:

- A backup restore file.
- The `seedSecondary.sh` and `seedSecondary.bat` scripts included in the backup restore file.

When you run the `seedSecondary.sh` script on a Unix-based system or `seedSecondary.bat` script on a Windows-based system as part of this tutorial, you must provide the replica set’s oplog size in gigabytes.

### See also:

See [Check the Size of the Oplog](#) on [Troubleshoot Replica Sets](#) if you do not have the size.

## 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 <backupRestoreName>.tar.gz  
mv <backupRestoreName> data  
mongod --port <alternatePort> --dbpath /data
```

Where <alternatePort> is not the usual port on which your secondary runs.

### Step 4: Run the seedSecondary script to create oplog collection and seed it with correct timestamp.

The seedSecondary script re-creates the oplog collection and seed it with the correct timestamp.

Use the appropriate script for your operating system:

UNIX-based	seedSecondary.sh
Windows	seedSecondary.bat

This script recreates the oplog collection and seeds it with the timestamp of the snapshot's creation. This allows the secondary to come back up to time without requiring a full initial sync. Ops Manager customizes this script for this particular snapshot and is included in the backup restore file.

To run the script, issue the following command, where:

<mongodPort>	The port of the mongod process
<oplogSizeInGigabytes>	The size of the replica set's oplog
<replicaSetName>	The name of the replica set
<primaryHost:primaryPort>	The hostname:port combination for the replica set's primary

- For **UNIX-based** systems:

```
./seedSecondary.sh <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>  
./seedSecondary.sh 27018 2 rs-1 primaryHost.example.com:27017
```

- For **Windows-based** systems:

```
.\\seedSecondary.bat <mongodPort> <oplogSizeInGigabytes> <replicaSetName> <primaryHost:primaryPort>  
.\\seedSecondary.bat 27018 2 rs-1 primaryHost.example.com:27017
```

### Step 5: Shut down the new secondary on the alternate port.

### Step 6: Start up the new secondary.

```
mongod --port <secondaryPort> --dbpath /data --replSet <replicaSetName>
```

### Step 7: Add the new secondary to the replica set on the primary host.

```
rs.add("<secondary-host>:<secondary-port>")
```

# 7 Security

**Security Options & Support Matrices** Describes Ops Manager security features.

**Firewall Configuration** Describes the ports used by Ops Manager components.

**Manage Ops Manager Ports** Describes how to change the default ports used by Ops Manager on Windows and Unix-based systems.

**SSL Connections to Ops Manager** Run Ops Manager 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** Enable SSL for connections to your MongoDB deployments.

**LDAP for Ops Manager Users** Configure Ops Manager to use LDAP to store user data and permissions.

**Access Control for MongoDB Deployments** Configure the Authentication Mechanisms used by your Ops Manager group for communication between the Ops Manager agents and your deployments.

**System-wide Two-Factor Authentication** Configure two-factor authentication.

## 7.1 Security Overview

### On this page

- [Overview](#)
- [Security Options Available in the Current Version of Ops Manager](#)
- [Supported User Authentication by Ops Manager Version](#)
- [Supported MongoDB Security Features on Linux](#)
- [Supported MongoDB Security Features on Windows](#)

### Overview

To ensure the security of your Ops Manager agents, Ops Manager servers, and MongoDB deployments, Ops Manager supports the security options described on this page.

### Security Options Available in the Current Version of Ops Manager

The following table displays the security options available for the different types of connections in your Ops Manager environment and provides links to setup instructions.

	Connections with Ops Manager	Connections with Backing Databases	Connections with MongoDB Deployments
Ops Manager	<i>Not applicable.</i>	<ul style="list-style-type: none"> <li>• <i>SSL</i></li> <li>• <i>MONGODB-CR</i></li> <li>• <i>LDAP</i> (Linux only)</li> <li>• <i>Kerberos</i></li> </ul>	<i>Connects through the Monitoring, Backup, and Automation agents. See the row for each agent in this table.</i>
Monitoring Agent	<ul style="list-style-type: none"> <li>• <i>SSL</i></li> </ul>	<i>Not applicable.</i>	<ul style="list-style-type: none"> <li>• <i>SSL</i></li> <li>• <i>MONGODB-CR</i></li> <li>• <i>LDAP</i> (Linux only)</li> <li>• <i>Kerberos</i> (Linux only)</li> <li>• <i>x.509</i></li> </ul>
Backup Agent	<ul style="list-style-type: none"> <li>• <i>SSL</i></li> </ul>	<i>Not applicable.</i>	<ul style="list-style-type: none"> <li>• <i>SSL</i></li> <li>• <i>MONGODB-CR</i></li> <li>• <i>LDAP</i> (Linux only)</li> <li>• <i>Kerberos</i> (Linux only)</li> <li>• <i>x.509</i></li> </ul>
Automation Agent	<ul style="list-style-type: none"> <li>• <i>SSL</i></li> </ul>	<i>Not applicable.</i>	<ul style="list-style-type: none"> <li>• <i>SSL</i></li> <li>• <i>MONGODB-CR</i></li> <li>• <i>LDAP</i> (Linux only)</li> <li>• <i>Kerberos</i> (Linux only)</li> <li>• <i>x.509</i></li> </ul>
Ops Manager user	<ul style="list-style-type: none"> <li>• <i>SSL</i></li> <li>• <i>Ops Manager Authentication</i></li> <li>• <i>LDAP</i></li> </ul>	<i>Not applicable.</i>	<i>See Authentication in the MongoDB manual.</i>

## Supported User Authentication by Ops Manager Version

The following table shows the available user authentication mechanisms and the release the mechanism became available.

Method	Ops Manager Versions
Authentication against Ops Manager Application Database	1.0+
Authentication against LDAP	1.4+

## Supported MongoDB Security Features on Linux

This section describes supported security options on Linux.

### Connections Between Ops Manager and the Backing Databases (Linux)

The following table shows security options for connections between Ops Manager and the *Ops Manager Application Database* and *Backup Data Storage* when both run on Linux.

	User-name/Password Authentication	MongoDB SSL Connections	Kerberos Authentication	MongoDB SSL Connections with Client Certificates	x509 Authentication	LDAP Authentication
Ops Manager Versions	1.0+	1.6+	1.3+	1.6+	1.6+	1.5+

### Connections Between Agents and MongoDB Deployments (Linux)

The following table shows the security options available for connections between agents and the MongoDB deployments they manage when the deployments run on Linux:

	User-name/Password Authentication	MongoDB SSL Connections	Kerberos Authentication	MongoDB SSL Connections with Client Certificates	x509 Authentication	LDAP Authentication
Monitoring Agent Backup Agent Automation Agent	1.0+	1.0+	1.3+	1.5+	1.8	1.5+
	1.4+	1.4+	1.4.1+	1.5+	1.8	1.5+
	1.6+	1.8	1.8	1.8	1.8	1.8

### Supported MongoDB Security Features on Windows

This section describes supported security options on Windows.

### Connections Between Ops Manager and the Backing Databases (Windows)

The following table shows security options for connections between Ops Manager and the *Ops Manager Application Database* and *Backup Data Storage* when both run on Windows.

	User-name/Password Authentication	MongoDB SSL Connections	Kerberos Authentication	MongoDB SSL Connections with Client Certificates	x509 Authentication	LDAP Authentication
Ops Manager Versions	1.5+	1.6+		1.6+	1.6+	Not applicable.

### Connections Between Agents and MongoDB Deployments (Windows)

The following table shows the security options available for connections between agents and the MongoDB deployments they manage when the deployments run on Windows:

	User-name/Password Authentication	MongoDB SSL Connections	Kerberos Authentication	MongoDB SSL Connections with Client Certificates	x509 Authentication	LDAP Authentication
Monitoring Agent Backup Agent Automation Agent	1.5+	1.5+		1.5+	1.8	<i>Not applicable.</i>
	1.5+	1.5+		1.5+	1.8	<i>Not applicable.</i>
	1.8	1.8		1.8	1.8	<i>Not applicable.</i>

**Note:** MongoDB for Windows does not support LDAP.

## 7.2 Firewall Configuration

### On this page

- Ports Required to Use Ops Manager
- Ports Needed to Administer Ops Manager
- Ports Needed to Integrate Ops Manager with SNMP
- Ports Needed to Authenticate with Ops Manager
- Ports Needed to Authenticate with MongoDB

### Overview

Ops Manager connects with a number of services which may also use SSL. This page explains the necessary ports to deploy the various components used with an Ops Manager deployment.

### Accessible Ports

The Ops Manager application must be able to connect to users and Ops Manager agents over or Secure HTTP (HTTPS). Ops Manager agents must be able to connect to MongoDB client MongoDB databases.

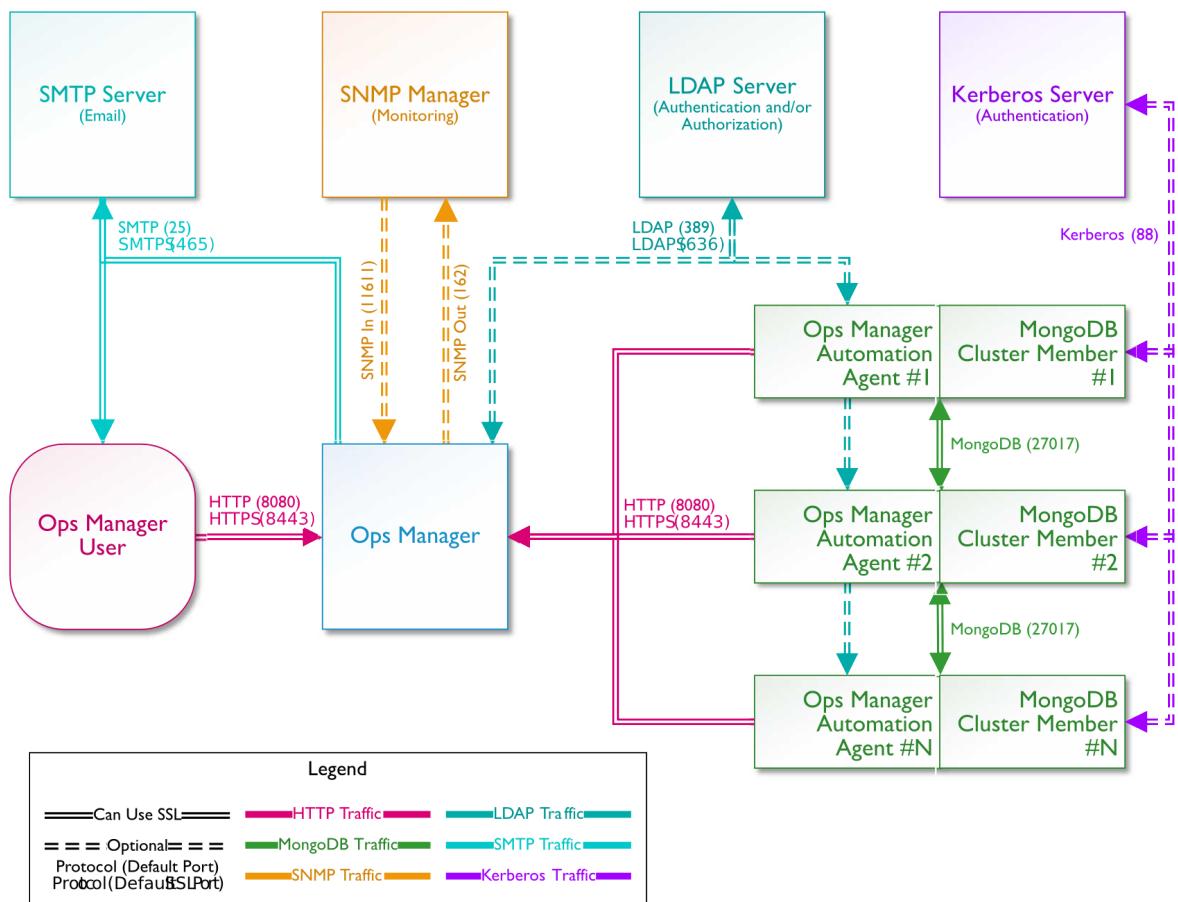
Though Ops Manager only requires open HTTP(S) and MongoDB network ports to connect with users and to databases, what ports are opened on a firewall depend upon what capabilities are enabled: encryption, authentication and monitoring.

This page defines which systems need to connect to which ports on other systems.

**Important:** To run Ops Manager without an Internet connection, see [Configure Local Mode if Servers Have No Internet Access](#) to ensure you have all of the necessary binaries to run Ops Manager without an Internet connection.

### Ports Required to Use Ops Manager

- Both Ops Manager users and Ops Manager agents must be able to connect to Ops Manager application over HTTP(S). See [Manage Ops Manager Ports](#) to set a non-default port for Ops Manager.
- Ops Manager must be able to connect to the backing MongoDB databases running mongod.



- For each Ops Manager group, Ops Manager agents must be able to connect to all client MongoDB processes (mongod or mongos).
- The Ops Manager application must also be able to send email to Ops Manager users.

To use Ops Manager, open the following ports to the specified servers.

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
HTTP	8080	TCP	Inbound	Web connection to Ops Manager from users and Ops Manager agents.	No
HTTPS	8443	TCP	Inbound	Web connection to Ops Manager from users and Ops Manager agents.	Yes
HTTP(S)	8090	TCP	Inbound	<p>Provides a health-check endpoint for monitoring Ops Manager through a monitoring service like Zabbix or Nagios. This is disabled by default.</p> <p>To enable it, see <a href="#">Enable the Health Check Endpoint</a>. When enabled, you can access the endpoint at:</p> <pre>http://&lt;opsmanagerhost&gt;:8090/healthcheck</pre> <p>The API endpoint provides the ability to check connections from the HTTP Service to the <a href="#">Ops Manager Application Database</a> and the <a href="#">Backup Snapshot Storage</a>.</p> <p>A successful response returns the following:</p> <pre>{   "mms_db": "OK",   "backup_db": "OK" }</pre>	Optional
MongoDB	27000 - 28000	TCP	Both	Connect to MongoDB application, backup and client databases.	Optional
SMTP	25	TCP	Outbound	Send emails from Ops Manager.	No
SMTPS	465	TCP	Outbound	Send emails from Ops Manager.	Yes

## **Ports Needed to Administer Ops Manager**

Most Ops Manager administration can be performed through the user interface. Some procedures require access to the operating system. To permit your administrators to access your Ops Manager and MongoDB servers, open the following ports to those servers.

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
Secure Shell (ssh)	22	TCP	Inbound	Linux System administration.	Yes
Remote Desktop Connection (RDP)	3389	TCP	Inbound	Windows System administration.	No

## **Ports Needed to Integrate Ops Manager with SNMP**

To send and receive SNMP messages to and from your MongoDB deployments must open the following ports between Ops Manager and your SNMP Manager.

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
SNMP	162	UDP	Outbound	Send Traps to SNMP Manager.	No
SNMP	11611	UDP	Inbound	Receive requests from SNMP Manager.	No

See [SNMP Heartbeat Settings](#) to set SNMP to use non-standard ports.

## **Ports Needed to Authenticate with Ops Manager**

MongoDB Enterprise users [can use LDAP](#) to authenticate Ops Manager users. To authenticate using LDAP, open the following ports on Ops Manager and your LDAP server.

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
LDAP	389	UDP	Both	Authenticate and/or authorize Ops Manager users against LDAP server.	No
LDAPS	636	UDP	Both	Authenticate and/or authorize Ops Manager users against LDAP server.	Yes

See [Authentication through LDAP](#) to configure LDAP URI strings including ports.

## **Ports Needed to Authenticate with MongoDB**

MongoDB Enterprise users can use Kerberos or LDAP to authenticate MongoDB users. To authenticate using LDAP or Kerberos, open the following ports between the MongoDB client databases, Ops Manager and the Kerberos or LDAP server(s).

Service	Default Port	Transport	Direction	Purpose	Uses SSL?
Kerberos	88	TCP / UDP	Out-bound	Request authentication for MongoDB users against Kerberos server.	No
Kerberos	88	UDP	In-bound	Receive authentication for MongoDB users against Kerberos server.	No
LDAP	389	UDP	Both	Authenticate and/or authorize MongoDB users against LDAP server.	No
LDAPS	636	UDP	Both	Authenticate and/or authorize MongoDB users against LDAP server.	Yes

See *Kerberos Authentication to the Application Database* to configure Kerberos for authentication to the Ops Manager application database.

## 7.3 Manage Ops Manager Ports

### On this page

- Overview
- Procedures

### Overview

Ops Manager uses the ports and health-check endpoints described in *Firewall Configuration*. The endpoints are disabled by default. You can change Ops Manager's default port and enable the endpoints.

### 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 Ops Manager URL in *Ops Manager Config* settings, 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: Edit the Ops Manager URL to use the new port.

1. In Ops Manager, click the *Admin* link in the upper right corner of the page.
2. Click the *General* tab and then click *Ops Manager Config*.
3. Update the *URL to Access Ops Manager* field to use the new port specified by the BASE\_PORT:

For example:

`http://mms.example.com:11700`

4. Click *Save*.

**Step 4: 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 the Windows registry, update the Ops Manager URL in *Ops Manager Config* settings, 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: When you are done editing the settings, click *OK*, and exit the Registry Editor.**

**Step 5: Edit the Ops Manager URL to use the new port.**

1. In Ops Manager, click the *Admin* link in the upper right corner of the page.
2. Click the *General* tab and then click *Ops Manager Config*.
3. Update the *URL to Access Ops Manager* field to use the new port specified by `-Dbase-port` in the `\MMS\Parameters\Java\Options` registry value.

For example:

```
http://mms.example.com:11700
```

4. Click *Save*.

**Step 6: 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 by right-clicking and choosing *Restart*.

Ops Manager will restart. The Ops Manager interface should be accessible at the new URL and port combination.

### Enable the Health Check Endpoint

Ops Manager provides an HTTP health-check endpoint on port 8090 that is disabled by default. Before you enable the endpoint, ensure that port 8090 is available for use.

To enable the endpoint:

**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: Uncomment `DEBUG.PORT`.** To enable the endpoint, remove the # symbol so that the line reads:

```
DEBUG.PORT=8090
```

**Step 3: Save the changes.**

**Step 4: Restart Ops Manager.**

## 7.4 Configure SSL Connections to Ops Manager

### On this page

- [Overview](#)
- [Run the Ops Manager Application Over HTTPS](#)

### 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.

By default, Ops Manager uses port 8443 for HTTPS access.

### 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: Click *Admin* to open the administration interface.**

**Step 2: Click the *General* tab, then click *Ops Manager Config*.**

**Step 3: Configure the PEM key file and password.**

Set the following:

- *HTTPS PEM Key File* to a PEM file containing an X509 certificate and private key.
- *HTTPS PEM Key File Password* to the password for the certificate.

**Step 4: Configure the Ops Manager Application URL with the new HTTPS information.**

Set *URL to Access Ops Manager* to the new URL and port (8443) for HTTPS access. The following is an example:

`https://mms.example.net:8443`

## 7.5 Configure the Connections to the Backing MongoDB Instances

### On this page

- Overview
- Prerequisites
- Procedures

### 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.

### Prerequisites

This tutorial assumes you have:

- deployed the *Ops Manager Application Database*
- deployed *Backup Data Storage*
- configured both databases to use access control and/or SSL.

---

**Note:** 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 to be able to connect to the database.

**Step 1: Open the `conf-mms.properties` file with `root` (Linux) or `Administrator` (Windows) privileges.** This file configures Ops Manager's connection to the *Ops Manager Application Database*.

**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: Configure any other authentication mechanism-specific settings in `conf-mms.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 4: Restart all the Ops Manager instances, including those with the Backup Daemon enabled.** If the Ops Manager Application Database is running over SSL, proceed to the *SSL configuration tutorial*.

Restart Ops Manager using the appropriate command for your distribution:

Installed on Linux with DEB or RPM package:

```
sudo service mongodb-mms restart
```

Installed on Linux from an Archive:

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

Installed on Windows:

1. Click *Control Panel*.
2. Click *System and Security*.
3. Click *Administrative Tools*.
4. Click *Services*.

5. In the **Services** list, right-click on the *MongoDB Ops Manager HTTP Service* and click *Restart*.
6. Optionally, in the **Services** list, right-click on the *MongoDB Backup Daemon Service* and click *Restart*.

### Configure SSL Connections to the Ops Manager Application Database

**Step 1: Open the `conf-mms.properties` file with root (Linux) or Administrator (Windows) privileges.** This file configures Ops Manager's connection to the *Ops Manager Application Database*.

**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 User Credentials*.

---

**Important:** On Microsoft Windows servers, file paths for `mongodb.ssl.CAFile` and `mongodb.ssl.PEMKeyFile` must escape their backslashes to work properly. If you store your SSL certificates in `D:\Certificates`, the file path would be written as `D:\\Certificates\\pemkeyfile.pem`.

---

**Step 3: Restart all the Ops Manager instances, including those with the Backup Daemon enabled.** Restart Ops Manager using the appropriate command for your distribution:

Installed on Linux with DEB or RPM package:

```
sudo service mongodb-mms restart
```

Installed on Linux from an Archive:

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

Installed on Windows:

1. Click *Control Panel*.
2. Click *System and Security*.
3. Click *Administrative Tools*.
4. Click *Services*.
5. In the **Services** list, right-click on the *MongoDB Ops Manager HTTP Service* and click *Restart*.
6. Optionally, in the **Services** list, right-click on the *MongoDB Backup Daemon Service* and click *Restart*.

## 7.6 Enable SSL for a Deployment

### On this page

- [Overview](#)
- [Procedures](#)

### Overview

In order for Ops Manager to monitor, deploy, or back up a MongoDB deployment that uses SSL, you must enable SSL for the Ops Manager group. The SSL settings apply to **all** deployments managed by Ops Manager.

Starting with Ops Manager 1.8, Ops Manager automatically configures the Monitoring and Backup agents to connect to the managed deployment over SSL when you activate SSL for the Ops Manager group. You no longer need to manually configure the agents' SSL settings.

If you are not using automation for a deployment, you can still configure the monitoring and backup agents manually. See: [Configure Monitoring Agent for SSL](#) and [Configure Backup Agent for SSL](#) for more information.

If at any point you wish to reset the authentication settings for your group and start again, click *Clear Settings* in the *Authentication & SSL Settings* dialog box to clear all authentication and security settings, automation users, and automation roles. You **cannot** clear the authentication and security settings if there are managed processes in your deployment.

See [Clear Security Settings](#) for more information.

For information on other group-wide settings, see [Create a Group](#).

### Procedures

**Warning:** For MongoDB 2.6 and below, you must use the MongoDB Enterprise Edition, which includes SSL, or add a custom build with SSL enabled. To configure the available MongoDB versions, see: [Configure Available MongoDB Versions](#).

#### Ensure Existing Deployments are Using SSL

If you wish to enable SSL for an Ops Manager group that includes MongoDB deployments, use the following procedure to ensure that the MongoDB deployments are configured to use SSL:

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Select the process that you wish to modify.

1. Click the first icon to the right of *Processes* to switch to that view.
2. Click the process that you wish to edit.
3. Click the *wrench icon* to the right of that process to modify it.

**Step 3:** Expand the *Advanced Options* area.

#### **Step 4: Set the SSL startup options.**

1. Click *Add Option* to add each option.

Option	Value
sslmode	Select requireSSL.
sslPemKeyFile	Provide the path to the client certificate.
sslPemKeyPassword	If you encrypted the PEM key file, provide its password.

2. When you have added the required settings, click *Apply*.

#### **Enable SSL for the Group**

You can manage both SSL and non-SSL MongoDB deployments in the same group.

**Important:** Prior to Ops Manager version 2.0.3, if you enable SSL, all MongoDB deployments in the group that are managed by Ops Manager must use SSL.

#### **Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

#### **Step 2: Navigate to *Authentication & SSL Settings*.**

1. Click the *Ellipsis* icon at the top of the page.
2. Select *Authentication & SSL Settings*.
3. Click *Next*.

**Step 3: On the *Select Authentication Mechanisms* screen, click *Next*.** If you wish to *enable one or more Authentication Mechanisms* for your Ops Manager group, select them and then click *Next*.

Click *Next* to move to the SSL screen.

#### **Step 4: Toggle the *Enable SSL* slider to *Yes*.**

**Step 5: Specify the path to the SSL CA file and choose the Client Certificate Mode, then click *Continue*.** The SSL CA file is a .pem file that contains the root certificate chain from the Certificate Authority. The Monitoring and Backup Agents use the CA file for connections to your deployment.

The *Client Certificate Mode* specifies whether client certificates are required for each mongod and mongos in the deployment.

- *OPTIONAL:* Ops Manager starts each mongod and mongos process with both `net.ssl.CAFile` and `net.ssl.allowConnectionsWithoutCertificates`. As such, mongod and mongos processes need not possess client certificates.
- *REQUIRED:* Ops Manager starts each mongod and mongos with the `net.ssl.CAFile` setting. Each mongod and mongos must possess a client certificate.

**Step 6: Provide SSL credentials for the Ops Manager Agents** Specify the path to the .pem file that contains both the TLS/SSL certificate and key for each agent. If needed, specify the password to de-crypt the .pem certificate-key file.

Ensure you use the correct input box for your operating system.

**Step 7:** Click *Review & Deploy* to review your changes.

**Step 8: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 7.7 Configure users and groups using LDAP with Ops Manager

### On this page

- [Overview](#)
- [Prerequisites](#)
- [Procedure](#)

### Overview

You can configure Ops Manager to use a Lightweight Directory Access Protocol (LDAP) service to manage user authentication. Users log in through Ops Manager, then Ops Manager searches the LDAP directory for the user and synchronizes the user's name and email addresses in the Ops Manager user records with the values in the LDAP user records.

LDAP is now configured within Ops Manager and not through a properties file. Go to *Admin > General > Ops Manager Config > User Authentication* to configure these settings.

---

**Note:** This page is about user authentication into the Ops Manager web interface. Separately, if your MongoDB deployment uses LDAP for external authentication of database users, see the related page for creating MongoDB database users for the Ops Manager agents. For more information, see [Configure Monitoring Agent for LDAP](#) and [Configure Backup Agent for LDAP Authentication](#).

This tutorial describes how to configure LDAP authentication for Ops Manager, map LDAP groups to global Ops Manager roles and to group-level Ops Manager roles.

### User Authentication

When a user logs in, Ops Manager searches for a matching user using an LDAP query.

- Ops Manager logs into LDAP as the *search user*, using the credentials specified in the *LDAP Bind Dn* and *LDAP Bind Password* fields.
- Ops Manager searches only under the base distinguished name specified in the *LDAP User Base Dn* field and matches the user according to the LDAP attribute specified in the *LDAP User Search Attribute* field.
- If a matching user is found, Ops Manager authenticates the supplied password against the LDAP password for the provided user.

### 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.

LDAP entries map to Ops Manager records as follows:

LDAP	Ops Manager
User	User
Group	Role

To use LDAP groups effectively, [create additional groups](#) within Ops Manager to control access to specific deployments in your organization, such as creating separate Ops Manager groups for development and production environments. You can then map an LDAP group to a role in the Ops Manager group to provide access to a deployment.

---

**Note:** Changes made to LDAP groups can take up to an hour to appear in Ops Manager.

---

## LDAP Over SSL

Use of LDAP over an SSL connection (LDAPS), is specified in the `conf-mms.properties` file in the fields [LDAP SSL CA File](#), [LDAP SSL PEM Key File](#), and [LDAP SSL PEM Key File Password](#).

### Using LDAP from the Fresh Install vs. Converting to LDAP

All prerequisites apply to either scenario. The additional requirements are:

Fresh LDAP Install	Conversion to LDAP
The Global Owner to be the first user created.	The Global Owner exist in both LDAP and Ops Manager and belong to the LDAP group that will map to the Ops Manager Global Owner role.

**Important:** Once Ops Manager is converted to LDAP Authentication, only the user with the Global Owner role changing the authentication method remains logged into Ops Manager. All other users are logged off and need to log back into Ops Manager using their LDAP username and password. Any users without an LDAP username and password can no longer log into Ops Manager.

---

## Prerequisites

The LDAP server must:

- Be installed, configured and accessible to Ops Manager.
- Embed each user's group memberships as an attribute of each user's LDAP Entry.
- Include a user that can search the needed base distinguished name(s) that have the users and groups that use Ops Manager.
- Include a group that you can specify in the Ops Manager *LDAP Global Role Owner* field.
  - 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.

---

### Example

If LDAP has an `admin` group for use by Ops Manager administrators, enter `admin` in the *LDAP Global Role Owner* field.

---

## Procedure

To configure LDAP authentication:

**Step 1: Define your user records in the LDAP system of your choice.**

**Step 2: Navigate to the *User Authentication* tab of the *Ops Manager Config* page.**

1. Click *Admin* link at the upper right corner of the page.
2. Click the *General* tab.
3. Click the *Ops Manager Config* page.
4. Click *User Authentication* tab.

**Step 3: Enter LDAP configuration settings.**

1. Enter values for the following **required LDAP configuration** fields:

Field	Needed Value	Example Value
<i>User Authentication Method</i>	Change to <i>LDAP</i> .	LDAP
<i>LDAP URI</i>	The hostname and port of the LDAP server.	ldap://ldap.example.com:389
<i>LDAP SSL CA File</i>	The path to a PEM key file containing the certificate for the CA who signed the certificate used by the LDAPS server. This optional field is used by the Ops Manager application to verify the identify of the LDAPS server and prevent man-in-the-middle Attacks. If this configuration is not provided, Ops Manager uses the default root CA certificate bundle that comes with the Java Runtime Environment (JRE). If your LDAPS server certificate cannot be verified by a root CA (i.e. if it is self-signed), requests to the LDAPS server fail.	/opt/cert/ca.pem
<i>LDAP SSL PEM Key File</i>	The path to a PEM key file containing a client certificate and private key. This field is optional and should be used only if your LDAPS server requires client certificates be passed by client applications. This is used to sign requests sent from the Ops Manager application server to the LDAPS server. This allows the LDAPS server to verify the identify of Ops Manager application server.	/opt/cert/ldap.pem
<i>LDAP SSL PEM Key File Password</i>	This password decrypts the LDAP SSL PEM Key File. If your client certificates specified in the LDAP SSL PEM Key File field are required by the LDAPS server and if the client certificate specified in LDAP SSL PEM Key File is stored encrypted on the file system, this field is required.	<encrypted-password>
<i>LDAP Bind Dn</i>	A credentialed user on the LDAP server to conduct searches for users.	cn=admin, dc=example, dc=com
<i>LDAP Bind Password</i>	Password for the Bind Dn user on the LDAP server.	<password>
<i>LDAP User Base Dn</i>	The Distinguished Name that Ops Manager uses to search for users on the LDAP server.	dc=example, dc=com
<i>LDAP User Search Attribute</i>	The field in the LDAP server that specifies the username.	uid
<i>LDAP User Group</i>	The attribute of LDAP users that specifies the groups of which the user is a member.  <b>Important:</b> 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 <i>LDAP User Group</i> field. See <a href="#">Authentication through LDAP</a> for more information and examples.	memberof
<i>LDAP Global Role Owner</i>	The LDAP group to which Ops Manager Global Owners belong.	cn=global-owner, ou=groups, dc=example, dc=com

**Note:** Each Global Role group provides the members of its associated LDAP group or groups with an Ops Manager [global role](#). Global roles provide access to all the Ops Manager [groups](#) in the Ops Manager deploy-

ment.

- 
2. Enter values for the following **Optional LDAP Configuration** fields if needed:

Field	Needed Value
<i>LDAP User First Name</i>	The attribute of LDAP users that specifies the user's first name.
<i>LDAP User Last Name</i>	The attribute of LDAP users that specifies the user's last name.
<i>LDAP User Email</i>	The attribute of LDAP users that specifies the user's email address.
<i>LDAP Global Role Backup Admin</i>	The LDAP group to which Ops Manager Global Backup Administrators belong.
<i>LDAP Global Role Monitoring Admin</i>	The LDAP group to which Ops Manager Global Monitoring Administrators belong.
<i>LDAP Global Role User Admin</i>	The LDAP group to which Ops Manager Global User Administrators belong.
<i>LDAP Global Role Read Only</i>	The LDAP group to which Ops Manager Global Read Only Users belong.

**Step 4: Click Save.**

**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 *LDAP Global Role Owner* field.

Upon successful login, Ops Manager displays your groups page.

**Step 6: Associate LDAP groups with group-level roles.**

1. Click the *Settings* tab.
2. Click the *My Groups* page.
3. Click the *Add Group* button.
4. Enter a name for the new Ops Manager group and enter the LDAP groups that should provide the permissions for each *group-level role*.
5. Check the checkbox to agree to the terms of service.
6. Click *Add Group*.

**Step 7: Add your MongoDB deployments.**

Specify the LDAP authentication settings when *adding a MongoDB deployment*.

## 7.8 Enable Authentication for an Ops Manager Group

**MongoDB Access Control Overview** Edit authentication credentials for host.

**Enable Username/Password Authentication** Configure your Ops Manager group to use MONGODB-CR / SCRAM-SHA-1 authentication for communication between the Ops Manager Agents and your deployments.

**Enable LDAP Authentication** Configure your Ops Manager group to use LDAP (Plain) authentication for communication between the Ops Manager Agents and your deployments.

**Enable Kerberos Authentication** Configure your Ops Manager group to use Kerberos (GSSAPI) authentication for communication between the Ops Manager Agents and your deployments.

**Enable x.509 Authentication** Configure your Ops Manager group to use x.509 Client Certificate (MONGODB-X509) authentication for communication between the Ops Manager Agents and your deployments.

**Clear Security Settings** Clear all authentication and security-related settings for your Ops Manager deployment. You can only reset the security settings if there are not managed processes in your deployment.

## Configure MongoDB Authentication and Authorization

### On this page

- [Overview](#)
- [Access Control Mechanisms](#)
- [Edit Host Credentials](#)

### 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. If you are using Automation to manage your MongoDB deployments, you will enable and configure authentication through the Ops Manager interface.

If you are not using Automation to manage your MongoDB deployments, you must configure the Monitoring and Backup agents manually.

### Access Control Mechanisms

**MONGODB-CR/SCRAM-SHA-1** In MongoDB 3.0 and later, MongoDB’s default authentication mechanism is a challenge and response mechanism (SCRAM-SHA-1). Previously, MongoDB used MongoDB Challenge and Response (MONGODB-CR) as the default.

SCRAM-SHA-1 is an IETF standard, [RFC 5802](#), that defines best practice methods for implementation of challenge-response mechanisms for authenticating users with passwords.

MONGODB-CR is a challenge-response mechanism that authenticates users through passwords.

To enable MONGODB-CR when using Automation, see *Enable SCRAM-SHA-1 / MONGODB-CR Authentication for your Ops Manager Group*.

To configure the agents to authenticate as users with the proper access without Automation, see:

- [Configure Monitoring Agent for MONGODB-CR](#)
- [Configure Backup Agent for MONGODB-CR](#)

**LDAP** MongoDB Enterprise provides support for proxy authentication of users. This allows administrators to configure a MongoDB cluster to authenticate users by proxying authentication requests to a specified Lightweight Directory Access Protocol (LDAP) service.

To enable LDAP for your Ops Manager group when using Automation, see: [Enable LDAP Authentication for your Ops Manager Group](#).

To configure the agents to authenticate as users with the proper access without Automation, see:

- [Configure Monitoring Agent for LDAP](#)
- [Configure Backup Agent for LDAP Authentication](#)

**Kerberos** MongoDB Enterprise supports authentication using a Kerberos service. Kerberos is an industry standard authentication protocol for large client/server systems.

To use MongoDB with Kerberos, you must have a properly configured Kerberos deployment, configure *Kerberos service principals* for MongoDB, and add the *Kerberos user principal*. If you are using Automation, you can [Enable Kerberos Authentication for your Ops Manager Group](#) from within the Ops Manager interface.

To create a Kerberos Principal and the associated MongoDB user, and to configure the Monitoring and Backup Agents to authenticate as users with the proper access **without** Automation, see:

- [Configure the Monitoring Agent for Kerberos](#)
- [Configure the Backup Agent for Kerberos](#)

Specify Kerberos as the MongoDB process's authentication mechanism when *adding the deployment* or when *editing the deployment*.

**x.509** MongoDB supports x.509 certificate authentication for use with a secure [TLS/SSL connection](#). The x.509 client authentication allows clients to authenticate to servers with certificates rather than with a username and password.

In Ops Manager, x.509 Client Certificate (MONGODB-X509) is only available on MongoDB Enterprise builds. If you have existing deployments running on a MongoDB Community build, you must *upgrade them to MongoDB Enterprise* before you can enable x.509 Client Certificate (MONGODB-X509) for your Ops Manager group.

To enable x.509 authentication for your Ops Manager group when using Automation, see: [Enable x.509 Authentication for your Ops Manager Group](#).

---

**Note:** Ops Manager does **not** currently support using x.509 certificates for membership authentication.

---

## Edit Host Credentials

If your deployment is managed by Ops Manager, you will configure the deployment to use the authentication mechanism from the Ops Manager interface. The [Manage MongoDB Users and Roles](#) tutorials describe how to configure an existing deployment to use each authentication mechanism.

If your deployment is **not** managed by Ops Manager, manually configure the Monitoring and Backup agents with the proper credentials **before** you edit the host's authentication credentials.

---

### See

[Configure Monitoring Agent for Access Control](#) and [Configure Backup Agent for Access Control](#) describe how to configure the Monitoring and Backup agents for access control.

---

Once the Monitoring and Backup agents are correctly configured, you can edit the deployment's authentication credentials using the following procedures.

## Edit Credentials for Monitoring a Host

---

**Important:** Before editing these credentials, configure the Monitoring Agent with the proper credentials. See [Configure Monitoring Agent for Access Control](#).

---

To edit the credential for Monitoring:

**Step 1:** Click the *Deployment* tab, then click 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:

<i>Auth Mechanism</i>	The authentication mechanism used by the host. Can specify <a href="#">MONGODB-CR</a> , <a href="#">LDAP (PLAIN)</a> , or <a href="#">Kerberos(GSSAPI)</a> .
<i>Current DB Username</i>	If the authentication mechanism is MONGODB-CR or LDAP, the username used to authenticate the Monitoring Agent to the MongoDB deployment. See <a href="#">Configure Monitoring Agent for MONGODB-CR</a> , <a href="#">Configure Monitoring Agent for LDAP</a> , or <a href="#">Configure the Monitoring Agent for Kerberos</a> for setting up user credentials.
<i>Current DB Password</i>	If the authentication mechanism is MONGODB-CR or LDAP, the password used to authenticate the Monitoring Agent to the MongoDB deployment. See <a href="#">Configure Monitoring Agent for MONGODB-CR</a> , <a href="#">Configure Monitoring Agent for LDAP</a> , or <a href="#">Configure the Monitoring Agent for Kerberos</a> for setting up user credentials.
<i>Update other hosts in replica set/sharded cluster as well</i>	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 Credentials for Backing up a Host

---

**Important:** Before editing these credentials, configure the Backup Agent with the proper credentials. See [Configure Backup Agent for Access Control](#).

---

To edit the credential for Backup:

**Step 1:** Select the *Backup* tab and then *Overview* page.

**Step 2:** On the line listing the process, click the ellipsis icon and click *Edit Credentials*.

**Step 3: Enter the credentials.** Edit the following information, as appropriate:

<i>Auth Mechanism</i>	The authentication mechanism the host uses. The options are: <ul style="list-style-type: none"><li>• <i>Username/Password</i></li><li>• <i>Kerberos</i> (Enterprise Only)</li><li>• <i>LDAP</i> (Enterprise Only)</li><li>• <i>X.509 Client Certificate</i></li></ul>
<i>DB Username</i>	For Username/Password or LDAP authentication, the username used to authenticate the Backup Agent to the MongoDB deployment. See <a href="#">Configure Backup Agent for MONGODB-CR</a> or <a href="#">Configure Backup Agent for LDAP Authentication</a> .
<i>DB Password</i>	For Username/Password or LDAP authentication, the password used to authenticate the Backup Agent to the MongoDB deployment.
<i>Allows SSL for connections</i>	If checked, the Backup Agent uses SSL to connect to MongoDB. See <a href="#">Configure Backup Agent for SSL</a> .

**Step 4: Click Save.**

#### Enable SCRAM-SHA-1 / MONGODB-CR Authentication for your Ops Manager Group

##### On this page

- [Overview](#)
- [Considerations](#)
- [Procedure](#)

##### Overview

Ops Manager enables you to configure the Authentication Mechanisms that the Ops Manager Agents use to connect to your MongoDB deployments from within the Ops Manager interface. You can enable multiple authentication mechanisms for your group, but you must choose a single mechanism for the Agents to use to authenticate to your deployment.

In MongoDB 3.0 and later, MongoDB's default authentication mechanism is a challenge and response mechanism (SCRAM-SHA-1). Previously, MongoDB used MongoDB Challenge and Response (MONGODB-CR) as the default.

**SCRAM-SHA-1** SCRAM-SHA-1 is an IETF standard, [RFC 5802](#), that defines best practice methods for implementation of challenge-response mechanisms for authenticating users with passwords.

SCRAM-SHA-1 verifies supplied user credentials against the user's name, password and database. The user's database is the database where the user was created, and the user's database and the user's name together serves to identify the user.

**MONGODB-CR** MONGODB-CR is a challenge-response mechanism that authenticates users through passwords.

MONGODB-CR verifies supplied user credentials against the user's name, password and database. The user's database is the database where the user was created, and the user's database and the user's name together serve to identify the user.

## Considerations

This tutorial describes how to enable Username/Password (MONGODB-CR/SCRAM-SHA-1) authentication for your Ops Manager deployment. Username/Password authentication is the only authentication mechanism available in Ops Manager when using the MongoDB Community version.

If at any point you wish to reset the authentication settings for your group and start again, click *Clear Settings* in the *Authentication & SSL Settings* dialog box to clear all authentication and security settings, automation users, and automation roles. You **cannot** clear the authentication and security settings if there are managed processes in your deployment.

See [Clear Security Settings](#) for more information.

## Procedure

This procedure describes how to configure and enable MONGODB-CR / SCRAM-SHA-1 authentication when using Automation. If your Monitoring or Backup agents are **not** managed by Ops Manager, you must manually configure them to use MONGODB-CR / SCRAM-SHA-1. See: [Configure Monitoring Agent for MONGODB-CR](#) and [Configure Backup Agent for MONGODB-CR](#) for instructions.

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Navigate to *Authentication & SSL Settings*.

1. Click the *Ellipsis* icon at the top of the page.
2. Select *Authentication & SSL Settings*.
3. Click *Next*.

**Step 3:** Check *Username/Password (MONGODB-CR/SCRAM-SHA-1)*, then click *Next*.

**Step 4:** Configure SSL if desired, and click *Continue*. If desired, enable SSL for the group.

---

**Note:** See [Enable SSL for a Deployment](#) for SSL setup instructions.

---

SSL is not required for use with *Username/Password (MONGODB-CR/SCRAM-SHA-1)* authentication.

**Step 5: Configure the Authentication Mechanism for the Agents.** If you enable more than one authentication mechanism, you must specify which one of the authentication mechanisms the Ops Manager agents should use to connect to your deployment.

1. Select *Username/Password (MONGODB-CR/SCRAM-SHA-1)* from the *Agent Auth Mechanism* drop-down menu.

2. Ops Manager automatically generates the Agents' usernames and passwords.

Ops Manager creates users for the agents with the required user roles in the admin database for each existing deployment in Ops Manager. When you add a new deployment, Ops Manager creates the required users in the new deployment.

3. Click *Save*.

You do not need to configure all of the agents, only the ones you installed.

---

#### Example

If you did not install the Backup agent, you do not need to configure the Backup agent.

---

#### **Step 6: Click *Review & Deploy* to review your changes.**

#### **Step 7: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

### **Enable LDAP Authentication for your Ops Manager Group**

#### On this page

- Considerations
- Procedure

Ops Manager enables you to configure the Authentication Mechanisms that the Ops Manager Agents use to connect to your MongoDB deployments from within the Ops Manager interface. You can enable multiple authentication mechanisms for your group, but you must choose a single mechanism for the Agents to use to authenticate to your deployment.

MongoDB Enterprise provides support for proxy authentication of users. This allows administrators to configure a MongoDB cluster to authenticate users by proxying authentication requests to a specified Lightweight Directory Access Protocol (LDAP) service.

LDAP (Plain) is only available on MongoDB Enterprise builds. If you have existing deployments running on a MongoDB Community build, you must *upgrade them to MongoDB Enterprise* before you can enable LDAP (Plain) for your Ops Manager group.

#### **Considerations**

MongoDB Enterprise for Windows does **not** include LDAP support for authentication. However, MongoDB Enterprise for Linux supports using LDAP authentication with an ActiveDirectory server.

MongoDB does not support LDAP authentication in mixed sharded cluster deployments that contain both version 2.4 and version 2.6 shards.

The [Authenticate Using SASL and LDAP with ActiveDirectory](#) and [Authenticate Using SASL and LDAP with OpenLDAP](#) tutorials in the MongoDB manual provide more information about setting up LDAP and SASL for MongoDB. Setting up LDAP and SASL is beyond the scope of this document.

## Procedure

This procedure describes how to configure and enable LDAP authentication when using Automation. If your Monitoring or Backup agents are **not** managed by Ops Manager, you must manually configure them to use LDAP. See: [Configure Monitoring Agent for LDAP](#) and [Configure Backup Agent for LDAP Authentication](#) for instructions.

If at any point you wish to reset the authentication settings for your group and start again, click *Clear Settings* in the *Authentication & SSL Settings* dialog box to clear all authentication and security settings, automation users, and automation roles. You **cannot** clear the authentication and security settings if there are managed processes in your deployment.

See [Clear Security Settings](#) for more information.

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Navigate to *Authentication & SSL Settings*.

1. Click the *Ellipsis* icon at the top of the page.
2. Select *Authentication & SSL Settings*.
3. Click *Next*.

**Step 3:** Check *LDAP (PLAIN)*, then click *Next*.

**Step 4:** Configure SSL if desired, and click *Continue*. If desired, enable SSL for the group.

---

**Note:** See [Enable SSL for a Deployment](#) for SSL setup instructions.

---

SSL is not required for use with *LDAP (PLAIN)* authentication.

**Step 5: Configure the Authentication Mechanism for the Agents.** If you enable more than one authentication mechanism, you must specify which one of the authentication mechanisms the Ops Manager agents should use to connect to your deployment.

1. Select *LDAP (PLAIN)* from the *Agent Auth Mechanism* drop-down menu.
2. For each Agent, provide:

Setting	Value
<Agent> LDAP Username	Enter the LDAP username.
<Agent> LDAP Password	Enter the password for Agent's LDAP Username.

3. Click *Save*.

You do not need to configure all of the agents, only the ones you installed.

---

## Example

If you did not install the Backup agent, you do not need to configure the Backup agent.

---

**Step 6:** Click *Review & Deploy* to review your changes.

**Step 7: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Enable Kerberos Authentication for your Ops Manager Group

### On this page

- [Overview](#)
- [Considerations](#)
- [Procedures](#)

### Overview

Ops Manager enables you to configure the Authentication Mechanisms that the Ops Manager Agents use to connect to your MongoDB deployments from within the Ops Manager interface. You can enable multiple authentication mechanisms for your group, but you must choose a single mechanism for the Agents to use to authenticate to your deployment.

[MongoDB Enterprise](#) supports authentication using a Kerberos service. Kerberos is an industry standard authentication protocol for large client/server systems.

To use MongoDB with Kerberos, you must have a properly configured Kerberos deployment, configure *Kerberos service principals* for MongoDB, and add the *Kerberos user principal*. The [Kerberos Authentication](#) section of the MongoDB Manual provides more detail about using MongoDB with Kerberos.

### Considerations

Kerberos (GSSAPI) is only available on MongoDB Enterprise builds. If you have existing deployments running on a MongoDB Community build, you must [upgrade them to MongoDB Enterprise](#) before you can enable Kerberos (GSSAPI) for your Ops Manager group.

This tutorial describes how to enable Kerberos for your Ops Manager group, and how to configure your Ops Manager Agents to connect to your Kerberized deployment.

Setting up and configuring a Kerberos deployment is beyond the scope of this document. This tutorial assumes you have configured a Kerberos service principal for each Agent and you have a valid keytab file for each Agent.

If at any point you wish to reset the authentication settings for your group and start again, click *Clear Settings* in the *Authentication & SSL Settings* dialog box to clear all authentication and security settings, automation users, and automation roles. You **cannot** clear the authentication and security settings if there are managed processes in your deployment.

See [Clear Security Settings](#) for more information.

### Procedures

This procedure describes how to configure and enable Kerberos authentication when using Automation. If your Monitoring or Backup agents are **not** managed by Ops Manager, you must manually configure them to use Kerberos. See: [Configure the Monitoring Agent for Kerberos](#) and [Configure the Backup Agent for Kerberos](#) for instructions.

**Configure an Existing Deployment for Kerberos-based Authentication** If you have one or more existing deployments managed by Ops Manager, the MongoDB deployment must be configured for Kerberos authentication before you can enable Kerberos authentication for your group.

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Select the process that you wish to modify.

1. Click the first icon to the right of *Processes* to switch to that view.
2. Click the process that you wish to edit.
3. Click the *wrench icon* to the right of that process to modify it.

**Step 3:** Expand the *Advanced Options* area.

**Step 4: Set the `kerberosKeytab` Startup option to point to the keytab file and click *Apply*.** If `kerberosKeytab` is not already set, use the *Add Option* button to add a new startup option, and select `kerberosKeytab` from the drop-down menu. Input the path to the keytab file as the value, and then click *Apply*.

When you have configured the Kerberos options for each deployment, you can proceed to enable Kerberos for your Ops Manager group.

### Enable Kerberos for your Ops Manager Group

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Navigate to *Authentication & SSL Settings*.

1. Click the *Ellipsis* icon at the top of the page.
2. Select *Authentication & SSL Settings*.
3. Click *Next*.

**Step 3:** Check *Kerberos (GSSAPI)*, then click *Next*.

**Step 4: Configure SSL if desired, and click *Continue*.** If desired, enable SSL for the group.

---

**Note:** See [Enable SSL for a Deployment](#) for SSL setup instructions.

---

SSL is not required for use with *Kerberos (GSSAPI)* authentication.

**Step 5: Configure the Authentication Mechanism for the Agents.** If you enable more than one authentication mechanism, you must specify which one of the authentication mechanisms the Ops Manager agents should use to connect to your deployment.

1. Select *Kerberos (GSSAPI)* from the *Agent Auth Mechanism* drop-down menu.

2. For each Agent, provide:

Setting	Value
<Agent> Kerberos Principal	Enter the Kerberos Principal.
<Agent> Keytab path	Provide the path and filename for the Agent's Keytab.

3. Click *Save*.

You do not need to configure all of the agents, only the ones you installed.

#### Example

If you did not install the Backup agent, you do not need to configure the Backup agent.

#### Step 6: Click *Review & Deploy* to review your changes.

#### Step 7: Review and approve your changes.

Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

### Enable x.509 Authentication for your Ops Manager Group

#### On this page

- Overview
- Prerequisites
- Procedures

#### Overview

Ops Manager enables you to configure the Authentication Mechanisms that the Ops Manager Agents use to connect to your MongoDB deployments from within the Ops Manager interface. You can enable multiple authentication mechanisms for your group, but you must choose a single mechanism for the Agents to use to authenticate to your deployment.

MongoDB supports x.509 client and member certificate authentication for use with a secure [TLS/SSL connection](#). The x.509 authentication allows users and other members to authenticate to servers with certificates rather than with a username and password.

In Ops Manager, x.509 Client Certificate (MONGODB-X509) is only available on MongoDB Enterprise builds. If you have existing deployments running on a MongoDB Community build, you must [upgrade them to MongoDB Enterprise](#) before you can enable x.509 Client Certificate (MONGODB-X509) for your Ops Manager group.

**Note:** As of Ops Manager 2.0, using x.509 certificates for membership authentication is supported.

#### Prerequisites

**Important:** A full description of Public Key Infrastructure (PKI), including certificates and Certificate Authorities,

is beyond the scope of this tutorial. This tutorial assumes prior knowledge of SSL and PKI as well as access to valid x.509 certificates.

---

To enable x.509 Authentication for Ops Manager, you must:

- Obtain valid certificate files (PEM Key files) generated and signed by a single certificate authority (CA) for:
    - Certificate Only in PEM file:
      - \* The CA itself
    - Certificate and Private Key in PEM file:
      - \* Each agent (Automation, Monitoring if used and Backup if used)
      - \* Each hostname of a managed MongoDB server (if member X.509 authentication is used)
      - \* Each client that may attach to the MongoDB instance
  - Generate LDAPv3 distinguished name from each Agents' PEM Key file. Consult the documentation for whichever SSL implementation you use.
- 

### Example

For OpenSSL, the command to generate the LDAPv3 DN from a PEM Key File called `automation.pem` is:

```
openssl x509 -in automation.pem -inform PEM -subject -nameopt RFC2253
```

---

**Note:** See [Client x.509 Certificate](#) in the MongoDB Manual for certificate requirements.

---

**Important:** If at any point you wish to reset the authentication settings for your group and start again, see [Clear Security Settings](#) for more information.

---

## Procedures

These procedures describe how to configure and enable x.509 authentication when using Automation. If Ops Manager does *not* manage your Monitoring or Backup agents, you must manually configure them to use x.509 authentication.

---

**Note:** See [Configure the Monitoring Agent User for x.509 Client Certificate Authentication](#) and [Configure Backup Agent User for x.509 Client Certificate Authentication](#) for instructions.

---

### Prepare an Existing Deployment for x.509 Certificate Authentication

**Important:** X.509 Client Certificate authentication requires SSL. If Ops Manager manages one or more existing deployments, SSL must be enabled on each process in the MongoDB deployment before enabling x.509 authentication.

---

**Note:** If SSL is already enabled, you may skip this procedure.

---

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Select the process that you wish to modify.**

1. Click the first icon to the right of *Processes* to switch to that view.

2. Click the process that you wish to edit.
3. Click the *wrench icon* to the right of that process to modify it.

**Step 3: Expand the *Advanced Options* area.**

**Step 4: Set the SSL startup options.**

1. Click *Add Option* to add each option.

Option	Value
sslmode	Select requireSSL.
sslPemKeyFile	Provide the path to the client certificate.
sslPemKeyPassword	If you encrypted the PEM key file, provide its password.

2. When you have added the required settings, click *Apply*.

**Configure an Existing Deployment for x.509 Member Certificate Authentication**

---

**Note:** This procedure is optional. It enables members of a replica set or sharded cluster to also use x.509 certificates to authenticate each other. If it is not configured, replica set and sharded cluster members can still authenticate with each other using *keyFile* authentication.

---

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Select the process that you wish to modify.**

1. Click the first icon to the right of *Processes* to switch to that view.
2. Click the process that you wish to edit.
3. Click the *wrench icon* to the right of that process to modify it.

**Step 3: Expand the *Advanced Options* area.**

**Step 4: Set the x.509 startup options.**

1. Click *Add Option* to add each option.

Option	Value
clusterAuthMode	Select x509.
clusterFile	Provide the path to the member PEM Key file.

2. When you have added the required settings, click *Apply*.

When you have configured the SSL options for each deployed process, you can proceed to enable x.509 authentication for your Ops Manager group.

**Enable x.509 Client Certificate Authentication for your Ops Manager Group**

**Step 1: Click the *Deployment* tab, then click the *Deployment* page.**

**Step 2: Navigate to *Authentication & SSL Settings*.**

1. Click the *Ellipsis* icon at the top of the page.
2. Select *Authentication & SSL Settings*.
3. Click *Next*.

**Step 3: Check *X.509 Client Certificate (MONGODB-X509)*, then click *Next*.**

**Step 4: Enable and configure SSL.**

1. Provide the following settings:

Setting	Value
<i>Enable SSL</i>	Select Yes.
<i>SSL CA File Path</i>	Provide the path on the server to the certificate authority PEM Key file.
<i>Client Certificate Mode</i>	Select REQUIRED.

2. Click *Next*.

**Step 5: Configure the Authentication Mechanism for the Agents.** If you enable more than one authentication mechanism, you must specify which one of the authentication mechanisms the Ops Manager agents should use to connect to your deployment.

1. Select *X.509 Client Certificate (MONGODB-X509)* from the *Agent Auth Mechanism* drop-down menu.
2. For each Agent, provide:

Setting	Value
<i>&lt;Agent&gt; Username</i>	Enter the LDAPv3 distinguished name derived from the Agent's PEM Key file.
<i>&lt;Agent&gt; PEM Key file</i>	Provide the path and filename for the Agent's PEM Key file on the server on the line for the appropriate operating system.
<i>&lt;Agent&gt; PEM Key Password</i>	Provide the password to the PEM Key file if it was encrypted.

3. Click *Save*.

You do not need to configure all of the agents, only the ones you installed.

---

**Example**

If you did not install the Backup agent, you do not need to configure the Backup agent.

---

**Step 6: Click *Review & Deploy* to review your changes.**

**Step 7: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Clear Security Settings

### On this page

- Overview
- Procedure

### Overview

Ops Manager enables you to *configure the security settings* that your deployments use through the Ops Manager user interface. If you wish to reset the security settings for your deployment, you may do so using the *Clear Settings* button. *Clear Settings* clears all authentication-related settings so you can start over from a blank configuration.

*Clear Settings* removes all authentication-related settings, including the authentication mechanisms for the deployment, automation users and roles, and defined users for the Monitoring and Backup agents.

You **cannot** reset the security settings for a deployment if your deployment includes any managed processes.

### Procedure

You may only reset the authentication settings if there are no managed processes associated with your deployment. As such, if you wish to reset the security settings, you must unmanage all processes, and then use the *Clear Settings* button to reset the security settings.

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Click the first of the two *Processes* icons.

**Step 3:** On the line listing the cluster, replica set, or process, click the ellipsis icon and select the option to remove it.

**Step 4:** Select whether to also stop monitoring the process. Select *Unmanage this item but continue to monitor* and click *Remove*. If prompted for an authentication code, enter the code.

You do not need to stop monitoring a process in order to reset the security settings.

**Step 5:** On the *Deployment* page, click the ellipsis icon at the top of the page and select *Authentication & SSL Settings*.

**Step 6:** Click *Clear Settings*. The *Clear Settings* button resets the security settings for your Ops Manager deployment. If you still have managed processes, the *Clear Settings* button is grayed out.

**Step 7:** To confirm that you wish to clear the security settings, click *Clear Settings*.

**Step 8:** Click *Review & Deploy* to review your changes.

**Step 9: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

Ops Manager will reset all security settings. You can now edit the security settings if you wish to. To resume management of any processes, see: *Add Monitored Processes to Automation*.

## 7.9 Manage Two-Factor Authentication for Ops Manager

### On this page

- [Overview](#)
- [Procedures](#)

### 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 `conf-mms.properties` configuration file can enable two-factor authentication through the file's `mms.multiFactorAuth.level` setting. Administrators can also enable two-factor authentication to use Twilio to send verification codes to users via SMS or voice call.

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. Resetting 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* for more information.

**Step 2: Set the `mms.multiFactorAuth.level` property to OPTIONAL, REQUIRED, or REQUIRED\_FOR\_GLOBAL\_ROLES.**

```
mms.multiFactorAuth.level=REQUIRED
```

When `mms.multiFactorAuth.level` is OPTIONAL, users can choose to set up two-factor authentication for their Ops Manager account.

When `mms.multiFactorAuth.level` is REQUIRED, all users **must** set up two-factor authentication.

When `mms.multiFactorAuth.level` is REQUIRED\_FOR\_GLOBAL\_ROLES, users who possess a *global role* must set up two-factor authentication, while two-factor authentication is optional for all other users.

### **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**

Resetting 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.

## 8 Groups and Users

**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.

### 8.1 Manage Groups

**Create a Group** Create a new Ops Manager group.

**Edit a Group's Configuration** Modify a group's settings.

**Remove a Group** Remove a group.

#### Create a Group

##### On this page

- [Overview](#)
- [Working with Multiple Environments](#)
- [Create a Group](#)
- [Additional Information](#)

#### Overview

A group provides access to a distinct deployment of MongoDB clusters, replica sets and standalone processes. Users must be members of a group to access the group's deployment. When you register as a new Ops Manager user, you must choose whether to join an existing group or create a new one.

Each group must have a globally unique name within the Ops Manager platform. Once you name a group, the group's name cannot be changed. The user who creates a group automatically has the *Owner* role for the group and can manage user access.

You can create multiple groups. Each group has a unique agent API key that is shared by the Monitoring, Backup and Automation Agents. Within each group, the Monitoring Agent must be able to connect to all hosts it monitors. If you have segregated environments separated by firewalls, you must create separate Ops Manager groups for each environment.

## 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.

### Create a Group

Choose a globally unique name for your group; this name must be unique within the Ops Manager platform. For security and auditing reasons, you cannot use a name used earlier, and once you name a group, the group's name cannot be changed.

Ops Manager automatically adds you as the first user to the group and assigns the group a set of default *alert configurations*.

**Step 1:** Click the *Settings* tab, then the *My Groups* page.

**Step 2:** Click *Add Group*.

**Step 3:** Follow the prompts to create the group.

### Additional Information

Use the following procedures to modify groups:

- *Edit a Group's Configuration*
- *Remove a Group*

### Edit a Group's Configuration

#### On this page

- Overview
- Edit Group Settings
- View Agent Status

#### Overview

You can configure your Ops Manager groups from the *Settings* tab. The tab gives access to group's settings, users and agents. To access group settings and agents, see the procedures on this page. To access the group's users, see *Manage Ops Manager Users and Roles*.

## Edit Group Settings

To modify group settings, click the *Settings* tab, then click *Group Settings*. For descriptions of the settings, see [Group Settings](#).

If you have *Global Owner* access, Ops Manager 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 *Settings* tab's *Group Settings* page apply to all users in the group:

Setting	Description
<i>Group Time Zone</i>	Sets your group's time zone.
<i>Collect Logs For All Hosts</i>	Activates or deactivates the collection of log data for all hosts. This overwrites the statuses set on the individual hosts.
<i>Collect Profiling Information for All Hosts</i>	Activates or deactivates Ops Manager collection of data from the MongoDB <a href="#">database profilers</a> running on your mongod instances. A mongod instance must have its profiler enabled in order for Ops Manager to collect data from it.
	When you change this setting, Ops Manager applies the change globally to all mongod processes in the group. For example, if you disable this setting, Ops Manager disables the collection of profiling data for all the group's processes. This setting does not affect whether the profiler is enabled on a given mongod process, only whether Ops Manager collects profiling data. To enable the collection of profiling data on a process-by-process basis, see <a href="#">Profile Databases</a> .
	When profiling is enabled, Ops Manager collects data from MongoDB's profiler to provide statistics about performance and database operations. Ensure exposing profile data to Ops Manager is consistent with your information security practices. Also be aware the profiler can consume resources which may adversely affect MongoDB performance.
	For more information, see <a href="#">Profile Databases</a> .
<i>Collect Database Specific Statistics</i>	Allows you to enable or disable the collection of database statistics. For more information, see <a href="#">How does Ops Manager gather database statistics?</a> .
<i>Reset Duplicates</i>	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.
<i>Preferred Hostnames</i>	Allows you to specify resolvable hostnames or IP address for your deployment's servers. Ops Manager keeps a list of the multiple ways to which each server is referred (hostname, FQDN, IP address) and uses heuristics to determine the best choice. Use this setting to guarantee Ops Manager uses a resolvable method. The method you choose will also be the method used to display the servers in Ops Manager.
	To specify a preferred hostname, click <i>Add</i> and do one of the following:
	<ul style="list-style-type: none"> <li>• To specify hostnames that end with a particular string, click the <i>Ends With</i> button and enter the string.</li> <li>• To specify hostnames that match a pattern, click the <i>Regex</i> button and enter a regular expression. An expression that uses “starts with” behavior must have <code>.*</code> at the end in order to correctly match. For example, to specify hostnames that start with <code>acme-</code>, enter: <code>^acme-.*</code></li> </ul>
<i>SUPPRESS MONGOS AUTOMATIC DISCOVERY</i>	Suppresses automatic discovery of all mongos processes in your deployment's sharded clusters.
<b>234</b>	If you use Ops Manager <a href="#">Backup</a> , this setting allows you to generate a public key for SCP backup restoration. If you restore a snapshot through SCP, Ops Manager uses the public key to verify the integrity of the data.
<i>Public Key for SCP Restores</i>	

**Admin-Only Group Settings** The following group settings in the *Admin Only* section of the *Settings* 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:

Setting	Description
<i>Mongos Deactivation Threshold</i>	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.
<i>Monitoring Agent Log Level</i>	Change the verbosity of the Monitoring Agent log.
<i>Automation Agent Log Level</i>	Change the verbosity of the Automation Agent log.

### **View Agent Status**

Click *Settings*, then *Agents* to display the following information about your agents:

Field	Description
<i>Status</i>	The time of the last ping from the agent.
<i>Type</i>	The type of agent.
<i>Host-name</i>	The hostname for the agent and any warnings, such as that the agent is down or out-of-date.
<i>State</i>	Indicates whether the agent is active.
<i>Ping Count</i>	The number of pings (i.e. data payloads) sent by the agent since midnight GMT. Typically agents send pings every minute.
<i>Version</i>	The version of the agent software running on this agent instance.
<i>Log</i>	Click <i>view logs</i> 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.

### **Remove a Group**

Please contact your Ops Manager administrator to remove a company or group from your Ops Manager account.

If you have administrative privileges, you can remove a group through system administration. Click *Admin*, then *General*, and then *Groups*.

## **8.2 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. See *Configure users and groups using LDAP with 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
- Procedures

### 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 *Settings* 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 *Settings* 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 *Settings* 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 *Settings* 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 *Settings* tab and then select the *Users* page.

**Step 2:** Click the trash can icon to the right of the user.

**Step 3:** 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 *Settings* 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 *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.

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](#)
- [Group Roles](#)

## 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.

## Group Roles

The following roles grant privileges within a group.

**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 the namespaces filter.
- 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, and the following additional privileges available only to Owner:

- Set up the *Backup* service.
- Update billing information.
- Enable the Public API.

## 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.

## 8.3 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
- Considerations
- Enable MongoDB Access Control
- Next Steps

#### 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. Ops Manager also creates users for the Monitoring and Backup agents if they are managed by Ops Manager. 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

Ops Manager supports various authentication mechanisms. You can choose which mechanisms you wish to use when you enable access control / authentication.

**Step 1:** Click the *Deployment* tab, then click the *Deployment* page.

**Step 2:** Navigate to *Authentication & SSL Settings*.

1. Click the *Ellipsis* icon at the top of the page.
2. Select *Authentication & SSL Settings*.
3. Click *Next*.

**Step 3:** Check the authentication mechanism, then click *Next*.

**Step 4:** Configure SSL if desired, and click *Continue*. If desired, enable SSL for the group.

---

**Note:** See [Enable SSL for a Deployment](#) for SSL setup instructions.

---

**Step 5: Configure the Authentication Mechanism for the Agents.** If you enable more than one authentication mechanism, you must specify which one of the authentication mechanisms the Ops Manager agents should use to connect to your deployment.

1. Select the authentication mechanism from the *Agent Auth Mechanism* drop-down menu.

Ops Manager automatically generates the Agents' usernames and passwords.

Ops Manager creates users for the agents with the required user roles in the admin database for each existing deployment in Ops Manager. When you add a new deployment, Ops Manager creates the required users in the new deployment.

You do not need to configure all of the agents, only the ones you installed.

---

### Example

If you did not install the Backup agent, you do not need to configure the Backup agent.

---

## **Step 6: Click *Review & Deploy* to review your changes.**

## **Step 7: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

MONGODB-CR/SCRAM-SHA-1 authentication is the only authentication mechanism available in Ops Manager when using the MongoDB Community version.

For detailed instructions for configuring the different authentication mechanisms, see: [Enable Authentication for an Ops Manager Group](#)

## **Next Steps**

To create your users and assign privileges, see [Manage MongoDB Users and Roles](#).

## **Manage MongoDB Users and Roles**

### **On this page**

- [Overview](#)
- [Considerations](#)
- [Procedures](#)

### **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 access or after, but the users are not created until you enable access control. Your MongoDB instances will not require user credentials if access control is not 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, while Ops Manager users access Ops Manager groups.

### **Considerations**

If you want Ops Manager to ensure that all deployments in a group have the same database users, use only the Ops Manager interface to manage users.

If you want certain deployments in a group to possess users not set at the group level, you can add them through direct connection to the MongoDB instances. The [MongoDB manual](#) describes how to add users from the mongo shell.

Unlike manually-created users, if you create custom roles through a direct connection to the MongoDB instances, Ops Manager will delete these roles. You must use the Ops Manager interface to create custom roles for your managed MongoDB deployments.

## Procedures

### Add a MongoDB User

**Step 1:** From the *Deployment* tab, select the *MongoDB Users* page.

**Step 2:** Click 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.

You can add users to the \$external database, much as you would to any other database. The \$external database allows MongoDB instances to consult an external source, such as Kerberos or an LDAP server, to authenticate. As such, you do not need to specify a password for the users that you add to \$external.

**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* to review your changes.

**Step 7:** Review and approve your changes. Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

### Edit a User's Roles

**Step 1:** From the *Deployment* tab, select the *MongoDB Users* page.

**Step 2:** Click the user's gear icon and select *Edit*.

**Step 3:** Edit the user's information. In the *Roles* list, you can both add and delete roles. The *Roles* list provides a drop-down as you start typing the name of the role. You can add both user-defined roles and built-in roles.

**Step 4:** Click *Save Changes*.

**Step 5:** Click *Review & Deploy* to review your changes.

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Remove a MongoDB User

**Step 1:** From the *Deployment* tab, select the *MongoDB Users* page.

**Step 2:** Click the user's gear icon and select *Remove*.

**Step 3:** To confirm, click *Delete User*.

**Step 4:** Click *Review & Deploy* to review your changes.

**Step 5: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## Manage Custom Roles

### On this page

- Overview
- Considerations
- Prerequisite
- Procedures

### 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](#).

## Considerations

Use only the Ops Manager interface to manage users and roles. Do not do so through direct connection to a MongoDB instance.

## Prerequisite

MongoDB access control *must be enabled* to apply roles. You can create roles before enabling access control or after, but they don't go into effect until you enable access control.

## Procedures

### Create a Custom MongoDB Role

**Step 1:** From the *Deployment* tab, select the *MongoDB Roles* page.

**Step 2:** Select the *Add Role* button.

**Step 3:** In the *Identifier* field, 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. Select the database from the drop-down menu. 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* to review your changes.

**Step 8:** Review and approve your changes. Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

**Edit a Custom Role** You can change a custom role's privileges. You cannot change its name or database.

**Step 1:** From the *Deployment* tab, select the *MongoDB Roles* page.

**Step 2:** Click the role's gear icon and select *Edit*.

**Step 3: Add or Remove privileges for that role.** 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. Select the database from the drop-down menu. 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.

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: Click *Save Changes*.**

**Step 5: Click *Review & Deploy* to review your changes.**

**Step 6: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

**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:** From the *Deployment* tab, select the *MongoDB Roles* page.

**Step 2:** Click the role's gear icon and select *Remove*.

**Step 3:** To confirm, click *Delete Role*.

**Step 4: Click *Review & Deploy* to review your changes.**

**Step 5: Review and approve your changes.** Ops Manager displays your proposed changes.

1. If they are acceptable, click *Confirm & Deploy*.
2. If they are unacceptable, click *Cancel* and you can make additional changes.

## 9 Account Management

**Access Your User Account** Access your user account settings.

**Edit Your User Account** Update account settings and personal preferences.

**Set up Two-Factor Authentication** Describes two-factor authentication, which is required for Ops Manager.

### 9.1 Access Your User Account

#### On this page

- Overview
- Account
- Personalization
- Public API Access
- My Groups

#### Overview

You can access your user account from the *Settings* tab. Modify your account using the pages described here.

#### Account

The *Account* page allows you to update your personal information. Access the page through the *Settings* tab. The page includes the settings described here.

<i>User Name</i>	Displays your username. You cannot change your username.
<i>Email Address</i>	Displays the email address Ops Manager associates with your account.
<i>Mobile Phone Number</i>	The number to use to receive SMS alerts, including two-factor authentication codes.
<i>Password</i>	Allows you to change your Ops Manager password. Passwords must be at least 8 characters long and contain at least one letter, one digit, and one special character.
<i>Two-Factor Authentication</i>	Ops Manager may require two factor authentication for login. For details, see <i>Manage Your Two-Factor Authentication Options</i> . To delete or reset two-factor authentication, contact your Ops Manager system administrator.

#### Personalization

The *Personalization* page allows you to configure the console to suit your needs and preferences. The available fields depend on the your *role*. To access the page, select the *Settings* tab.

<i>My Time Zone</i>	Select your time zone.
<i>My Date Format</i>	Select your preferred date format.

## Public API Access

The *Public API Access* lets you *generate keys* for access to the *Public API* and lets you *configure access to whitelisted operations*.

## My Groups

The *My Groups* page displays the Ops Manager groups to which you belong. From here you can *add a group*.

## 9.2 Edit Your User Account

### On this page

- Overview
- Change Password
- Change Email Address
- Change Mobile Phone Number
- Change Display Settings and Newsletter Preference
- Additional Information

### Overview

You can update your password, email address, and mobile phone number. You can also modify your Ops Manager display settings and newsletter preference. You **cannot** change your username.

---

**Note:** To change your two-factor authentication settings, see *Manage Your Two-Factor Authentication Options*.

---

### Change Password

**Step 1:** Click the *Settings* tab, then click *Account*.

**Step 2:** Click the pencil icon for the *Password* field.

**Step 3:** Change your password and click *Save*.

### Change Email Address

**Step 1:** Click the *Settings* tab, then click *Account*.

**Step 2:** Click the pencil icon for the *Email* field.

**Step 3:** Enter your email address and click *Save*.

### Change Mobile Phone Number

Ops Manager uses your mobile phone number to send receive SMS alerts, including two-factor authentication codes.

**Step 1:** Click the **Settings** tab, then click **Account**.

**Step 2:** Click the pencil icon for the **Mobile Phone Number** field.

**Step 3:** Enter your phone number and click **Save**.

### Change Display Settings and Newsletter Preference

**Step 1:** Click the **Settings** tab, then click **Personalization**.

**Step 2:** Edit personal settings.

Edit the following, as desired. The available fields depend on your *role*.

<i>My Time Zone</i>	Select your time zone.
<i>My Date Format</i>	Select your preferred date format.

### Additional Information

To generate a key for access to the *Public API*, see [Enable the Public API](#).

## 9.3 Manage Your Two-Factor Authentication Options

### On this page

- [Overview](#)
- [Configure Two-Factor Authentication with Text or Voice](#)
- [Configure Two-Factor Authentication with Google Authenticator](#)
- [Generate New Recovery Codes](#)

### 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 *Settings* tab’s *Account* page in Ops Manager.

---

**Note:** To enable or disable two-factor authentication for the entire Ops Manager environment, see [Manage Two-Factor Authentication for 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 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.

## Other Two Factor Authentication Implementations

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 its 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](#).

## Configure Two-Factor Authentication with Text or Voice

**Step 1:** In Ops Manager, select the *Settings* 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 *Settings* 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 *Settings* 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.

## **10 Administer Ops Manager**

***Administration Overview*** Administer the Ops Manager system. This access is available only to administrators with global access.

***Administer Backups*** Administer Backup Daemons and snapshot storage.

***Add a Message to the Interface*** Add a message to selected pages in the Ops Manager interface.

***Start and Stop Ops Manager Application*** Manage Ops Manager.

***Back Up Ops Manager*** Options for backing up Ops Manager.

### **10.1 Administration Overview**

## On this page

- [Introduction](#)
- [General Tab](#)
- [Backup Tab](#)
- [Alerts Tab](#)
- [Control Panel Tab](#)

## Introduction

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 system administration interface. For configuration settings, see [Ops Manager Configuration](#).

## General Tab

The *General* tab gives access to system topology, users, messages, and other information.

### Overview Page

This page displays the Ops Manager network topology and provides reports on system use and activity.

### Ops Manager Config

The *General* tab's *Ops Manager Config* page lets you update your Ops Manager settings. For information on each field, see [Ops Manager Configuration](#).

### Users Page

This 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. The *Last Access* column displays the date and time of the last access event.

To edit a user record:

**Step 1: Click the *pencil icon* at the far right of the user record.**

Step 2:	Change any desired parameters in the <i>Edit User</i> interface.
Section	Possible Values
Two Factor Authentication	Change how the user can enable two factor authentication. You can either add or change their <i>Mobile Number</i> . You can also indicate if the user's <i>Google Authenticator</i> has been configured.
Profile Info	Change the user's email address.
Groups and Roles	Add or remove the user from one of the <i>roles</i> for each group in the Ops Manager Application.
Actions	Disable a user's ability to use Ops Manager. <i>Lock Account</i> prevents the user from logging into Ops Manager. <i>Clear Two Factor Auth</i> disables two factor authentication for the user.
Global Roles	Add or remove the user from one of the <i>roles</i> that apply across all groups in the Ops Manager Application.

**Step 3: To save these changes, click *Save*.** To delete a user record:

**Step 1: Click the *trash can icon* at the far right of the user record.**

**Step 2: Click *Delete*.**

## Groups Page

This page lists all *groups* created with their date created and the last date and time an agent for the group pinged Ops Manager. To view a group's details, click the group name. To delete a group, click the group's trash icon.

## Logs Page

This page 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.

## Version Manifest Page

This page lists all released MongoDB versions. Ops Manager requires this list to run in local mode. See *Version Manifest* in the *Configure Local Mode if Servers Have No Internet Access* tutorial for more information.

## Messages Page

This page displays messages you have added to the Ops Manager interface. You can add a message to any page of the Ops Manager interface to notify users of information and events, such as impending maintenance windows. To add messages, see *Add a Message to the Interface*

## Audits Page

This 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 *Admin* interface's *Backup* tab provides information on Backup resources, including *jobs*, *daemons*, and *block-stores*. For an overview of Backup and Backup resources, see *Backup Flows*.

## Jobs Page

This page lets you manage Backup jobs and Backup resources for each group. The top part of the page displays *Active Jobs* and the bottom part displays *Stopped Jobs*. The following fields on the page have a yellow background if 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.

**Manage Backup Jobs** From the *Jobs* page, you can do the following:

Task	Procedure
Assign a group to particular set of <i>Backup Daemons</i> or <i>blockstores</i> .	Click the group, select the daemons or <i>blockstores</i> , and select <i>Save Changes</i> .
View a job's log.	Click the name of the job and then click the <i>Logs</i> link. Contact MongoDB Support if you need help interpreting the error message.
View information about a job.	Click the job. From the job's page you can access logs, conf calls, and other information, and you can download diagnostics.
Move a job to a new Backup Daemon.	Click the job. On the job's page, click the <i>Move head</i> link, select the new <i>head</i> , and click the <i>Move Head</i> button.
Filter the page to display the jobs assigned to a particular daemon or blockstore.	Click the name of the daemon or blockstore.

**Manage Backup Resources** From the *Jobs* page, you can assign backup resources to a particular *group*.

You can make these changes for existing backups, but only new *backup jobs* will follow the new rules. Making these changes does not affect existing deployments. For additional procedural information, see *Move Jobs from a Lost Backup Daemon to another Backup Daemon*.

Task	Procedure
Assign a group to particular <i>daemons</i> , <i>blockstores</i> or <i>oplog stores</i> . Assign a group to a labelled set of daemons, blockstores, and oplog stores.	Click the group to open the group's assignment page; make the assignments; select <i>Save Changes</i> . First assign the label on the Admin page for each resource. See <i>Daemons Page</i> , <i>Snapshot Storage Page</i> , and <i>Oplog Stores Page</i> . Then, on the <i>Jobs</i> page: <ol style="list-style-type: none"><li>1. Click the group name to open the group's assignment page.</li><li>2. In the <i>Assignment Labels</i> list box, click <i>Select Labels</i>.</li><li>3. Select the label. Each selected label must exist on at least one daemon, one blockstore, and one oplog store. If you select multiple labels, the resource must meet all selected labels. Keep in mind that the resource must also meet any other selected criteria on this page. For example, for a daemon to match, it must also meet any selections you have made in the <i>Backup Daemon</i> field.</li><li>4. Click <i>Save Changes</i>.</li></ol>

## Job Timeline Page

This 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

This page 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

This page displays the last 100 requested restores and their progress. To show all restores ever requested, click *Show All*.

## Resource Usage Page

This 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

This page lists active and recent *groom* jobs. 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. A scheduling process determines when grooms are necessary.

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*.

You can also *Configure the Size of the Blocks in the Blockstore*.

## Groom Priority

This 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.

See also: [Configure the Size of the Blocks in the Blockstore](#).

## Daemons Page

This 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
<i>Show Detailed JSON</i>	Displays the Backup Daemon JSON. When the JSON displays, the <i>View raw runtime data</i> link appears under the code, allowing you to view the raw data.
<i>Move all heads</i>	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 <i>Move all heads</i> button.
<i>Delete Daemon Assignment</i>	Deletes a Backup Daemon.
<i>Backup Jobs</i>	Allows the daemon to take more jobs. If a disk fills on daemon server, you can deselect this so the server isn't overloaded.
<i>Restore Jobs</i>	Allows the daemon to take more backup jobs. Deselect this, for example, when performing maintenance on a daemon's server.
<i>Resource Usage</i>	Allows the daemon to take more restore jobs.
<i>Garbage Collection</i>	Collects information on <i>blockstore</i> use (i.e., on how many blocks are still referenced by snapshots). Ops Manager uses this information to generate the <a href="#">Resource Usage Page</a> page and to prioritize garbage collection.
<i>Head Disk Type</i>	Enables garbage collection.
<i>Assignment Labels</i>	Indicates whether the disk type is SSD or HDD. If you change a daemon to SSD, then only jobs with an <i>oplog</i> greater than 1GB/hour will go to this daemon, except in the case that no HDD daemon is available. Jobs with an oplog less than 1GB/hour can go to this daemon only if no HDD daemon is available. <b>Warning:</b> If you run Ops Manager 2.0.3 or earlier, jobs with an oplog less than 1GB/hour <b>cannot</b> be bound to an to SSD daemon.
	Creates one or more labels that can be used to assign the daemon to a specific group.

## Snapshot Storage Page

Ops Manager provides two ways to store MongoDB snapshots, on a File System and in a MongoDB database (*block-stores*).

This page lists your file systems and blockstores and provides the ability to add, edit or delete blockstores or file system stores.

**File System Storage Snapshots are stored on a regular File System, in a** configured directory. There will be a sub-directory per Snapshot and the files in that Snapshot will be individually compressed.

**Database Storage (Blockstore) Snapshots are stored in a MongoDB database in a** compressed, de-duplicated format. Depending on data usage patterns the de-duplication can provide significant storage savings without the need for any specialized hardware or other software.

**Additional Information** For additional information on managing snapshots, see:

- [Manage Blockstore Snapshot Storage](#)
- [Manage File System Snapshot Storage](#)
- [Assign Snapshot Stores to Specific Data Centers](#)
- [Configure the Size of the Blocks in the Blockstore](#)
- [Groom Priority](#), to manually direct Ops Manager to move blocks between blockstores

## 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.

**Warning:** Do not modify the oplog store's connection string to point to a different replica set. Existing data will not be copied and a resync will be required.

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.
- Assign labels that can be used to assign the oplog store to a specific group. Enter labels in *Assignment Labels*. Separate multiple labels with commas.

After saving changes to the oplog store's values, you must restart all the Ops Manager instances, including those running activated Backup Daemons, for the changes to take effect.

## Alerts Tab

The Admin interface's *Alerts* tab sets and displays global and system alerts. *Global alerts* apply a group-level alert condition to multiple groups. *System alerts* monitor the health of Ops Manager.

## Control Panel Tab

### Message History Page

This page displays alert messages sent and the recipients. To filter the list, click *Filter by Recipient*.

### Send Test Message

Use this page to send messages to users to test validity of their addresses.

## 10.2 Administer Backups

**Manage Blockstore Snapshot Storage** Manage the snapshots that back up your deployments to a database.

**Manage File System Snapshot Storage** Manage the snapshots that back up your deployments to a file system.

**Configure Block Size** Configure the size of the blocks in the Backup Blockstore database for a given replica set.

**Manage Backup Daemon Jobs** Manage job assignments among the backup daemon

### Manage Blockstore Snapshot Storage

#### On this page

- Prerequisites
- Procedures

Ops Manager can backup databases to either another database (blockstore), to a local file system or a combination of both. This tutorial covers using blockstores to backup your MongoDB databases. Blockstores can exist on any database the Ops Manager server can access.

---

**Note:** Use multiple snapshot stores when you must add capacity or when you must meet data localization or other regulatory requirements. [Assign Snapshot Stores to Specific Data Centers](#) describes how to assign snapshot stores to different data centers.

---

#### Prerequisites

Before creating any blockstore snapshot stores:

- Ensure a storage volume with sufficient capacity is attached to the Ops Manager or MongoDB server to store the blockstores and the oplog MongoDB database.
- [Deploy the dedicated MongoDB instance\(s\)](#) to host the blockstores and oplog stores for this snapshot store.

#### Procedures

##### Add a Blockstore

**Step 1: Navigate to the *Snapshot Storage* page.**

1. Click the *Admin* link.
2. Click the *Backup* tab.
3. Click the *Snapshot Storage* page.

**Step 2: Click *Create New Blockstore*.**

Field	Contents
<i>Name</i> <i>Datastore Type</i> drop-down menu	A name for the blockstore <ul style="list-style-type: none"><li>• If you select <i>Standalone</i>, the IP address and port number of the MongoDB instance.</li><li>• If you select <i>Replica Set</i>, the IP addresses and port numbers of the members in the replicated list in the <i>&lt;hostname&gt;:port</i> format in the <i>MongoDB Host List</i>.</li><li>• If you select <i>Shared</i>, enter a comma-separated list of <i>&lt;hostname&gt;:port</i> for the <i>MongoDB Host List</i>.</li></ul>
<i>Username</i>	The username that can access the database. If the database uses authentication.
<i>Password</i>	The password of the username that can access the database, if the database uses authentication.
<i>Connection Options</i>	Any configuration file options for the connection. This field supports unescaped JSON syntax, see <a href="#">Connection String Usage</a> in the MongoDB manual.
<i>Use SSL</i> checkbox <i>New Assignment Enabled</i> checkbox	If the database uses SSL, this checkbox is selected. This enables the blockstore once selected. If you leave this unchecked, the blockstore will not accept newly started backups can be accepted.

**Step 3: Provide the Blockstore details.**

**Step 4: Click *Create*.**

**Edit an Existing Blockstore** Once created, blockstores are listed directly on the Snapshot Storage page in a table. Each row contains the settings for each Blockstore.

**Step 1: Navigate to the *Snapshot Storage* page.**

1. Click the *Admin* link.
2. Click the *Backup* tab.
3. Click the *Snapshot Storage* page.

**Step 2: Go to the row for the Blockstore you want to edit.**

Step 3: In the MongoDB Connection column, edit the necessary fields:	
Field	Contents
<host-name>:<port>	The hostname and port number of the MongoDB database.
<i>MongoDB Auth Username</i>	The name of the user authorized to access the database if authentication was selected.
<i>MongoDB Auth Password</i>	The password of the user authorized to access the database if authentication was selected.
<i>Encrypted Credentials checkbox</i>	Checked if the username and password had used the <code>credentialstool</code> for encryption.
<i>Use SSL checkbox</i>	Checked if the database uses SSL.
<i>Connection Options</i>	Include any <a href="#">configuration file options</a> for the MongoDB instance. This field supports un-escaped values only. For syntax, see <a href="#">Connection String URI Format</a> in the MongoDB manual.
<i>Assignment Labels</i>	A comma-separated list of labels to assign the blockstores to specific groups.
<i>Blockstore Max Capacity (GB)</i>	This field is not used.
<i>Load Factor</i>	<p>A proportional value of how backup jobs are assigned to the given snapshot store compared to other snapshot stores.</p> <p>By default snapshot stores assign one shard per snapshot store.</p> <p>If a snapshot store has more than one shard assigned, it would result in one snapshot store being backed up often than the another. This ratio of backup jobs to shards can be changed using the load factor.</p> <p>An example of how the Load Factor affects backups:</p> <p>You are backing up a 5 shard cluster. Your deployment has filestore (A) with one shard and blockstore (B) with four shards. These blockstores should not get equally distribution of backup jobs, as B is four times larger than A. B should get a Load Factor of 4 and A should get a Load Factor or 1.</p>
<i>Write Concern</i>	Select the proper <a href="#">Write Concern</a> .

**Step 4: Check the checkbox in the *Assignment Enabled* column.**

**Step 5: Click *Save*.**

**Step 6: Restart all the Ops Manager instances, including those running activated Backup Daemons.**

**Warning:** After you save changes to the connection string, you must restart all the Ops Manager instances, including those running activated Backup Daemons, for the changes to take effect.

## Delete a Blockstore

**Step 1: Navigate to the *Snapshot Storage* page.**

1. Click the *Admin* link.
2. Click the *Backup* tab.
3. Click the *Snapshot Storage* page.

**Step 2: Click the *Delete <blockstore>* link under the name of the blockstore you want to delete.**

## Manage File System Snapshot Storage

### On this page

- [Prerequisites](#)
- [Procedures](#)

Ops Manager can backup databases to either another database (blockstore), to a local file system or a combination of both. This tutorial covers using file system stores to backup your MongoDB databases.

If you use file system snapshot storage and use multiple Ops Manager instances (including those activated as Backup Daemons), all Ops Manager instances must have the same view of the file system snapshot storage. You can achieve this through a Network File System (NFS) application or something similar. If the Ops Manager instances do not share the same view of file system snapshot storage, Backup restores will not be possible and Ops Manager will not be able to remove expired snapshots.

---

**Note:** Use multiple snapshot stores when you must add capacity or when you must meet data localization or other regulatory requirements. [Assign Snapshot Stores to Specific Data Centers](#) describes how to assign snapshot stores to different data centers.

---

### Prerequisites

Before creating any file system snapshot stores:

- Ensure a storage volume with sufficient capacity is attached to the Ops Manager server to store the snapshot files and the oplog MongoDB database.
- [Deploy the dedicated MongoDB instance\(s\)](#) to host the oplog stores for this snapshot store.
- If you use multiple Ops Manager instances (including those activated as Backup Daemons), set up a Network File System (NFS) application or something similar to ensure the all instances share the same view of file system snapshot storage.

---

**Important:** If the Ops Manager instances do not share the same view of file system snapshot storage, Backup restores will not be possible and Ops Manager will not be able to remove expired snapshots.

---

### Procedures

#### Add a File System Store

**Step 1: Navigate to the *Snapshot Storage* page.**

1. Click the *Admin* link.
2. Click the *Backup* tab.
3. Click the *Snapshot Storage* page.

**Step 2: Click *Create New File System Store*.**

**Step 3: Complete the File System Store details.**

Field	Contents
<i>File System Store</i>	A name for the file system store.
<i>Name</i>	
<i>Path</i>	The file system path in which the snapshot will be stored.
<i>New Assignment</i>	This enables the filestore once it has been created. If you leave this checked, the filestore is created but no newly started backups can be assigned to it.
<i>Enabled checkbox</i>	

**Step 6: Click *Create*.**

**Edit an Existing File System Store** Once created, file system stores are listed directly on the Snapshot Storage page in a table. Each row contains the settings for each File System Store.

**Step 1: Navigate to the *Snapshot Storage* page.**

1. Click the *Admin* link.
2. Click the *Backup* tab.
3. Click the *Snapshot Storage* page.

**Step 2: Go to the row for the filestore you want to edit.****Step 3: In the *Store Path* column, edit the necessary fields:**

Field	Contents
<i>Store Path</i>	The location where file system-based backups are stored.
<i>Assignment</i>	A comma-separated list of labels to assign the filestores to.
<i>Labels</i>	
<i>Load Factor</i>	A proportional value of how backup jobs are assigned to snapshot stores. By default snapshot stores assign one shard per snapshot. If a snapshot store has more than one shard assigned, it will be backed up often than the other. This ratio of backup job factor.
<i>Write Concern</i>	An example of how the Load Factor affects backups: You are backing up a 5 shard cluster. Your deployment has two blockstores (B) with four shards. These blockstores should have twice as many backup jobs as A, as B is four times larger than A. B should get a Load Factor of 1. Select the proper Write Concern.

**Step 4: Check the checkbox in the *Assignment Enabled* column.****Step 5: Click *Save*.****Step 6: Restart all the Ops Manager instances, including those running activated Backup Daemons.**

**Warning:** After you save changes to the connection string, you must restart all the Ops Manager instances, including those running activated Backup Daemons, for the changes to take effect.

## Delete a File System Store

### Step 1: Navigate to the *Snapshot Storage* page.

1. Click the *Admin* link.
2. Click the *Backup* tab.
3. Click the *Snapshot Storage* page.

### Step 2: Click the *Delete <filestore>* link under the name of the filestore you want to delete.

## Configure the Size of the Blocks in the Blockstore

### On this page

- [Overview](#)
- [Considerations](#)
- [Procedure](#)

### Overview

If you use the *Backup Blockstore Database* for snapshot storage, then when you back up a replica set, 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.

By default, each block in the Backup Blockstore is 64KB, but you can configure the blocksize to suit your use case.

### Considerations

In general, increasing blocksize results in faster snapshots and restores, but requires more disk space. These competing factors should be considered when determining if you wish to tune the blocksize.

For users with update and delete-intensive workloads, and thus with a poor de-duplication rate, increasing the blocksize provides performance improvements **without** requiring extra disk space. With update and delete-intensive workloads, no matter how small you make the blocksize the entire block file will need to be rewritten. Since the entire file is always being rewritten there isn't any difference in storage space if you change the blocksize.

Users with insert-only workloads will also see the performance benefits of increasing blocksize without requiring additional disk space. With an insert-only workflow, the existing blocks never change: increasing blocksize then enables easier block management and the best possible performance on snapshot and restore.

### Procedure

#### Step 1: Open the *Admin* menu, and then select *Backup* tab.

#### Step 2: Select the *Jobs* tab.

#### Step 3: Click on the name of the replica set that you wish to edit.

**Step 4: Select the desired minimum blocksize from the drop-down menu..** Choose from 64KB through 15MB. By default, Ops Manager uses 64kb blocks.

---

**Note:** The updated minimum blocksize only applies to new and updated files in future snapshots. Existing blocks are not resized.

---

### Step 5: Click *Update*

## Move Jobs from a Lost Backup Daemon to another Backup Daemon

### On this page

- [Overview](#)
- [Procedure](#)

### 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 Data Storage*. 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 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 *does* save incoming oplog data. Once the jobs are on the new Backup Daemon's server, Ops Manager 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 restore* 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.

## 10.3 Add a Message to the Interface

### On this page

- Overview
- Procedures

### Overview

You can display a message on any page of the Ops Manager interface to notify users of information or events, such as impending maintenance windows.

### Procedures

#### Add a Message

1. Click the *Admin* link in the upper right corner of the Ops Manager interface.
2. Click the *General* tab and then click *Messages*.
3. Click the *Add Message* button.
4. Enter the message and the page or page prefix. 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 `http://docs.opsmanager.mongodb.com//` character.

For example, entering the page prefix `http://docs.opsmanager.mongodb.com//settings/profile` will display a message on the default *Settings* tab and *Account* page but not on any other tab or page in the application.

5. Click the *Active* checkbox to make the message live. Optionally, you can leave the box unchecked to disable the message.
6. Click *Add*.

Once added, active messages take 60 seconds before they display.

#### Disable a Message

1. Click the *Admin* link in the upper right corner of the Ops Manager interface.
2. Click the *General* tab and then click *Messages*.
3. Click the orange square button to the right of the alert.

To re-enable a disabled message, click the grey button.

## Delete a Message

1. Click the *Admin* link in the upper right corner of the Ops Manager interface.
2. Click the *General* tab and then click *Messages*.
3. Click the garbage can icon to the right of the alert.

## 10.4 Start and Stop Ops Manager Application

### On this page

- Start the Ops Manager Server
- Stop the Ops Manager Application
- Startup Log File Output
- Optional: Run as Different User
- Optional: Ops Manager Application Server Port Number

### 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 Ops Manager deployment, you can start the Ops Manager Application 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.

To open Ops Manager, enter the URL specified in the URL to Access Ops Manager setting in the Ops Manager UI. If you are opening Ops Manager for the first time, enter the following URL, where `<host>` is the fully qualified domain name of the Ops Manager server. Ops Manager prompts you to register a new user when you login for the first time.

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

### Stop the Ops Manager Application

Enter the following command:

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

### Startup Log File Output

The Ops Manager Application 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 Ops Manager Application 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: SelectChannelConne
[null] LoginService=HashLoginService identityService=org.eclipse.jetty.security.DefaultIdentityService
[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 trap
[main] INFO ServerMain:266 - Started 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 the Ops Manager Application:

```
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.example.com:<newport>
```

2. Edit <install\_dir>/conf/mms.conf

```
BASE_PORT=<newport>
```

3. Restart the Ops Manager Application:

```
sudo <install_dir>/bin/mongodb-mms restart
```

## 10.5 Back Up Ops Manager

### On this page

- [Back Up with the Public API](#)
- [Shut Down and Back Up](#)

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 package 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 allows you 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 a snapshot as a typical restore.

See [the API documentation](#) for how to restore jobs using this method.

This method backs up the snapshots only: you **cannot** use the backups to restore Ops Manager in the event that a Blockstore database is lost.

## Shut Down and Back Up

To perform a full backup of all snapshots contained in Ops Manager, including the point-in-time restores:

1. Shut down the Ops Manager application service
2. Shut down the Ops Manager application databases
3. Shut down the Ops Manager backup databases.
4. If you backed up using blockstores, back up the file systems that store the application and backup databases (and/or files) while they are offline.
5. If you backed up using file system stores, backup the snapshot files on the file system.

The *Shut Down and Back Up* approach has the advantage of providing all of Ops Manager's backup snapshots *and* 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 are monitoring and automation capabilities. As such, if a failure occurs in one of your MongoDB instances, you may lose the data that Ops Manager did not back up prior to shutdown nor are you alerted of that fact because monitoring is unavailable.

# 11 API

[\*\*Public API Principles\*\*](#) Overview of the Public API.

[\*\*Public API Resources\*\*](#) The resources exposed by the Public API.

[\*\*Enable the Public API\*\*](#) Enable the Public API for group.

[\*\*Configure Public API Access\*\*](#) Generate Public API keys and configure access to whitelisted operations.

[\*\*Public API Tutorials\*\*](#) Enable the API, and create and modify a deployment.

## 11.1 Public API Principles

### On this page

- Overview
- HTTP Methods
- JSON
- Linking
- Lists
- Envelopes
- Pretty Printing
- Response Codes
- Errors
- Authentication
- Automation
- Additional Information

### 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.
- HEAD - Retrieve the response header without the JSON representation of the 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 milliseconds, 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://cloud.mongodb.com/api/public/v1.0/groups/xxx/hosts/yyy"
    },
    {
      "rel": "http://mms.mongodb.com/group",
      "href": "https://cloud.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 response 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": "self",  
      "href": "https://www.mongodb.com/api/public/v1.0/groups/xxx/hosts?pageNum=2&itemsPerPage=10"  
    },  
    {  
      "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:

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0?pretty=true"
```

---

**Note:** All the examples in this document show pretty-printed JSON 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 successful 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, the provided credentials are incorrect, or the user associated with the given API key is not allowed to access the requested resource.
403	Forbidden	Access to the specified resource is not permitted.
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 whitelists that allow access to the resource only from the IP addresses listed. Each user configures their own whitelist of IP addresses that allow access to the resource.
- 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 401 error.

## Automation

The [Automation Configuration Resource](#) and [Automation Status](#) resources provide 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.

## 11.2 Public API Resources

The Ops Manager *Public API* exposes the following resources:

*Root*

*Hosts*

*Agents*

*Metrics*

*Clusters*

*Groups*

*Users*

*Alerts*

*Alert Configurations*

*Maintenance Windows*

*Backup Configurations*

*Snapshot Schedule*

*Snapshots*

*Checkpoints*

*Restore Jobs*

*Whitelist*

*Automation Configuration*

*Automation Status*

### Root

#### On this page

- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Example](#)

This is the starting point 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
<a href="http://mms.mongodb.com/groups">http://mms.mongodb.com/groups</a>	Groups accessible to the current API user.
<a href="http://mms.mongodb.com/user">http://mms.mongodb.com/user</a>	The current API user.

### Example

Retrieve the root resource:

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0"
```

**HTTP/1.1 200 OK**

```
{  
  "throttling" : false,  
  "links" : [ ... ]  
}
```

## Hosts

### On this page

- Overview
- Endpoints
- Sample Entity
- Entity Fields
- Links
- Examples

## Overview

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.

## Endpoints

**Get All Hosts in a Group** 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
```

### Get a Host by ID

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID
```

**Get a Host By Name and Port** Get a single host by its hostname and port combination. You can specify either the primary hostname or an alias.

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/byName/HOSTNAME:PORT
```

**Create a Host** Create a new host in the group. When you create a new host, Ops Manager only knows the information that you provided. Thus, the document returned in the response document will include blank values while Ops Manager discovers the missing values.

```
POST /api/public/v1.0/groups/GROUP-ID/hosts
```

You may specify the following fields when creating a host:

- `hostname` - Required.
- `port` - Required.
- `username` - Required if `authMechanismName` is `MONGODB_CR`. Otherwise illegal.
- `password` - Required if `authMechanismName` is `MONGODB_CR`. Otherwise illegal.
- `sslEnabled` - Must be `true` if the `authMechanismName` is `MONGODB_X509`. 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.

## Update a Host

PATCH /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID

You can specify the following fields:

- username
- password
- sslEnabled
- logsEnabled
- alertsEnabled
- profilerEnabled
- muninPort
- authMechanismName

If authMechanismName is NONE then any existing value for username and password will be cleared out. If authMechanismName is MONGODB\_CR, then you must provide both the username and password.

## Delete a Host

DELETE /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID

## 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",
  "hasStartupWarnings": true,
  "sslEnabled": false,
  "logsEnabled": false,
  "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" ],
"displayName": "Newton",
"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	number	Port that MongoDB process (mongod or mongos) listens on.
typeName	string	Type for this host. Possible values are: <ul style="list-style-type: none"> <li>• STANDALONE</li> <li>• REPLICA_PRIMARY</li> <li>• REPLICA_SECONDARY</li> <li>• REPLICA_ARBITER</li> <li>• RECOVERING</li> <li>• MASTER</li> <li>• SLAVE</li> <li>• SHARD_MONGOS</li> <li>• SHARD_CONFIG</li> <li>• SHARD_STANDALONE</li> <li>• SHARD_PRIMARY</li> <li>• SHARD_SECONDARY</li> <li>• NO_DATA</li> </ul> The host's type for new hosts added to Ops Manager will be NO_DATA until the Monitoring Agent receives its first ping.
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.
hasStartupWarnings	boolean	Are there startup warnings for this host?
sslEnabled	boolean	Is SSL enabled for this host? Must be true if the authMechanismName is MONGODB_X509.
logsEnabled	boolean	Is Ops Manager collecting logs for this host?
uptimeMsec	number	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 <a href="#">Replica Set Member States</a> for possible values.
created	date	Date this host was created or first discovered by Ops Manager.
hostEnabled	boolean	Is this host currently enabled? Hosts

## Links

Relation	Description
<code>self</code>	Me
<code>http://mms.mongodb.com</code>	The cluster this host belongs to. Only present if the host is part of a replica set or master/slave.
<code>http://mms.mongodb.com</code>	The parent cluster. Only present if the host is part of a sharded cluster.
<code>http://mms.mongodb.com</code>	The group that this host belongs to.
<code>http://mms.mongodb.com</code>	All available metrics for the host.
<code>http://mms.mongodb.com</code>	All snapshots for the <i>config server</i> . Only present if the host is a config server and if the config server is running as a standalone, not as a replica set.
<code>http://mms.mongodb.com</code>	All restore jobs for the <i>config server</i> . Only present if the host is a config server and if the config server is running as a standalone, not as a replica set.

## Examples

### Create a New Host

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST "https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e/hosts/680ab31647f343333333333333333333"
{
  "hostname": "localhost",
  "port": 27017
}

HTTP/1.1 201 Created
Location: https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e/hosts/680ab31647f343333333333333333333

{
  "id" : "4059580c20c4581872ef24d0b8f5dca0",
  "groupId" : "5196d3628d022db4cbc26d9e",
  "hostname" : "localhost",
  "port" : 27017,
  "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://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e/hosts/680ab31647f343333333333333333333"
{
  "sslEnabled": true,
  "username": "mongo",
  "password": "M0ng0DB!:")
}
```

```
HTTP/1.1 200 OK
```

```
{
  "id" : "680ab316473d6b28f966364b947134fc",
  "groupId" : "533c5895b91030606f21033a",
  "hostname" : "localhost",
  "port" : 26000,
  "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://cloud.mongodb.com/api/public/v1.0/groups/533c5895b91030606f21033a"
```

```
HTTP/1.1 200 OK
```

```
{
  "id" : "56e9378f601dc49360a40949c8a6df6c",
  "groupId" : "533c5895b91030606f21033a",
  "hostname" : "mymongo.test.com",
  "port" : 26000,
  "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" ],
  "displayName": "Leibniz",
  "links" : [ ... ]
}
```

## Get All Hosts

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/533c5895b91030606f21033a"
```

```
HTTP/1.1 200 OK
```

```
{
  "totalCount" : 2,
  "results" : [
    {
      "id" : "56e9378f601dc49360a40949c8a6df6c",
      "groupId" : "533c5895b91030606f21033a",
      "hostname" : "mymongo.test.com",
      "port" : 26000,
      "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" ],
      "displayName": "Leibniz",
      "links" : [ ... ]
    },
    {
      ...
    }
  ]
}
```

## Delete a Host

```
curl -u "username:apiKey" --digest -i -X DELETE "https://cloud.mongodb.com/api/public/v1.0/groups/533c5895b91030606f21033a/hosts/56e9378f601dc49360a40949c8a6df6c"
HTTP/1.1 200 OK
```

## Agents

### On this page

- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

## Endpoints

### Get All Agents

Return links to all agents

```
GET /api/public/v1.0/groups/GROUP-ID/agents
```

**Get All Agents of a Type** Get all agents of the specified TYPE.

```
GET /api/public/v1.0/groups/GROUP-ID/agents/TYPE
```

You can specify MONITORING, BACKUP, or AUTOMATION, as in:

```
GET /api/public/v1.0/groups/GROUP-ID/agents/MONITORING  
GET /api/public/v1.0/groups/GROUP-ID/agents/BACKUP  
GET /api/public/v1.0/groups/GROUP-ID/agents/AUTOMATION
```

## Sample Entity

### Monitoring Agent

```
{  
  "confCount": 59,  
  "hostname": "example",  
  "isManaged": true,  
  "lastConf": "2015-06-18T14:21:42Z",  
  "lastPing": "2015-06-18T14:21:42Z",  
  "pingCount": 6,  
  "stateName": "ACTIVE",  
  "typeName": "MONITORING"  
}
```

### Backup Agent

```
{  
  "confCount": 12,  
  "hostname": "example",  
  "isManaged": true,  
  "lastConf": "2015-06-18T14:26:29Z",  
  "stateName": "ACTIVE",  
  "typeName": "BACKUP"  
}
```

### Automation Agent

```
{  
  "confCount": 141,  
  "hostname": "example-1",  
  "lastConf": "2015-06-18T14:27:42Z",  
  "stateName": "ACTIVE",  
  "typeName": "AUTOMATION"  
}
```

## Entity Fields

Name	Type	Description
typeName	string	Specifies the type of agent, can be MONITORING, BACKUP, or AUTOMATION.
hostname	string	Primary hostname. A host typically may have aliases, so the primary is the best available name as decided by Ops Manager.
confCount	number	Number of configuration calls.
lastConf	timestamp	Date and time of last configuration call.
stateName	string	The current state of the agent. stateName can return the following values: <ul style="list-style-type: none"> <li>• ACTIVE: the agent is active and operating</li> <li>• STANDBY: the agent is on standby</li> <li>• NO_PROCESSES: the agent is not managing, monitoring, or backing up any processes.</li> </ul>
pingCount	number	<b>Only applicable to Monitoring agents.</b> The number of pings that the Monitoring agent has sent to the hostname URL.
isManaged	Boolean	<b>Only applicable to Monitoring and Backup agents.</b> Specifies whether or not Ops Manager manages the agent.
lastPing	timestamp	<b>Only applicable to Monitoring agents.</b> Time of most recent ping.
tag	string	<b>Only applicable to Backup agents.</b> The agent's tag, if there is one.

## Links

Relation	Description
self	Me
<a href="http://mms.mongodb.com/group">http://mms.mongodb.com/group</a>	The group the agents belong to
<a href="http://mms.mongodb.com/monitoringAgents">http://mms.mongodb.com/monitoringAgents</a>	Links to all monitoring agents.
<a href="http://mms.mongodb.com/backupAgents">http://mms.mongodb.com/backupAgents</a>	Links to all backup agents.
<a href="http://mms.mongodb.com/automationAgents">http://mms.mongodb.com/automationAgents</a>	Links to all automation agents.

## Examples

### Get All Monitoring Agents in a Group

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/xxxxxxxxxxxxxx"
```

HTTP/1.1 200 OK

```
{
  "links": [
    {
      "href": "https://cloud.mongodb.com/api/public/v1.0/groups/xxxxxxxxxxxxxxxxxxxxxx/agents/MONITORING",
      "rel": "self"
    }
  ],
  "results": [
    {
      "confCount": 59,
      "hostname": "example",
      "isManaged": true,
      "lastConf": "2015-06-18T14:21:42Z",
      "lastPing": "2015-06-18T14:21:42Z",
      "pingCount": 6,
      "stateName": "ACTIVE",
      "typeName": "MONITORING"
    }
  ],
  "totalCount": 1
}
```

## Metrics

### On this page

- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

## Endpoints

**Get All Available Metrics** Get a list of all available metrics for the host. Each entity in the list will be a partial metric entity. No data is 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
```

**Get a Host Metric** 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.

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/metrics/METRIC-NAME
```

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`, and `DAY`. The default is `MINUTE`.

- **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 a Database Metric** Get the data points for the specified host, database metric, and database name. The only available database-level metric is DB\_LOCK\_PERCENTAGE.

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/metrics/DB-METRIC-NAME/DB-NAME
```

You may also provide additional query parameters:

- **granularity** - The size of the epoch. Acceptable values are: MINUTE, HOUR, and DAY. The default is MINUTE.
- **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 a Hardware Metric** 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\_.

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/metrics/HW-METRIC-NAME/DEVICE-NAME
```

You may also provide additional query parameters:

- **granularity** - The size of the epoch. Acceptable values are: MINUTE, HOUR, and DAY. The default is MINUTE.
- **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`.

## 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	string	The name of the metric. Possible values are: <ul style="list-style-type: none"><li>• ASSERT_MSG</li><li>• ASSERT_REGULAR</li><li>• ASSERT_USER</li><li>• ASSERT_WARNING</li><li>• BACKGROUND_FLUSH_AVG</li><li>• CACHE_ACTIVITY</li><li>• CACHE_USAGE</li><li>• COMPUTED_MEMORY</li><li>• CONNECTIONS</li><li>• CURSORS_TOTAL_CLIENT_CURSORS_SIZE</li><li>• CURSORS_TOTAL_OPEN</li><li>• CURSORS_TOTAL_TIMED_OUT</li><li>• DB_DATA_SIZE_TOTAL</li><li>• DB_LOCK_PERCENTAGE</li><li>• DB_STORAGE_TOTAL</li><li>• EFFECTIVE_LOCK_PERCENTAGE</li><li>• EXTRA_INFO_PAGEFAULTS</li><li>• GLOBAL_ACCESSES_NOT_IN_MEMORY</li><li>• GLOBAL_LOCK_CURRENT_QUEUE_READER</li><li>• GLOBAL_LOCK_CURRENT_QUEUE_TOTAL</li><li>• GLOBAL_LOCK_CURRENT_QUEUE_WRITE</li><li>• GLOBAL_PAGEFAULT_EXCEPTIONS_THRESHOLD</li><li>• INDEX_COUNTERS_BTREE_ACCESSES</li><li>• INDEX_COUNTERS_BTREE_HITS</li><li>• INDEX_COUNTERS_BTREE_MISSES</li><li>• INDEX_COUNTERS_BTREE_MISS_RATIO</li><li>• JOURNALING_COMMITS_IN_WRITE_LOCK</li><li>• JOURNALING_MB</li><li>• MEMORY_MAPPED</li><li>• MEMORY_RESIDENT</li><li>• MEMORY_VIRTUAL</li><li>• MUNIN_CPU_IOWAIT</li><li>• MUNIN_CPU_IRQ</li><li>• MUNIN_CPU_NICE</li><li>• MUNIN_CPU_SOFTIRQ</li><li>• MUNIN_CPU_STEAL</li><li>• MUNIN_CPU_SYSTEM</li><li>• MUNIN_CPU_USER</li></ul>

## Links

Relation	Description
<code>self</code>	Me
<code>http://mms.mongodb.com/group</code>	The group that the host belongs to.
<code>http://mms.mongodb.com/host</code>	The host to which the metric pertains.

## Examples

### Get All Available Metrics

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/51b9361d5ae9048f0aab01f4"
```

HTTP/1.1 200 OK

```
{
  "totalCount" : 53,
  "results" : [ {
    "hostId" : "04cf770dc43c9ff21747ecf71ff9ee78",
    "groupId" : "51b9361d5ae9048f0aab01f4",
    "metricName" : "ASSERT_REGULAR",
    "units" : "RAW",
    "links" : [ {
      "rel" : "self",
      "href" : "https://cloud.mongodb.com/api/public/v1.0/groups/51b9361d5ae9048f0aab01f4/hosts/04cf"
    } ]
  }, {
    "hostId" : "04cf770dc43c9ff21747ecf71ff9ee78",
    "groupId" : "51b9361d5ae9048f0aab01f4",
    "metricName" : "ASSERT_WARNING",
    "units" : "RAW"
    "links" : [ {
      "rel" : "self",
      "href" : "https://cloud.mongodb.com/api/public/v1.0/groups/51b9361d5ae9048f0aab01f4/hosts/04cf"
    } ]
  }, ... ],
  "links" : [ ... ]
}
```

### Get a Single Metric

The following example gets hour-level data for the past 12 hours.

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/51b9361d5ae9048f0aab01f4"
```

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" : 381.2
  }
]
```

```

    "value" : 407.23
}, {
  "timestamp" : "2014-08-29T16:00:00Z",
  "value" : 365.3124
}, ... ],
"links" : [ ... ]
}

```

## Clusters

### On this page

- [Overview](#)
- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

### Overview

MongoDB supports two types of clusters: *replica sets* and *sharded clusters*. A sharded cluster can contain replica sets within it: each shard can be a replica set and the config server can also be a replica set. These relationships are 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.

### Endpoints

**Get All Clusters** Get all clusters in a group.

```
GET /api/public/v1.0/groups/GROUP-ID/clusters
```

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.

**Get a Cluster** Get a single cluster by ID.

```
GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID
```

**Update a Cluster** Update a cluster by ID. The only property that you may modify is the `clusterName`, since Ops Manager discovers all other cluster properties. This operation is only available on clusters of type SHARDED and SHARDED\_REPLICA\_SET.

```
PATCH /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID
```

## 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	string	Specifies what kind of cluster this is. Possible values are: <ul style="list-style-type: none"> <li>• REPLICA_SET</li> <li>• SHARDED: a sharded cluster where each shard is a <i>standalone</i> instance. No shards are replica sets.</li> <li>• SHARDED_REPLICA_SET: a sharded cluster that contains at least one shard that is a replica set.</li> <li>• CONFIG_SERVER_REPLICA_SET: config servers deployed as a replica set.</li> <li>• MASTER_SLAVE</li> </ul>
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 or CONFIG_SERVER_REPLICA_SET.
lastHeartbeat	date	The approximate last time Ops Manager processed a ping from this cluster.

## Links

Relation	Description
<code>self</code>	Me
<code>http://mms.mongodb.com</code>	The parent clusters Only present if the type is SHARDED or REPLICA_SET within a sharded cluster.
<code>http://mms.mongodb.com</code>	The group that this cluster belongs to.
<code>http://mms.mongodb.com</code>	The members shards that belong to this cluster. Only present if the type is SHARDED_REPLICA_SET.
<code>http://mms.mongodb.com</code>	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://cloud.mongodb.com/api/public/v1.0/groups/533c5895b9103f3d4730040be257defe88"

HTTP/1.1 200 OK

{
  "id" : "533c5895b9103f3d4730040be257defe88",
  "typeName" : "SHARDED_REPLICA_SET",
  "clusterName" : "Animals",
  "lastHeartbeat" : "2014-04-03T15:26:58Z",
  "links" : [ ... ]
}
```

### Get all Clusters

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups"

HTTP/1.1 200 OK

{
  "totalCount" : 3,
  "results" : [ {
    "id" : "533c5895b9103f3d4730040be257defe88",
    "typeName" : "SHARDED_REPLICA_SET",
    "clusterName" : "Animals",
    "lastHeartbeat" : "2014-04-03T15:26:58Z",
    "links" : [ ... ]
  }, {
    "id" : "533c5895b9103f3d4730040be257defe85",
    "typeName" : "REPLICA_SET",
    "clusterName" : "Animals",
    "shardName" : "cats",
    "replicaSetName" : "cats",
    "lastHeartbeat" : "2014-04-03T15:24:54Z",
    "links" : [ ... ]
  }, {
    "id" : "533c5895b9103f3d4730040be257defe83",
    "typeName" : "REPLICA_SET",
    "clusterName" : "Animals",
    "shardName" : "dogs",
    "links" : [ ... ]
  } ]
}
```

```
        "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://cloud.mor.../clusters/Zoo"
{
    "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

- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

## Endpoints

**Get All Groups for a User** Get all groups for the current user.

```
GET /api/public/v1.0/groups
```

**Create a Group** Specify only the name field.

```
POST /api/public/v1.0/groups
```

The publicApiEnabled field is automatically set to true for groups created with the API. The response includes the agentApiKey for the group.

**Get a Group** Get a single group by its ID.

```
GET /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**.

```
DELETE /api/public/v1.0/groups/GROUP-ID
```

### Get All Users in a Group

```
GET /api/public/v1.0/groups/GROUP-ID/users
```

**Add Users to a Group** Add one or more existing users to a group.

```
POST /api/public/v1.0/groups/GROUP-ID/users
```

You must send an array of entities, even if you're only adding a single user. For each user that you wish to add, specify the user ID and the *roles* that the user should possess.

If you specify a user that is already a member of the group, their existing roles will be overwritten with the specified permissions.

### Remove a User from a Group

```
DELETE /api/public/v1.0/groups/GROUP-ID/users/USER-ID
```

## 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	number	Number of Monitoring Agents sending regular pings to Ops Manager.
replicaSetCount	number	Total number of replica sets for this group.
shardCount	number	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 <code>true</code> is by enabling the Public API for the group in the <b>Settings</b> 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
<a href="http://mms.mongodb.com/hosts">http://mms.mongodb.com/hosts</a>	All hosts in the group.
<a href="http://mms.mongodb.com/users">http://mms.mongodb.com/users</a>	All users in the group.
<a href="http://mms.mongodb.com/clusters">http://mms.mongodb.com/clusters</a>	All clusters in the group.
<a href="http://mms.mongodb.com/alerts">http://mms.mongodb.com/alerts</a>	All open alerts for the group.
<a href="http://mms.mongodb.com/alertConfigs">http://mms.mongodb.com/alertConfigs</a>	All alert configurations for the group.
<a href="http://mms.mongodb.com/backupConfigs">http://mms.mongodb.com/backupConfigs</a>	All backup configurations for the group.
<a href="http://mms.mongodb.com/agents">http://mms.mongodb.com/agents</a>	All agents for the group.

## Examples

### Get a Group

```
curl -u "username:apiKey" --digest -i "https://cloud.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://cloud.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://cloud.mongodb.com/api/public/v1.0/groups"
{
  "name": "API Example 2"
}'
```

```
HTTP/1.1 201 Created
```

```
Location: https://cloud.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://cloud.mongodb.com/api/public/v1.0/groups/5329c8dfe4b0b07a83d67e7d/users"
[
  {
    "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://cloud.mongodb.com/api/public/v1.0/groups/5356823bc0ed0"
HTTP/1.1 200 OK

```

### Get Users in a Group

```

curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/5356823bc0ed0/users"

```

HTTP/1.1 200 OK

```

{
  "totalCount" : 2,
  "results" : [
    {
      "id" : "5357e25a300490374243f425",
      "username" : "user1@example.com",
      "emailAddress" : "user1@example.com",
      "firstName" : "User",
      "lastName" : "One",
      "roles" : [
        {
          "groupId" : "5356823bc0edc2788a835ed0",
          "roleName" : "GROUP_USER_ADMIN"
        }
      ],
      "links" : [ ... ]
    }
  ]
}

```

```

}, {
  "id" : "5356823b3004dee37132bb7b",
  "username" : "user2@example.com",
  "emailAddress" : "user2@example.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://cloud.mongodb.com/api/public/v1.0/groups/5356823b3004dee37132bb7b"
HTTP/1.1 200 OK
```

## Users

### On this page

- Endpoints
- Sample Entity
- Entity Fields
- Links
- Examples

### 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 a User** Create a new user. All fields are required.

```
POST /api/public/v1.0/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
```

**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@qa.example.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	string	The name of the role. Possible values are: <ul style="list-style-type: none"><li>• GLOBAL_AUTOMATION_ADMIN</li><li>• GLOBAL_BACKUP_ADMIN</li><li>• GLOBAL_MONITORING_ADMIN</li><li>• GLOBAL_OWNER</li><li>• GLOBAL_READ_ONLY</li><li>• GLOBAL_USER_ADMIN</li><li>• GROUP_AUTOMATION_ADMIN</li><li>• GROUP_BACKUP_ADMIN</li><li>• GROUP_MONITORING_ADMIN</li><li>• GROUP_OWNER</li><li>• GROUP_READ_ONLY</li><li>• GROUP_USER_ADMIN</li></ul>

## Links

Relation	Description
self	Me
<a href="http://mms.mongodb.com/whitelist">http://mms.mongodb.com/whitelist</a>	The user's whitelist.

## Examples

### Get a User by ID

```
curl -u "username:apiKey" --digest -i "https://cloud.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://cloud.mongodb.com/api/public/v1.0/groups/533daa30879bb2da07807696"

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 the First User

```
curl -H "Content-Type: application/json" -i -X POST "http://<ops-manager-url>:<port>/api/public/v1.0/
{
  "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://cloud.mongodb.com/api/public/v1.0/users"
{
  "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://cloud.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://cloud.mongodb.com/api/v1.0/groups/533dc19ce4b00835ff81e2eb/users/jane"
{
    "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

- [Overview](#)
- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

### Overview

The `alerts` resource allows you to retrieve alerts by their status, alert ID, or `alert configuration` and to acknowledge alerts.

When Ops Manager detects an alert condition, it opens an alert. If the alert configuration contains no notification delay, the alert status goes immediately to OPEN. If the configuration contains a delay, Ops Manager sets the alert to TRACKING until the delay period ends, after which Ops Manager sets the alert to OPEN if the condition persists.

If an alert configuration has multiple notifications, each with its own notification delay, Ops Manager uses the smallest delay value to determine when to move an alert from TRACKING to OPEN. An alert configuration sets notification delay in the `delayMin` field in the notification array.

## Endpoints

**Get All Alerts** Gets all alerts regardless of status.

```
GET /api/public/v1.0/groups/GROUP-ID/alerts
```

Use the `status` query parameter with one of these possible values: TRACKING, OPEN, or CLOSED to get all alerts with the specified status.

```
GET /api/public/v1.0/groups/GROUP-ID/alerts?status=STATUS
```

**Get an Alert** Gets a single alert by ID.

```
GET /api/public/v1.0/groups/GROUP-ID/alerts/ALERT-ID
```

**Get Alert Configurations that Triggered an Alert** Gets the alert configuration(s) that triggered the specified alert.

```
GET /api/public/v1.0/groups/GROUP-ID/alerts/ALERT-ID/alertConfigs
```

**Acknowledge an Alert** Updates an existing alert. The only field you may modify is the `acknowledgedUntil` field.

```
PATCH /api/public/v1.0/groups/GROUP-ID/alerts/ALERT-ID
```

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

The fields in the return document depend on the alert type. The `typeName` field specifies the alert type. The fields shown here are common to all alert types.

```
{
  "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",
  // Additional fields follow, depending on the type of alert
  "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	string	The type of alert. Possible values are: <ul style="list-style-type: none"> <li>HOST</li> <li>HOST_METRIC</li> <li>AGENT</li> <li>BACKUP</li> <li>GROUP</li> <li>REPLICA_SET</li> <li>USER</li> </ul>
eventTypeName	string	The name of the event that triggered the alert. The possible values here depend on the typeName: <ul style="list-style-type: none"> <li><b>HOST alert type. Possible values:</b> <ul style="list-style-type: none"> <li>- HOST_DOWN</li> <li>- HOST_RECOVERING</li> <li>- VERSION_BEHIND</li> <li>- HOST_EXPOSED</li> </ul> </li> <li><b>HOST_METRIC alert type. Possible value:</b> <ul style="list-style-type: none"> <li>- OUTSIDE_METRIC_THRESHOLD</li> </ul> </li> <li><b>AGENT alert type. Possible values:</b> <ul style="list-style-type: none"> <li>- MONITORING_AGENT_DOWN</li> <li>- MONITORING_AGENT_VERSION_BEHIND</li> <li>- BACKUP_AGENT_DOWN</li> <li>- BACKUP_AGENT_VERSION_BEHIND</li> <li>- BACKUP_AGENT_CONF_CALL_FAILURE</li> </ul> </li> <li><b>BACKUP alert type. Possible values:</b> <ul style="list-style-type: none"> <li>- OPLOG_BEHIND</li> <li>- CLUSTER_MONGOS_IS_MISSING</li> <li>- RESYNC_REQUIRED</li> <li>- RS_BIND_ERROR</li> <li>- BACKUP_TOO_MANY_RETRIES</li> <li>- BACKUP_IN_UNEXPECTED_STATE</li> <li>- LATE_SNAPSHOT</li> <li>- BAD_CLUSTERSHOTS</li> </ul> </li> </ul>

## Links

Relation	Description
<code>self</code>	Me
<code>http://mms.mongodb.net</code>	The group that this alert was triggered for.
<code>http://mms.mongodb.net</code>	The configuration that triggered this alert.
<code>http://mms.mongodb.net</code>	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 <code>alertConfig</code> link instead.
<code>http://mms.mongodb.net</code>	The host that triggered this alert. Only present for alerts of type HOST or REPLICASET.
<code>http://mms.mongodb.net</code>	The replicaset or sharded cluster that triggered this alert. Only present for alerts of type REPLICASET or BACKUP.

## Examples

## Get an Alert

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022e0000000001"
```

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 All Open Alerts

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022e0000000001"
```

HTTP/1.1 200 OK

```

        "number" : 0.0,
        "units" : "RAW"
    },
    "links" : [ ... ]
} ],
"links" : [ ... ]
}

```

## Get Alert Configurations that Triggered an Alert

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e"
```

```
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@example.com"
    } ],
    "links" : [ ... ]
  } ],
  "links" : [ ... ]
}
```

## Acknowledge an Alert

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i -X PATCH "https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e/alerts/533dc45ee4b00835ff81ec2a"
```

```
{
  "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@example.com",
  "created" : "2014-04-03T20:28:14Z",
  "updated" : "2014-04-03T20:33:14Z",
  "lastNotified" : "2014-04-03T20:33:23Z",
  "lastAcknowledged" : "2014-04-03T20:33:23Z"
}
```

```
"metricName": "ASSERTS_REGULAR",
"currentValue" : {
    "number" : 0.0,
    "units" : "RAW"
},
"links" : [ ... ]
}
```

## Alert Configurations

### On this page

- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

An *alert configuration* defines the conditions that trigger an alert and the methods of notification.

### Endpoints

**Get All Alert Configurations in a Group** Get all alert configurations for a group.

```
GET /api/public/v1.0/groups/GROUP-ID/alertConfigs
```

**Get an Alert Configuration** Get a single alert configuration by its ID.

```
GET /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID
```

**Get All Open Alerts Triggered by an Alert Configuration** Get all open alerts that were triggered by an alert configuration.

```
GET /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID/alerts
```

**Create an Alert Configuration** All fields are required except `created` and `updated`.

```
POST /api/public/v1.0/groups/GROUP-ID/alertConfigs
```

**Update an Alert Configuration** For most updates, you must use `PUT` and send the entire entity. The only exception is for updates to the `enabled` field, as described in [Enable or Disable an Alert Configuration](#).

```
PUT /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID
```

**Enable or Disable an Alert Configuration** Use to enable/disable an alert configuration by setting the `enabled` field.

```
PATCH /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID
```

## Delete an Alert Configuration

```
DELETE /api/public/v1.0/groups/GROUP-ID/alertConfigs/ALERT-CONFIG-ID
```

## 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": "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": "SNMP",
      "intervalMin": 5,
      "delayMin": 0,
      "snmpAddress": "example.com:161"
    },
    {
      "typeName": "EMAIL",
      "intervalMin": 5,
      "delayMin": 0,
      "emailAddress": "somebody@example.com"
    },
    {
      "typeName": "SMS",
      "intervalMin": 5,
      "delayMin": 0,
      "mobileNumber": "(212) 212-1212"
    }
  ]
}
```

```

    "typeName": "HIP_CHAT",
    "intervalMin": 5,
    "delayMin": 0,
    "notificationToken": "123456abcdef",
    "roomName": "My Chat Room"
},
{
    "typeName": "SLACK",
    "intervalMin": 5,
    "delayMin": 0,
    "channelName": "my-channel",
    "apiToken": "abcdef123456abcdef123456abcdef"
},
{
    "typeName": "FLOWDOCK",
    "intervalMin": 5,
    "delayMin": 0,
    "orgName": "my-org",
    "flowName": "my-flow",
    "flowdockApiToken": "abcdef123456abcdef123456abcdef12"
},
{
    "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 groupId	string string	Unique identifier. ID of the group that owns this alert configuration.
typeName	string	The type of this alert configuration. Supports the same values as the <code>typeName</code> field of the <code>alerts</code> resource. <ul style="list-style-type: none"> <li>• HOST</li> <li>• HOST_METRIC</li> <li>• AGENT</li> <li>• BACKUP</li> <li>• GROUP</li> <li>• REPLICA_SET</li> <li>• USER</li> </ul>

Continued on next page

Table 2 – continued from previous page

Name	Type	Description
eventTypeName	string	<p>The type of event that will trigger an alert. The possible values depend on the <code>typeName</code>:</p> <ul style="list-style-type: none"> <li>• <b>HOST alert type. Possible values:</b> <ul style="list-style-type: none"> <li>- HOST_DOWN</li> <li>-</li> <li>- HOST_RECOVERING</li> <li>-</li> <li>- VERSION_BEHIND</li> <li>- HOST_EXPOSED</li> </ul> </li> <li>• <b>HOST_METRIC alert type. Possible value:</b> <ul style="list-style-type: none"> <li>- OUTSIDE_METRIC_THRESHOLD</li> </ul> </li> <li>• <b>AGENT alert type. Possible values:</b> <ul style="list-style-type: none"> <li>- MONITORING_AGENT_DOWN</li> <li>- MONITORING_AGENT_VERSION_BEHIND</li> <li>- BACKUP_AGENT_DOWN</li> <li>- BACKUP_AGENT_VERSION_BEHIND</li> <li>- BACKUP_AGENT_CONF_CALL_FAILURE</li> </ul> </li> <li>• <b>BACKUP alert type. Possible values:</b> <ul style="list-style-type: none"> <li>- OPLOG_BEHIND</li> <li>- CLUSTER_MONGOS_IS_MISSING</li> <li>- RESYNC_REQUIRED</li> <li>- RS_BIND_ERROR</li> <li>- BACKUP_TOO_MANY_RETRIES</li> <li>- BACKUP_IN_UNEXPECTED_STATUS</li> <li>- LATE_SNAPSHOT</li> <li>- BAD_CLUSTERSHOTS</li> <li>- SYNC_SLICE_HAS_NOT_PROGRESS</li> </ul> </li> <li>• <b>GROUP alert type. Possible values:</b> <ul style="list-style-type: none"> <li>- USERS_AWAITING_APPROVAL</li> <li>- USERS_WITHOUT_MULTI_FACTOR</li> </ul> </li> <li>• <b>REPLICA_SET alert type. Possible values:</b> <ul style="list-style-type: none"> <li>- CONFIGURATION_CHANGED</li> <li>- PRIMARY_ELECTED</li> <li>- TOO_FEW_HEALTHY_MEMBERS</li> </ul> </li> </ul>

Table 2 – continued from previous page

Name	Type	Description
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 <i>all</i> 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: <ul style="list-style-type: none"><li>• <b>AGENT</b> - None.</li><li>• <b>BACKUP</b> - None.</li><li>• <b>HOST</b> and <b>HOST_METRIC</b><ul style="list-style-type: none"><li>- Possible values are:<ul style="list-style-type: none"><li>- HOSTNAME</li><li>- PORT</li><li>-</li><li>- HOSTNAME_AND_PORT</li><li>- REPLICA_SET_NAME</li><li>- TYPE_NAME</li></ul></li><li>• <b>REPLICA_SET</b> - Possible values are:<ul style="list-style-type: none"><li>- REPLICA_SET_NAME</li><li>- SHARD_NAME</li><li>- CLUSTER_NAME</li></ul></li></ul></li></ul>
matchers.operator	string	The operator to test the field's value. Possible values are: <ul style="list-style-type: none"><li>• EQUALS</li><li>• NOT_EQUALS</li><li>• CONTAINS</li><li>• NOT_CONTAINS</li><li>• STARTS_WITH</li><li>• ENDS_WITH</li><li>• REGEX</li></ul>
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: <ul style="list-style-type: none"><li>• PRIMARY</li><li>• SECONDARY</li><li>• STANDALONE</li><li>• CONFIG</li><li>• MONGOS</li></ul>

Continued on next page

Table 2 – continued from previous page

Name	Type	Description
notifications	object array	Notifications to send when an alert condition is detected.
notifications.typeName	string	The type of alert notification. Possible values are: <ul style="list-style-type: none"> <li>• GROUP</li> <li>• USER</li> <li>• SNMP (Only available to Ops Manager installations)</li> <li>• EMAIL</li> <li>• SMS (Only available to MongoDB Cloud Manager installations)</li> <li>• HIPCHAT</li> <li>• SLACK</li> <li>• FLOWDOCK</li> <li>• PAGER_DUTY</li> </ul>
notifications.intervalMin	number	The number of minutes to wait between successive notifications for unacknowledged alerts that are not resolved.
notifications.delayMin	number	The number of minutes to wait after an alert condition is detected before sending out the first notification.
notifications.groupID	string	ID of the group to which to send the notification.
notifications.groupName	string	Name of the group to which to send the notification.
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	The email address to which to send notification. Only present for notifications of type EMAIL.

Continued on next page

Table 2 – continued from previous page

Name	Type	Description
notifications.notification	string	A 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.channelName	string	The Slack channel name. Only present for SLACK notifications.
notifications.apiToken	string	The Slack API token or Bot token. Only present for SLACK notifications.
notifications.orgName	string	The Flowdock organization name in lower-case letters. This is the name that appears after www.flowdock.com/app/ in the URL string. Only present for FLOWDOCK notifications.
notifications.flowName	string	The flow name as it appears in the “flow email address” setting in Flowdock. For example: flowname@example.flowdock.com.
notifications.flowdockApiToken	string	The Flowdock “personal API token.” Only present for FLOWDOCK notifications.
notifications.serviceKey	string	PagerDuty service key. Only present for PAGER_DUTY notifications.
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: <ul style="list-style-type: none"> <li>• GREATER_THAN</li> <li>• LESS_THAN</li> </ul>
metricThreshold.threshold	number	The threshold value outside of which an alert will be triggered.

Continued on next page

Table 2 – continued from previous page

Name	Type	Description
metricThreshold.units	string	<p>The units for the threshold value. Depends on the type of metric. For example, a metric that measures memory consumption would have a byte measurement, while a metric that measures time would have a time unit. Possible values are:</p> <ul style="list-style-type: none"> <li>• RAW</li> <li>• BITS</li> <li>• BYTES</li> <li>• KILOBITS</li> <li>• KILOBYTES</li> <li>• MEGABITS</li> <li>• MEGABYTES</li> <li>• GIGABITS</li> <li>• GIGABYTES</li> <li>• TERABYTES</li> <li>• PETABYTES</li> <li>• MILLISECONDS</li> <li>• SECONDS</li> <li>• MINUTES</li> <li>• HOURS</li> <li>• DAYS</li> </ul>
metricThreshold.mode	string	<p>The mode to use when computing the current metric value. Possible values are:</p> <ul style="list-style-type: none"> <li>• AVERAGE</li> <li>• TOTAL</li> </ul>

## Links

Relation	Description
self	Me
<a href="http://mms.mongodb.com/group">http://mms.mongodb.com/group</a>	The group that owns this alert configuration.
<a href="http://mms.mongodb.com/alerts">http://mms.mongodb.com/alerts</a>	Open alerts triggered by this alert configuration.

## Examples

### Get All Alert Configurations in a Group

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e"
```

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@example.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.example.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://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022
```

```
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.example.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://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022
```

```
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,
      "unit" : "RAW"
    }
  }]
}
```

```

        "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://cloud.mongodb.com/api/public/v1.0/groups/4d1b6314e528c81a1f200e03/alertConfigs"
{
    "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://cloud.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://cloud.mongodb.com/api/v1/groups/4d1b6314e528c81alf200e03/alerts?_id=5357ce3e3004d83bd9c7864c"
{
  "groupId" : "4d1b6314e528c81alf200e03",
  "typeName" : "REPLICA_SET",
  "eventTypeName" : "RESYNC_REQUIRED",
  "enabled" : true,
  "matchers" : [ {
    "fieldName" : "REPLICA_SET_NAME",
    "operator" : "EQUALS",
    "value" : "rs1"
  }],
  "notifications" : [ {
    "typeName" : "EMAIL",
    "emailAddress" : "sos@example.com",
    "intervalMin" : 60,
    "delayMin" : 5
  },
  {
    "typeName" : "GROUP",
    "intervalMin" : 120,
    "delayMin" : 60,
    "smsEnabled" : true,
    "emailEnabled" : false
  }]
}
}
```

HTTP/1.1 200 OK

```
{
  "id" : "5357ce3e3004d83bd9c7864c",
  "groupId" : "4d1b6314e528c81alf200e03",
  "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@example.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://cloud.mongodb.com/api/public/v1.0/groups/4d1b6314e528c81a1f200e03/alerts/5357ce3e3004d83bd9c7864c"
{
  "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@example.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://cloud.mongodb.com/api/public/v1.0/groups/4d1b6314e528c81a1f200e03/alerts/5357ce3e3004d83bd9c7864c"
```

HTTP/1.1 200 OK

## Maintenance Windows

### On this page

- Overview
- Endpoints
- Sample Entity
- Entity Fields
- Links
- Examples

## Overview

The maintenanceWindows resource turns off alert notifications for specified alert types for a period of time, to allow maintenance to occur.

## Endpoints

### Get All Maintenance Windows

Get all maintenance windows with end dates in the future.

```
GET /api/public/v1.0/groups/GROUP-ID/maintenanceWindows
```

### Get a Single Maintenance Window by its ID

```
GET /api/public/v1.0/groups/GROUP-ID/maintenanceWindows/WINDOW-ID
```

### Create a Maintenance Window

```
POST /api/public/v1.0/groups/GROUP-ID/maintenanceWindows
```

The resource requires the following fields:

- alertTypeNames
- startDate
- endDate

### Update a Maintenance Window

```
PATCH /api/public/v1.0/groups/GROUP-ID/maintenanceWindows/WINDOW-ID
```

### Delete a Maintenance Window

```
DELETE /api/public/v1.0/groups/GROUP-ID/maintenanceWindows/WINDOW-ID
```

## Sample Entity

The following is a sample return document.

```
{
  "id" : "5628faffd4c606594adaa3b2",
  "groupId" : "541ed2009436399a1f54e01b",
  "created" : "2015-10-22T15:04:31Z",
  "updated" : "2015-10-22T15:04:31Z",
  "startDate" : "2015-10-23T22:00:00Z",
  "endDate" : "2015-10-23T23:30:00Z",
  "alertTypeNames" : [ "REPLICA_SET" ],
  "description" : "new description",
  "links" : [ ... ]
}
```

## Entity Fields

Name	Type	Description
id	string	unique identifier
groupId	string	ID of the group that owns this maintenance window config
created	date	When this maintenance window config was created.
updated	date	When this maintenance window config was last updated.
startDate	date	Start date for the maintenance window.
endDate	date	The end date for the maintenance window. It must be after startDate.
alertTypeNames	string array	Alert types to silence during maintenance window. For example: HOST, REPLICA_SET, AGENT, BACKUP
description	string	Description of the maintenance window. This field might not exist.

## Links

Relation	Description
self	Me
group	The group that owns this maintenance window config.

## Examples

**Get All Maintenance Windows** Get all maintenance windows for a group with start dates in the future.

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/541ed2009436399a1f54e01b"
```

HTTP/1.1 200 OK

```
{
  "results" : [ {
    "alertTypeNames" : [ "BACKUP" ],
    "created" : "2015-10-22T15:04:31Z",
    "description" : "new description",
    "endDate" : "2015-10-23T23:30:00Z",
    "groupId" : "541ed2009436399a1f54e01b",
    "id" : "5628faaffd4c606594adaaa3b2",
    "startDate" : "2015-10-23T22:00:00Z",
    "updated" : "2015-10-22T15:04:31Z",
    "links": [ ... ]
  }, {
    "alertTypeNames" : [ "AGENT", "BACKUP" ],
    "created" : "2015-10-22T15:40:09Z",
    "endDate" : "2015-10-23T23:30:00Z",
    "groupId" : "541ed2009436399a1f54e01b",
    "id" : "56290359d4c606594adaafe8",
    "startDate" : "2015-10-23T22:00:00Z",
    "updated" : "2015-10-22T15:40:09Z",
    "links": [ ... ]
  },
  "links" : [ ... ],
  "totalCount" : 2
}
```

## Get a Single Maintenance Window by its ID

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/541ed2009436399a1f54e01b/maintenanceWindows"
HTTP/1.1 200 OK

{
  "alertTypeNames" : [ "AGENT", "BACKUP" ],
  "created" : "2015-10-22T15:04:31Z",
  "description" : "new description",
  "endDate" : "2015-10-23T23:30:00Z",
  "groupId" : "541ed2009436399a1f54e01b",
  "id" : "5628faffd4c606594adaa3b2",
  "startDate" : "2015-10-23T22:00:00Z",
  "updated" : "2015-10-22T15:04:31Z",
  "links" : [ ... ]
}
```

## Create a Maintenance Window

```
curl -i -u "username:apiKey" --digest -X POST -H "Content-Type: application/json" "https://cloud.mongodb.com/api/public/v1.0/groups/541ed2009436399a1f54e01b/maintenanceWindows"
{
  "startDate" : "2015-10-23T22:00:00Z",
  "endDate" : "2015-10-23T23:30:00Z",
  "alertTypeNames" : [ "REPLICA_SET" ],
  "description" : "new description"
}'
```

HTTP/1.1 200 Created

Location: <https://cloud.mongodb.com/api/public/v1.0/groups/541ed2009436399a1f54e01b/maintenanceWindows>

```
{
  "alertTypeNames" : [ "REPLICA_SET" ],
  "created" : "2015-10-22T20:49:35Z",
  "description" : "my description",
  "endDate" : "2015-10-23T23:30:00Z",
  "groupId" : "541ed2009436399a1f54e01b",
  "id" : "56294bdf4c643eef5176b73",
  "startDate" : "2015-10-23T22:00:00Z",
  "updated" : "2015-10-22T20:49:35Z",
  "links" : [ ... ]
}
```

## Update a Maintenance Window

```
curl -i -u "username:apiKey" --digest -X PATCH -H "Content-Type: application/json" "https://cloud.mongodb.com/api/public/v1.0/groups/541ed2009436399a1f54e01b/maintenanceWindows/56294bdf4c643eef5176b73"
{
  "alertTypeNames" : [ "HOST", "REPLICA_SET" ]
}'
```

HTTP/1.1 201 OK

```
{
  "alertTypeNames" : [ "HOST", "REPLICA_SET" ],
  "created" : "2015-10-22T20:49:35Z",
  "description" : "my description",
  "endDate" : "2015-10-23T23:30:00Z",
  "groupId" : "541ed2009436399a1f54e01b",
  "id" : "56294bdf4c643eef5176b73",
  "startDate" : "2015-10-23T22:00:00Z",
```

```
"updated" : "2015-10-23T16:11:49Z",
"links" : [ ... ]
}
```

## Delete a Maintenance Window

```
curl -i --digest -u "username:apiKey" -X DELETE "https://cloud.mongodb.com/api/public/v1.0/groups/541.../maintenanceWindows/55.../"

HTTP/1.1 200 OK

{ }
```

## Backup Configurations

### On this page

- [Overview](#)
- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

### Overview

A backup configuration determines the settings used to back up a sharded cluster or replica set. The `backupConfigs` resource lets you view and update backup configurations. In certain cases, you can also create backup configurations, as described in the [Update a Backup Configuration](#) section. The `backupConfigs` resource supports only the GET and PATCH methods.

To update or create a backup configuration, you must issue the request from an IP address on your user account's [whitelist](#). The `backupConfigs` resource accepts PATCH requests only from whitelisted IP addresses.

### Endpoints

#### Get All Backup Configurations for a Group

```
GET /api/public/v1.0/groups/GROUP-ID/backupConfigs
```

**Get a Single Backup Configuration** 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/CLUSTER-ID
```

**Update a Backup Configuration** To change the state of an existing backup configuration you use the PATCH method. Send **only those fields** where you will change the value.

You **cannot** modify the `username` and `password` fields for a replica set or sharded cluster that is managed by Ops Manager Automation.

PATCH 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.

```
PATCH /api/public/v1.0/groups/GROUP-ID/backupConfigs/CLUSTER-ID
```

When modifying the `statusName` property, these are the acceptable transitions:

- `STARTED`: valid if the current status is `STOPPED` or `INACTIVE`.
- `STOPPED`: valid if the current status is `STARTED`.
- `TERMINATING`: valid if the current status is `STOPPED`.

You **cannot** change the `statusName` to `INACTIVE` or `PROVISIONING`.

Every *sharded cluster* and *replica set* has a backup configuration that is set to `INACTIVE` by default. If you *have already activated Backup for your account*, then you can back up a cluster or replica set through the API by changing the `statusName` property to `STARTED`.

## Sample Entity

If you issue a PATCH request, send only those fields whose values are being changed.

```
{
  "groupId": "xxx",
  "clusterId": "yyy",
  "statusName": "STARTED",
  "storageEngineName" : "WIRED_TIGER",
  "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	string	The current (or desired) status of the backup configuration. Possible values are: <ul style="list-style-type: none"> <li>• INACTIVE</li> <li>• PROVISIONING</li> <li>• STARTED</li> <li>• STOPPED</li> <li>• TERMINATING</li> </ul>
storageEngine	string	The storage engine used for the backup. Possible values are: <ul style="list-style-type: none"> <li>• MEMORY_MAPPED</li> <li>• WIRED_TIGER</li> </ul>
authMechanismName	string	The name of the authentication mechanism to use when connecting to the sync source database. Only present when using authentication. Possible values are: <ul style="list-style-type: none"> <li>• MONGODB_CR</li> <li>• GSSAPI</li> <li>• PLAIN</li> <li>• MONGODB_X509</li> <li>• NONE</li> </ul>
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. You can send this field to Ops Manager only when updating the backup configuration for a replica set or sharded cluster that is <b>not</b> managed by Ops Manager Automation.
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. GET requests do <i>not</i> include this field. You can send this field to Ops Manager only when updating the backup configuration for a replica set or sharded cluster that is <b>not</b> managed by Ops Manager Automation.
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 re-

## Links

Relation	Description
self <a href="http://mms.mongodb.com/cluster">http://mms.mongodb.com/cluster</a>	Me
group <a href="http://mms.mongodb.com/snapshotSchedule">http://mms.mongodb.com/snapshotSchedule</a>	The cluster that this backup configuration is for. The group that owns this backup configuration.
	The snapshot schedule for this backup configuration.

## Examples

### Get a Single Backup Configuration

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e"
```

HTTP/1.1 200 OK

```
{  
  "groupId" : "5196d3628d022db4cbc26d9e",  
  "clusterId" : "5196e5b0e4b0fca9cc88334a",  
  "statusName" : "STARTED",  
  "storageEngineName" : "WIRED_TIGER",  
  "sslEnabled" : false,  
  "excludedNamespaces" : [ ],  
  "links" : [ ... ]  
}
```

### Get All Backup Configurations for a Group

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/5196d3628d022db4cbc26d9e"
```

HTTP/1.1 200 OK

```
{  
  "totalCount" : 3,  
  "results" : [ {  
    "groupId" : "5196d3628d022db4cbc26d9e",  
    "clusterId" : "5196e5b0e4b0fca9cc88334a",  
    "statusName" : "STARTED",  
    "storageEngineName" : "WIRED_TIGER",  
    "sslEnabled" : false,  
    "excludedNamespaces" : [ ],  
    "links" : [ ... ]  
  }, {  
    "groupId" : "5196d3628d022db4cbc26d9e",  
    "clusterId" : "51a2ac88e4b0371c2dbf46ea",  
    "statusName" : "STARTED",  
    "storageEngineName" : "MEMORY_MAPPED",  
    "sslEnabled" : false,  
    "excludedNamespaces" : [ ],  
    "links" : [ ... ]  
  }, {  
    "groupId" : "5196d3628d022db4cbc26d9e",  
    "clusterId" : "52d33abee4b0ca49bc6acd6c",  
    "statusName" : "STOPPED",  
    "storageEngineName" : "WIRED_TIGER",  
    "sslEnabled" : false,  
    "excludedNamespaces" : [ ],  
    "links" : [ ... ]  
  } ]
```

```
        "links" : [ ... ]
    } ],
    "links" : [ ... ]
}
```

## Update a Backup Configuration

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X PATCH "https://cloud.mongodb.com/api/public/v1.0/groups/GROUP-ID/backupConfigs/CLUSTER-ID"
{
  "statusName": "STOPPED"
}

HTTP/1.1 202 Accepted

{
  "groupId" : "5196d3628d022db4cbc26d9e",
  "clusterId" : "5196e5b0e4b0fca9cc88334a",
  "statusName" : "STOPPED",
  "storageEngineName" : "WIRED_TIGER",
  "sslEnabled" : false,
  "excludedNamespaces" : [ ],
  "links" : [ ... ]
}
```

## Snapshot Schedule

### On this page

- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

This resource allows you to view and configure various properties of snapshot creation and retention for a replica set or cluster.

### Endpoints

**Get the Snapshot Schedule** Get the snapshot schedule 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/backupConfigs/CLUSTER-ID/snapshotSchedule
```

**Update the Snapshot Schedule** Change the parameters of snapshot creation and retention. Any combination of the snapshot schedule's attributes can be modified.

```
PATCH /api/public/v1.0/groups/GROUP-ID/backupConfigs/CLUSTER-ID/snapshotSchedule
```

## Sample Entity

```
{  
  "groupId": "525ec8394f5e625c80c7404a",  
  "clusterId": "5640e0cbe4b012df7a3fe85a",  
  "snapshotIntervalHours": 6,  
  "snapshotRetentionDays": 3,  
  "clusterCheckpointIntervalMin": 15,  
  "dailySnapshotRetentionDays": 14,  
  "weeklySnapshotRetentionWeeks": 6,  
  "monthlySnapshotRetentionMonths": 12,  
  "pointInTimeWindowHours": 24  
}
```

## 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	number	Number of hours between snapshots. Supported values are 6, 8, 12, and 24.
snapshotRetentionDays	number	Number of days to keep recent snapshots. Supported values are 1 - 5.
clusterCheckpointIntervalMinutes	number	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. Supported values are 15, 30, or 60.
dailySnapshotRetentionDays	number	Number of days to retain daily snapshots. Supported values are 1 - 365.
weeklySnapshotRetentionWeeks	number	Number of weeks to retain weekly snapshots. Supported values are 1 - 52.
monthlySnapshotRetentionMonths	number	Number of months to retain monthly snapshots. Supported values are 1 - 36.
pointInTimeWindowHours	number	The number of hours in the past for which a point-in-time snapshot can be created.

## Links

Relation	Description
self	Me
<a href="http://mms.mongodb.com/cluster">http://mms.mongodb.com/cluster</a>	The cluster that this backup configuration is for.
<a href="http://mms.mongodb.com/group">http://mms.mongodb.com/group</a>	The group that owns this backup configuration.
<a href="http://mms.mongodb.com/backupConfig">http://mms.mongodb.com/backupConfig</a>	The backup configuration that this schedule belongs to.

## Examples

### Get a Snapshot Schedule

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/525ec8394f5e625c80c7404a"
```

```
HTTP/1.1 200 OK
```

```
{
```

```

    "groupId" : "525ec8394f5e625c80c7404a",
    "clusterId" : "53bc556ce4b049c88baec825",
    "snapshotIntervalHours" : 6,
    "snapshotRetentionDays" : 2,
    "dailySnapshotRetentionDays" : 7,
    "weeklySnapshotRetentionWeeks" : 4,
    "monthlySnapshotRetentionMonths" : 13,
    "pointInTimeWindowHours": 24,
    "links": [ ... ]
}

```

## Update a Snapshot Schedule

```

curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X PATCH "https://cloud.mongodb.com/api/v1.0/groups/525ec8394f5e625c80c7404a/schedules/53bc556ce4b049c88baec825"
{
  "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,
  "pointInTimeWindowHours": 24,
  "links": [ ... ]
}

```

## Snapshots

### On this page

- [Overview](#)
- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

### Overview

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*.

You can retrieve snapshot metadata for the whole *cluster* or *replica set*, or for a single *config server* in a cluster.

In order to perform a restore of the snapshot data, you must create a *restore job*.

## Endpoints

**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
```

**Get a Single Snapshot for a Cluster**

```
GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/snapshots/SNAPSHOT-ID
```

**Delete a Snapshot** 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.

```
DELETE /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/snapshots/SNAPSHOT-ID
```

**Get All Snapshots for a Config Server** Get all snapshots for a *config server*. HOST-ID must be the ID of a config server.

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/snapshots
```

**Get a Single Snapshot for a Config Server**

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/snapshots/SNAPSHOT-ID
```

## Sample Entity

The following is a sample snapshot document.

```
{
  "clusterId" : "348938fbdbca74718cba",
  "complete" : true,
  "created" : {
    "date" : "2015-07-29T02:57:17Z",
    "increment" : 1
  },
  "expires" : "2016-07-27T02:57:17Z",
  "groupId" : "2847387cd717dabc348a",
  "id" : "5875f665da588548965b",
  "isPossiblyInconsistent" : false,
  "missingShards": [
    {
      "id": "472837bcdb278abcd7",
      "groupId": "2847387cd717dabc348a",
      "typeName": "REPLICA_SET",
      "clusterName": "Cluster 1",
      "shardName": "shard002",
      "replicaSetName": "rs2",
      "lastHeartbeat": "2014-02-26T17:32:45Z"
    }
  ],
  "parts" : [ {
    "clusterId" : "55b83f6ce4b074d9e2f2e741",
    "dataSizeBytes" : 177588,
    "fileSizeBytes" : 201326640,
    "lastModified" : "2015-07-29T02:57:17Z"
  }]
}
```

```
"mongodVersion" : "3.0.5",
"replicaSetName" : "rs0",
"storageSizeBytes" : 11235328,
"typeName" : "REPLICA_SET"
}, {
"clusterId" : "55b83f6ce4b074d9e2f2e73d",
"dataSizeBytes" : 3450212,
"fileSizeBytes" : 201326640,
"mongodVersion" : "3.0.5",
"replicaSetName" : "rs1",
"storageSizeBytes" : 21721088,
"typeName" : "REPLICA_SET"
}, {
"dataSizeBytes" : 3211448,
"fileSizeBytes" : 201326640,
"hostId" : "8e42be22e49d3002c1744f2f5c6327eb",
"mongodVersion" : "3.0.5",
"storageSizeBytes" : 25989120,
"typeName" : "CONFIG_SERVER"
} ]
}
```



## 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. Not present for a config server snapshot.
hostId	string	ID of the <i>config server</i> represented by the snapshot. Present only for a config server 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? <code>isPossiblyInconsistent</code> is only present for <i>sharded cluster</i> 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 <b>cluster</b> document containing a <code>self</code> 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	string	The type of server represented by the part. Possible values are: <ul style="list-style-type: none"> <li>• REPLICA_SET</li> <li>• CONFIG_SERVER</li> </ul>
parts.clusterId	string	ID of the replica set. Not present for a config server. <span style="float: right;">339</span>
parts.replicaSetName	string	Name of the replica set. Not present for a config server.
parts.hostId	string	ID of a config server. Not present for a replica set.

## Links

Relation	Description
self	Me
http://mms.mongodb.com/cluster	The cluster that this snapshot belongs to.
http://mms.mongodb.com/group	The group that owns this snapshot.

## Examples

### Get All Snapshots

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/525ec8394f5e0
```

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://cloud.mongodb.com/api/public/v1.0/groups/525ec8394f5e0
```

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://cloud.mongodb.com/api/public/v1.0/groups/52...  
HTTP/1.1 200 OK
```

## Checkpoints

### On this page

- [Overview](#)
- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

### Overview

This resource allows you to view *Checkpoints* metadata. You can use checkpoints to create custom snapshots of a cluster at points in time between regular snapshots.

### Endpoints

**Get All Checkpoints** Get all checkpoints for a cluster.

```
GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/checkpoints
```

**Get One Checkpoint** Get a single checkpoint.

```
GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/checkpoints/CHECKPOINT-ID
```

## Sample Entity

```
{  
    "clusterId" : "348938fbdbca74718cba",  
    "completed" : "2015-07-31T23:19:51Z",  
    "groupId" : "2847387cd717dabc348a",  
    "id" : "5875f665da588548965b",  
    "parts" : [ {  
        "replicaSetName" : "shard_1",  
        "shardName" : "shard_1",  
        "tokenDiscovered" : true,  
        "tokenTimestamp" : {  
            "date" : "2015-07-31T23:19:51Z",  
            "increment" : 1  
        },  
        "typeName" : "REPLICA_SET"  
    }, {  
        "replicaSetName" : "shard_0",  
        "shardName" : "shard_0",  
        "tokenDiscovered" : true,  
        "tokenTimestamp" : {  
            "date" : "2015-07-31T23:19:51Z",  
            "increment" : 1  
        },  
        "typeName" : "REPLICA_SET"  
    }, {  
        "hostId" : "11d9c414d292453725d2232e25c02525",  
        "tokenDiscovered" : true,  
        "tokenTimestamp" : {  
            "date" : "2015-07-31T23:19:51Z",  
            "increment" : 6  
        },  
        "typeName" : "CONFIG_SERVER"  
    } ],  
    "restorable" : true,  
    "started" : "2015-07-31T23:19:51Z",  
    "timestamp" : "2015-07-31T23:19:47Z"  
}
```

## Entity Fields

Name	Type	Description
clusterId	string	The ID of the cluster represented by the checkpoint.
completed	BSON timestamp	The point-in-time the checkpoint completed and the balancer restarted.
groupId	string	The ID of the group that owns the checkpoint.
id	string	The checkpoint ID.
parts	array of parts	The individual parts that comprise the complete checkpoint. There will be one element for each shard plus one element for the config servers.
parts.replicaSetName	string	Name of the replica set. Not present for a config server.
parts.shardName	string	The name of the shard.
parts.tokenDiscovered	Boolean	Indicates whether the token exists.
parts.tokenTimestamp	document	The timestamp of an entry in the <i>oplog</i> , as specified by the entry's <code>ts</code> field. The <code>ts</code> field is a <a href="#">BSON</a> timestamp and has two components: the timestamp, which is the value in seconds since the Unix epoch, and the increment, which is an incrementing ordinal for operations within a given second.
parts.typeName	string	The type of server represented by the part. Possible values are: <ul style="list-style-type: none"> <li>• <code>REPLICA_SET</code>, which indicates the part is a shard.</li> <li>• <code>CONFIG_SERVER</code></li> </ul>
restorable	Boolean	Indicates whether the checkpoint can be used for a restore.
started	BSON timestamp	The point-in-time Ops Manager stopped the <a href="#">balancer</a> and began the checkpoint.
timestamp	BSON timestamp	The point-in-time the checkpoint restores to.

## Links

Relation	Description
<code>self</code>	The checkpoint
<code>http://mms.mongodb.com/cluster</code>	The cluster the checkpoint belongs to.
<code>http://mms.mongodb.com/group</code>	The group that owns the checkpoint.

## Examples

### Get All Checkpoints

```

curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/2847387cd717cd55bc0658e4b097a3efe06f1f"
HTTP/1.1 200 OK

{
  "links" : [ ... ],
  "results" : [ {
    "clusterId" : "55bbd551e4b07f5222418262",
    "completed" : "2015-07-31T23:35:52Z",
    "groupId" : "2847387cd717dabc348a",
    "id" : "55bc0658e4b097a3efe06f1f",
    "links" : [ {
      "href" : "https://cloud.mongodb.com/api/public/v1.0/groups/2847387cd717dabc348a/clusters/55bbd551e4b07f5222418262",
      "rel" : "self"
    } ],
    "parts" : [ {
      "replicaSetName" : "shard_1",
      "shardName" : "shard_1",
      "tokenDiscovered" : true,
      "tokenTimestamp" : {
        "date" : "2015-07-31T23:35:52Z",
        "increment" : 1
      },
      "typeName" : "REPLICA_SET"
    }, {
      "replicaSetName" : "shard_0",
      "shardName" : "shard_0",
      "tokenDiscovered" : true,
      "tokenTimestamp" : {
        "date" : "2015-07-31T23:35:52Z",
        "increment" : 1
      },
      "typeName" : "REPLICA_SET"
    }, {
      "hostId" : "11d9c414d292453725d2232e25c02525",
      "tokenDiscovered" : true,
      "tokenTimestamp" : {
        "date" : "2015-07-31T23:35:52Z",
        "increment" : 2
      },
      "typeName" : "CONFIG_SERVER"
    } ],
    "restorable" : true,
    "started" : "2015-07-31T23:35:52Z",
    "timestamp" : "2015-07-31T23:34:47Z"
  },
  ...
],
  "totalCount" : 6
}

```

## Get One Checkpoint

```

curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/2847387cd717cd55bc0658e4b097a3efe06f1f"
HTTP/1.1 200 OK

{

```

```

"clusterId" : "55bbd551e4b07f5222418262",
"completed" : "2015-07-31T23:19:51Z",
"groupId" : "2847387cd717dabc348a",
"id" : "55bc0297e4b00759dc7b8b8c",
"links" : [ ... ],
"parts" : [ {
    "replicaSetName" : "shard_1",
    "shardName" : "shard_1",
    "tokenDiscovered" : true,
    "tokenTimestamp" : {
        "date" : "2015-07-31T23:19:51Z",
        "increment" : 1
    },
    "typeName" : "REPLICA_SET"
}, {
    "replicaSetName" : "shard_0",
    "shardName" : "shard_0",
    "tokenDiscovered" : true,
    "tokenTimestamp" : {
        "date" : "2015-07-31T23:19:51Z",
        "increment" : 1
    },
    "typeName" : "REPLICA_SET"
}, {
    "hostId" : "11d9c414d292453725d2232e25c02525",
    "tokenDiscovered" : true,
    "tokenTimestamp" : {
        "date" : "2015-07-31T23:19:51Z",
        "increment" : 6
    },
    "typeName" : "CONFIG_SERVER"
} ],
"restorable" : true,
"started" : "2015-07-31T23:19:51Z",
"timestamp" : "2015-07-31T23:19:47Z"
}

```

## Restore Jobs

### On this page

- [Overview](#)
- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

### Overview

The `restoreJobs` 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](#).

## Endpoints

**Get All Restore Jobs for a Cluster** 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
```

Using the batchId query parameter, you can retrieve 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?batchId=BATCH-ID
```

### Get a Single Restore Job for a Cluster

```
GET /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/restoreJobs/JOB-ID
```

**Get All Restore Jobs for a Legacy Mirrored Config Server** Get all restore jobs for a legacy mirrored *config server* (a config server that is **not** a replica set). HOST-ID is the ID of the config server. To get restore jobs for a config server that **is** a replica set, use CLUSTER-ID.

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/restoreJobs
```

---

**Note:** For information on running a config server as a replica set, see [Create the Config Server Replica Set](#) in the MongoDB manual.

---

**Get a Single Restore Job for a Legacy Mirrored Config Server** Get a single restore job for a legacy mirrored *config server* (a config server that is **not** a replica set). HOST-ID is the ID of the config server. To get a single restore job for a config server that **is** a replica set, you must instead query for the restore job using CLUSTER-ID.

```
GET /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/restoreJobs/JOB-ID
```

**Create a Restore Job for a Cluster** Create a restore job for the specified CLUSTER-ID. You can create a restore job for either an existing snapshot or a custom snapshot. A custom snapshot is created from a specific point-in-time for a replica set or from a checkpoint for a sharded cluster.

```
POST /api/public/v1.0/groups/GROUP-ID/clusters/CLUSTER-ID/restoreJobs
```

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.

The [Create Restore Jobs](#) examples describe each restore job type.

**Create a Restore Job for a Legacy Mirrored Config Server** Create a restore job for a legacy mirrored *config server* (a config server that is **not** a replica set). HOST-ID is the ID of a config server that is not a replica set.

```
POST /api/public/v1.0/groups/GROUP-ID/hosts/HOST-ID/restoreJobs
```

## Sample Entity

The following is one example of a return document. The fields in a return document depend on the type of restore and other factors:

```
{  
  "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/...",  
    "expires" : "2014-07-09T18:42:43Z",  
    "expirationHours": 4,  
    "maxDownloads": 1,  
    "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.
hostId	string	ID of the <i>config server</i> to which this restore job belongs. Only present for a restore of a legacy mirrored <i>config server</i> .
createdTimestamp	timestamp BSON timestamp	When the restore job was requested. Timestamp of the latest oplog entry in the restored snapshot.
statusName	string	Current status of the job. Possible values are: <ul style="list-style-type: none"> <li>• FINISHED</li> <li>• IN_PROGRESS</li> <li>• BROKEN</li> <li>• KILLED</li> </ul>
pointInTimeDelivery	boolean object	Is this job for a point-in-time restore? Additional details about how the restored snapshot data will be delivered.
delivery.methodName	string	How the data will be delivered. Possible values are: <ul style="list-style-type: none"> <li>• HTTP</li> <li>• SCP</li> </ul>
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.expirationHours	number	The number of hours the download URL will be valid once the restore job is complete. Only present if delivery.methodName is HTTP.
delivery.maxDownloads	number	The number of times the download URL can be used. This must be 1 or greater. Only present if delivery.methodName is HTTP.
delivery.statusName	string	Current status of the downloadable file. Possible values are: <ul style="list-style-type: none"> <li>• NOT_STARTED</li> <li>• IN_PROGRESS</li> <li>• READY</li> <li>• FAILED</li> <li>• INTERRUPTED</li> </ul>
		349

## Links

Relation	Description
self	Me
<a href="http://mms.mongodb.com/cluster">http://mms.mongodb.com/cluster</a>	The cluster to restore.
<a href="http://mms.mongodb.com/snapshot">http://mms.mongodb.com/snapshot</a>	The snapshot to restore.
<a href="http://mms.mongodb.com/group">http://mms.mongodb.com/group</a>	The group that owns the cluster.

## Examples

### Get All Restore Jobs

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/525ec8394f5e...  
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/ae6bc7a8bfdd5a99a0c118c73845dc7...",  
            "expires" : "2014-07-09T18:43:21Z",  
            "expirationHours": 1,  
            "maxDownloads": 1,  
            "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/ae6bc7a8bfdd5a99a0c118c73845dc7...",  
            "expires" : "2014-07-09T18:42:43Z",  
            "expirationHours": 1,  
            "maxDownloads": 1,  
        }  
    } ]  
}
```

```

        "statusName" : "READY"
    },
    "links" : [ ... ]
}
},
"links": [ ... ]
}
```

## Get a Single Restore Job

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/525ec8394f5e0...
```

```
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/...",
    "expires" : "2014-07-09T18:42:43Z",
    "expirationHours": 1,
    "maxDownloads": 1,
    "statusName" : "READY"
  },
  "links" : [ ... ]
}
```

**Get a Restore Job for a Legacy Mirrored Config Server** Get a restore job for a legacy mirrored *config server* (a config server that is **not** a replica set).

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/groups/525ec8394f5e0...
```

```
HTTP/1.1 200 OK
```

```
{
  "created" : "2015-06-19T20:08:59Z",
  "delivery" : {
    "expires" : "2015-06-19T21:08:59Z",
    "methodName" : "HTTP",
    "statusName" : "READY",
    "url" : "https://api-backup.mongodb.com/backup/restore/v2/pull/fa16fef25e5753a9ff3a278d6e02f571/...",
    "expirationHours": 4,
    "maxDownloads": 2
  },
  "groupId" : "558452c9e4b06adf8b239d92",
  "hostId" : "0db2ee9eadfbcd225a60057abcd4352b",
  "id" : "558476dbe4b08b5fb379a698",
  "links" : [ ... ],
}
```

```

    "pointInTime" : false,
    "snapshotId" : "558475dce4b082d419ce05f3",
    "statusName" : "FINISHED",
    "timestamp" : {
        "date" : "2015-06-19T20:04:00Z",
        "increment" : 1
    }
}

```

## Create Restore Jobs

### Create a Restore Job for an Existing Snapshot via HTTPS

```

curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST "https://cloud.mongodb.com/api/v1.0/cluster//snapshot//restore"
{
    "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 Restore Job for an Existing Snapshot via SCP

```

curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST "https://cloud.mongodb.com/api/v1.0/cluster//snapshot//restore"
{
    "snapshotId": "53bd439ae4b0774946a16490",
    "delivery": {
        "methodName": "SCP",
        "formatName": "ARCHIVE",
        "hostname": "dbserver.example.com",
        "port": 22,
        "username": "admin",
        "password": "secret",
        "passwordTypeName": "PASSWORD",
        "targetDirectory": "/data/backup"
    }
}

```

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,
    "delivery": {
      "methodName": "SCP",
      "formatName": "ARCHIVE",
      "hostname": "dbserver.example.com",
      "port": 22,
      "username": "admin",
      "password": "secret",
      "passwordTypeName": "PASSWORD",
      "targetDirectory": "/data/backup"
    },
    "links" : [ ... ]
  } ]
}
```

### Create a Point-In-Time Restore Job for a Replica Set

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST "https://cloud.mongodb.com/api/v1.0/clusters//replicaSets//jobs" -d {
  "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" : [ ... ]
  } ]
}
```

## Create a Checkpoint Restore Job for a Sharded Cluster

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST "https://cloud.mongodb.com/api/public/v1.0/operations/checkpoints/restore?shardId=0&shardIndex=0&shardType=shard" -d '{ "checkpointId": "a74718cba2847387cd7" }'
```

## Create a Restore Job for a Legacy Mirrored Config Server

Create a restore job for a legacy mirrored *config server* (a config server that is **not** a replica set).

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST "https://cloud.mongodb.com/api/public/v1.0/operations/checkpoints/restore?shardId=0&shardIndex=0&shardType=shard" -d '{ "snapshotId": "558476dbe4b08b5fb379a698" }'
```

## Whitelist

### On this page

- [Endpoints](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Links](#)
- [Examples](#)

Retrieves and updates addresses on a user's *whitelist*. The resource modification operations POST and DELETE are themselves whitelisted. You can only add an IP address to a whitelist if the request originates from an IP address on the existing whitelist.

### Endpoints

#### Get a User's Whitelist

```
GET /api/public/v1.0/users/USER-ID/whitelist
```

Retrieves the IP 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 a Single Whitelist Entry

```
GET /api/public/v1.0/users/USER-ID/whitelist/IP-ADDRESS
```

Gets the whitelist entry for a single IP address.

#### Add Entries to a User's Whitelist

Add one or more IP addresses to the user's whitelist.

```
POST /api/public/v1.0/users/USER-ID/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 an Entry from a User's Whitelist

Remove an IP address from the whitelist.

```
DELETE /api/public/v1.0/users/USER-ID/whitelist/IP-ADDRESS
```

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 <i>only</i> updated when a resource that is protected by the whitelist is accessed.
count	number	The total number of requests that originated from this IP address. Note that this field is <i>only</i> updated when a resource that is protected by the whitelist is accessed.

## Links

Relation	Description
self	Me
<a href="http://mms.mongodb.com/user">http://mms.mongodb.com/user</a>	The user that owns this whitelist.

## Examples

### Get a User's Whitelist

```
curl -i -u "username:apiKey" --digest "https://cloud.mongodb.com/api/public/v1.0/users/5356823b3004de...
```

```
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://cloud.mongodb.com/api/public/v1.0/users/5356823b3004de...  
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://cloud.mongodb.com/api/public/v1.0/users/5356823b3004dee37132bb7b/whitelist"  
[ {  
    "ipAddress" : "76.54.32.10"  
  }, {  
    "ipAddress" : "2.3.4.5"  
} ]'  
  
HTTP/1.1 201 Created  
Location: https://cloud.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://cloud.mongodb.com/api/public/v1.0/users/5356823b3004dee37132bb7b/whitelist/  
HTTP/1.1 200 OK
```

## Automation Configuration Resource

## On this page

- [Overview](#)
- [Endpoints](#)
- [Other Representations of the Automation Configuration](#)
- [Automation Configuration Entity](#)
- [Examples](#)

## Overview

The Public API provides the `automationConfig` endpoint to let you manipulate your group's *automation configuration*. The configuration defines the various MongoDB clusters, replica sets and standalones in the deployment and defines how each process runs. The Automation Agents build the deployment according to the goals specified. Each Automation Agent is responsible for the MongoDB processes that run on its host. The configuration also defines the deployment's Monitoring and Backup Agents and can optionally specify which version of the Automation Agent to run.

When a running MongoDB 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.

For example automation configurations, please see the following page on GitHub: <https://github.com/10gen-labs/mms-api-examples/tree/master/automation/>.

## Endpoints

**Get the Automation Configuration** Retrieve the current automation configuration for a group.

```
GET /api/public/v1.0/groups/GROUP-ID/automationConfig
```

**Update the Automation Configuration** Update a group's automation configuration. For steps for updating an automation configuration, see [Deploy a Cluster through the API](#).

```
PUT /api/public/v1.0/groups/GROUP-ID/automationConfig
```

Use PUT to update a group's automation configuration. **Do not** use PATCH. For steps for updating an automation configuration, see [Update the Automation Configuration](#).

To make updates to the Monitoring Agent or Backup Agent *other than* `hostname`, you must do so through a different endpoint. See [Update the Monitoring Agent or Backup Agent](#).

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.

**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.

**Update the Monitoring Agent or Backup Agent** Some attributes of the *monitoringVersions* and *backupVersions* objects cannot be updated through the *automationConfig* endpoint. Ops Manager provides the following endpoints to update these fields.

---

**Important:** Do **not** use these endpoints to update the agent's host. To update *monitoringVersions.hostname* or *backupVersions.hostname*, use the endpoint to *update the entire configuration*.

---

Update Monitoring Agent attributes:

```
PUT /api/public/v1.0/groups/GROUP-ID/automationConfig/monitoringAgentConfig
```

Update Backup Agent attributes:

```
PUT /api/public/v1.0/groups/GROUP-ID/automationConfig/backupAgentConfig
```

The endpoints update the following agent attributes. Pass an object with all the attributes *that you use*, whether or not the value is changing. If you do not use a field, do not include it:

- `logPath` (required for non-Windows operating systems)
- `logPathWindows` (required for Windows operating systems)
- `logRotate`
  - `logRotate.sizeThresholdMB`
  - `logRotate.timeThresholdHrs`
  - `logRotate.numUncompressed`
  - `logRotate.percentOfDiskspace`
- `username`
- `password`
- `kerberosPrincipal`
- `kerberosKeytab`
- `sslPEMKeyFile`
- `sslPEMKeyFileWindows` (only required if deployment contains Windows processes)
- `sslPEMKeyPwd`

### Other Representations of the Automation Configuration

The Automation Agent 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.

Users with *global roles* can view Ops Manager's internal representation of the deployment configuration, which is much larger than the automation configuration and includes additional fields used only internally. **Never** use this representation to update the automation configuration. The representation is called the `Raw AutomationConfig` is viewable through the *Deployment* page.

### Automation Configuration Entity

This section describes the fields that comprise the automation configuration. For additional examples, see the following page on GitHub: <https://github.com/10gen-labs/mms-api-examples/tree/master/automation/>.

**This section includes the following:**

- Configuration Version
- Download Base
- MongoDB Versions Specifications
- Automation Agent
- Monitoring Agent
- Backup Agent
- MongoDB Instances
- Replica Sets
- Sharded Clusters
- Cluster Balancer
- Authentication
- SSL
- MongoDB Roles
- Kerberos
- Indexes

**Configuration Version** This lists the version of the automation configuration.

```
"version" : <integer>
```

Name	Type	Description
version	integer	The version of the configuration.

**Download Base** The download base is the path to the directory where automatic version downloads will be targeted and scripts for starting processes will be created.

```
"options" : {
    "downloadBase" : <string>,
    "downloadBaseWindows" : <string>
}
```

Name	Type	Description
options	object	The options object is required and must contain both the downloadBase and downloadBaseWindows fields.
options.downloadBase	string	The directory on Linux and Unix (including Mac OS X) platforms for automatic version downloads and startup scripts.
options.downloadBaseWindows	string	The directory on Windows platforms for automatic version downloads and startup scripts.

**MongoDB Versions Specifications** 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>,

```

```

        "modules" : [ <string>, ... ],
        "bits" : <integer>,
        "win2008plus" : <Boolean>,
        "winVCRestUrl" : <string>,
        "winVCRestOptions" : [ <string>, ... ],
        "winVCRestDll" : <string>,
        "winVCRestVersion" : <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.
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.
mongoDbVersions.builds	array of objects	Objects that define the builds for this MongoDB instance.
mongoDbVersions.builds.platform	string	The platform for this MongoDB instance.
mongoDbVersions.builds.url	string	The URL from which to download MongoDB for this instance.
mongoDbVersions.builds.gitIdentifier	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.
mongoDbVersions.builds.modules	array	The list of modules for this version. Corresponds to the modules field returned by MongoDB 3.2+ buildInfo command.
mongoDbVersions.builds.processorBits	integer	The processor's bus width. Specify either 64 or 32.
mongoDbVersions.builds.requiresWindows7OrLater	Boolean	Set to true if this is a Windows build that requires either Windows 7 later or Windows Server 2008 R2 or later.
mongoDbVersions.builds.microsoftVisualCPlusPlusRedistributableUrl	string	The URL from which the required version of the Microsoft Visual C++ redistributable can be downloaded.
mongoDbVersions.builds.microsoftVisualCPlusPlusRedistributableCommandLineOptions	array	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.
mongoDbVersions.builds.microsoftVisualCPlusPlusRuntimeDlLName	string	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.
mongoDbVersions.builds.microsoftVisualCPlusPlusRuntimeMinimumVersion	string	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.

**Automation Agent** The agentVersion object is optional and specifies the version of Automation Agent.

```

"agentVersion" : {
    "name" : <string>,
    "directoryUrl" : <string>
}

```

Name	Type	Description
agentVersion	object	<i>Optional</i> 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.
agentVersion.name	string	The desired version of the Automation Agent (e.g. “1.8.1.1042-1”).
agentVersion.downloadUrl	string	The URL from which to download Automation Agent.

**Monitoring Agent** The monitoringVersions array is optional and specifies the version of the Monitoring Agent.

```
"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	<i>Optional.</i> Objects that define version information for each Monitoring Agent.
monitoringVersions.name	string	The desired version of the Monitoring Agent (e.g. “2.9.1.176-1”). For MongoDB compatibility with Automation, see <a href="#">MongoDB Compatibility</a> .
monitoringVersions.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.
monitoringVersions.urls	object	The platform- and build-specific URLs from which to download the Monitoring Agent.
monitoringVersions.urlList	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.
monitoringVersions.baseUrl	string	The base URL used for the mmsBaseUrl setting in the <a href="#">Monitoring Agent Configuration</a> .
monitoringVersions.logPath	string	<i>Optional.</i> The directory where the agent stores its logs. The default is to store logs in /dev/null.
monitoringVersions.logs	object	<i>Optional.</i> Enables log rotation for the MongoDB logs for a process.
monitoringVersions.logRotate.sizeThresholdMB	number	The maximum size in MB for an individual log file before rotation.
monitoringVersions.logRotate.timeThresholdHrs	number	The maximum time in hours for an individual log file before rotation.
monitoringVersions.logRotate.numUncompressed	number	<i>Optional.</i> The maximum number of total log files to leave uncompressed, including the current log file. The default is 5. In earlier versions of Ops Manager, this field was named maxUncompressed. The earlier name is still recognized, though the new version is preferred.
monitoringVersions.logRotate.percentage	number	<i>Optional.</i> The maximum percentage of total disk space all log files should take up before deletion. The default is .02.

**Backup Agent** The backupVersions array is optional and specifies the version of the Backup Agent.

```
"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	<i>Optional.</i> Objects that define version information for each Backup Agent.
backupVersions.name	string	The desired version of the Backup Agent (e.g. “3.1.1.263-1”).
backupVersions.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.
backupVersions.urls	object	The platform- and build-specific URLs from which to download the Backup Agent.
backupVersions.urls.<object for os>.version	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.
backupVersions.baseUrl	string	The base URL used for the mothership and https settings in the <a href="#">Backup Agent Configuration</a> . For example, for baseUrl="https://cloud.mongodb.com, the backup configuration fields would have these values:
backupVersions.logPath	string	<i>Optional.</i> The directory where the agent stores its logs. The default is to store logs in /dev/null.
backupVersions.logRotate	object	<i>Optional.</i> Enables log rotation for the MongoDB logs for a process.
backupVersions.logRotate.name.size	number	The maximum size in MB for an individual log file before rotation.
backupVersions.logRotate.name.time	integer	The maximum time in hours for an individual log file before rotation.
backupVersions.logRotate.name.number	number	<i>Optional.</i> The maximum number of total log files to leave uncompressed, including the current log file. The default is 5.
backupVersions.logRotate.name.percentage	number	<i>Optional.</i> The maximum percentage of total disk space all log files should take up before deletion. The default is .02.

**MongoDB Instances** The processes array determines the configuration of your MongoDB instances. You can also use the array to restore an instance.

```

"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>,
    "backupRestoreUrl" : <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.
processes.name	string	A unique name to identify the instance.
processes.processType	string	Either mongod or mongos.
processes.version	string	The name of the mongoDbVersions specification used with this instance.
processes.<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 <a href="#">supported configuration options</a> .
processes.disabled	Boolean	<i>Optional.</i> Set to true to shut down the process.
processes.manualMode	Boolean	<i>Optional.</i> Set to true to operate this process in manual mode. The Automation Agent will take no actions on the process.
processes.hostname	string	<i>Optional.</i> The name of the host this process should run on. This defaults to localhost.
processes.cluster	string	<i>Optional.</i> Required for a mongos. The name of the cluster. This must correspond to the sharding.name field in the sharding array for the mongos.
processes.numCores	integer	<i>Optional.</i> The number of cores the process should be bound to. The Automation Agent will spread processes out across the cores as evenly as possible.
processes.logRotate	object	<i>Optional.</i> Enables log rotation for the MongoDB logs for a process.
processes.logRotate.sizeThresholdMB	number	The maximum size in MB for an individual log file before rotation. The file rotates immediately if the file meets either this sizeThresholdMB or the processes.logRotate.timeThresholdHrs limit.
processes.logRotate.timeThresholdHrs	integer	The maximum runtime in hours for an individual log file before the next rotation. The time is since the last rotation. The log file rotates immediately if the file meets either this timeThresholdHrs or the processes.logRotate.sizeThresholdMB limit.
processes.logRotate.uncompressedCount	integer	<i>Optional.</i> The maximum number of total log files to leave uncompressed, including the current log file. The default is 5.
processes.logRotate.uncompressedPercentage	number	<i>Optional.</i> The maximum percentage of total disk space that can be used to store the log files. If this limit is exceeded, the compressed log files are deleted to meet this limit, starting with the oldest log files first. The default is .02.
processes.authSchemaVersion	integer	<i>Optional.</i> 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.
processes.alias	string	<i>Optional.</i> 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.
processes.backupRestoreUrl	string	<i>Optional.</i> This is used only when creating a restore and specifies the delivery url for the restore. See <a href="#">Automate Backup Restoration through the API</a> .

**Replica Sets** The replicaSets array is optional and defines each replica set's configuration.

```

"replicaSets" : [
    {
        "_id" : <string>,
        "version" : <integer>
        "members" : [
            {
                "_id" : <integer>,
                "host" : <string>
            },
            ...
        ],
        "force" : {
            "currentVersion" : <integer>
        }
    },
    ...
]

```

Name	Type	Description
replicaSets	array of objects	<i>Optional.</i> Objects that define the configuration of each <a href="#">replica set</a> . The Automation Agent uses the values in this array to create valid <a href="#">replica set configuration documents</a> . 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: <code>_id</code> ; <code>version</code> ; and <code>members</code> . For more information on the configuration documents, see <a href="#">repSetGetConfig</a> in the MongoDB manual.
replicaSets. <code>string</code>	<code>string</code>	The name of the replica set.
replicaSets. <code>integer</code>	<code>integer</code>	The version of the replica set configuration.
replicaSets. <code>array of objects</code>	<code>array of objects</code>	Objects that define each member of the replica set. The <code>members.host</code> field must specify the host's name as listed in <code>processes.name</code> . The Automation Agent expands the <code>host</code> field to create a valid replica set configuration. For more information on <code>members</code> objects, see <a href="#">repSetGetConfig</a> in the MongoDB manual.
replicaSets. <code>object</code>	<code>object</code>	<i>Optional.</i> An object that contains the <code>currentVersion</code> field set to a version number. Automation will force a reconfiguration of the replica set if and only if the value of <code>currentVersion</code> equals the current version of the replica set. You can use <code>force</code> to reconfigure a replica set that has lost members and can't reach a majority of votes.

**Sharded Clusters** The sharding array is optional and defines the configuration of each sharded cluster.

```

"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	<i>Optional.</i> Objects that define the configuration of each <i>sharded cluster</i> . 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.
sharding.name	string	The name of the cluster. This must correspond with the value in <code>processes.cluster</code> for a <code>mongos</code> .
sharding.configServer	array of servers	String values that provide the names of each <i>config server</i> 's hosts. The host names are the same names as are used in each host's <code>processes.name</code> field.
sharding.collections	array of objects	Objects that define the sharded <i>collections</i> and their <i>shard keys</i> .
sharding.collectionNames	string	The <i>namespace</i> of the sharded collection. The namespace is the combination of the database name and the name of the collection. For example, <code>testdb.testcoll</code> .
sharding.collectionKeys	array of arrays	The collection's <i>shard keys</i> . 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.
sharding.shards	array of objects	Objects that define the cluster's <i>shards</i> .
sharding.shardName	string	The name of the shard.
sharding.shardReplicaSet	string	The name of the shard's replica set, as specified in the <code>replicaSets._id</code> field.

**Cluster Balancer** The `balancer` object is optional and defines balancer settings for each cluster.

```

"balancer": {
    "<clusterName1>": <object>,
    "<clusterName2>": <object>,
    ...
}

```

Name	Type	Description
balanceObject	object	<i>Optional.</i> 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 <code>stopped</code> and <code>activeWindow</code> fields, as described in the procedure to schedule the balancing window <a href="#">in this tutorial</a> in the MongoDB manual.

**Authentication** The `auth` object is optional and defines authentication-related settings.

```

"auth" : {
    "autoUser": <string>,
    "autoPwd": <string>,
    "disabled": <Boolean>,
}

```

```
"deploymentAuthMechanisms": [ <string>, <string>, ... ],
"key" : <string>,
"keyfile" : <string>,
"usersDeleted" : [
    {
        "user" : <string>,
        "dbs" : [ <string>, ... ]
    }
],
"usersWanted" : [
    {
        "db" : <string>,
        "user" : <string>,
        "roles" : [ <string>, ... ],
        "pwd" : <32-character hex string>,
        "initPwd" : <string>,
        "userSource" : <string>,
        "otherDBRoles" : {
            <string> : [ <string>, ... ]
        }
    }
]
}
```

Name	Type	Description
auth	object	<i>Optional.</i> Defines authentication-related settings.
auth.autoUser	string	The username that the Automation agent uses when connecting to an instance.
auth.autoPwd	string	The password that the Automation agent uses when connecting to an instance.
auth.disabled	Boolean	Specifies whether authentication is enabled or disabled. Set to <code>true</code> to disable authentication, or <code>false</code> to enable authentication.
auth.deploymentMechanisms	array	List the supported authentication mechanisms for the processes in the deployment. Specify MONGODB-CR for MONGODB-CR / SCRAM-SHA-1 authentication, MONGODB-X509 for x.509 Client Certificate authentication, PLAIN for LDAP authentication, and GSSAPI for authenticating with Kerberos.
auth.disabled	boolean	<i>Optional.</i> Indicates if auth is disabled. If not specified, disabled defaults to <code>false</code> .
auth.key	string	The contents of the key file that Ops Manager uses to authenticate to the MongoDB processes. The <code>key</code> is not required if <code>disabled</code> is <code>true</code> .
auth.keyfile	string	The path and name of the key file that Ops Manager uses to authenticate to the MongoDB processes. The <code>keyfile</code> is not required if <code>disabled</code> is <code>true</code> .
auth.usersDeleted	array of objects	<i>Optional.</i> Objects that define the authenticated users to be deleted from specified databases or from all databases. This array must contain two fields: the <code>auth.usersDeleted.user</code> field and the <code>auth.usersDeleted.dbs</code> field.
auth.usersDeleted.name	string	The user's name.
auth.usersDeleted.dbs	array	String values that list the names of the databases from which the authenticated user is to be deleted.
auth.usersWanted	array of objects	<i>Optional.</i> Contains objects that define authenticated users to add to specified databases. Each object must have the <code>auth.usersWanted.db</code> , <code>auth.usersWanted.user</code> , and <code>auth.usersWanted.roles</code> fields, and then have exactly one of the following fields: <code>auth.usersWanted.pwd</code> , <code>auth.usersWanted.initPwd</code> , or <code>auth.usersWanted.userSource</code> .
auth.usersWanted.db	string	The database to which to add the user.
auth.usersWanted.user	string	The name of the user.
auth.usersWanted.roles	array	String values that list the <code>roles</code> to be assigned the user from the user's database, which is specified in <code>auth.usersWanted.db</code> .
auth.usersWanted.pwd	character hex string	The <b>MONGODB-CR</b> hash of the password assigned to the user. If you set this field, <b>do not</b> set the <code>auth.usersWanted.initPwd</code> or <code>auth.usersWanted.userSource</code> fields.
auth.usersWanted.initPwd	string	An initial cleartext password assigned to the user. If you set this field, <b>do not</b> set the <code>auth.usersWanted.pwd</code> or <code>auth.usersWanted.userSource</code> fields.
auth.usersWanted.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 <a href="#">Privilege Documents page in the MongoDB 2.4 manual</a> . If you set this field, <b>do not</b> set the <code>auth.usersWanted.pwd</code> or <code>auth.usersWanted.initPwd</code> fields.
auth.usersWanted.otherDB	object	<i>Optional.</i> If the <code>auth.usersWanted.db</code> field specifies <code>admin</code> 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** The `ssl` object is optional and enables SSL for encrypting connections.

```
"ssl" : {
    "CAFFilePath" : <string>
}
```

Name	Type	Description
<code>ssl</code>	object	<p><i>Optional.</i> Enables SSL for encrypting connections. To use SSL, be sure to choose a package that supports SSL.</p> <p>Starting in MongoDB 3.0, most MongoDB distributions now include support for SSL.</p> <p>All <a href="#">MongoDB Enterprise</a> supported platforms include SSL support.</p>

**MongoDB Roles** The `roles` array is optional and describes user-defined roles.

```
"roles" : [
    {
        "role" : <string>,
        "db" : <string>,
        "privileges" : [
            {
                "resource" : { ... },
                "actions" : [ <string>, ... ]
            },
            ...
        ],
        "roles" : [
            {
                "role" : <string>,
                "db" : <string>
            }
        ]
    },
    ...
]
```

Name	Type	Description
<code>roles</code>	array of objects	<p><i>Optional.</i> The <code>roles</code> 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 <code>:manual:‘ system roles collection &lt;/reference/system-roles-collection&gt;</code>, except for the <code>_id</code> field, which is not included here.</p>

**Kerberos** The `kerberos` object is optional and defines a kerberos service name used in authentication.

```
"kerberos": {
    "serviceName": <string>
}
```

Name	Type	Description
<code>kerberos</code>	object	<p><i>Optional.</i> A key-value pair that defines the kerberos service name agents use to authenticate via kerberos.</p>

**Indexes** The indexConfigs array is optional and defines indexes to be built for specific replica sets.

```
"indexConfigs" : [
  {
    "key" : [
      [ <string> : <val> ],
      ...
    ],
    "rsName" : <string>,
    "dbName" : <string>,
    "collectionName" : <string>
  },
  ...
]
```

Name	Type	Description
indexConfigs	array of objects	<i>Optional.</i> Objects that define specific indexes to be built for specific replica sets.
indexConfigs.key	array of arrays	The index's keys. This “array of arrays” contains a single array if the index has just one key.
indexConfigs.rsName	string	The replica set that the index is build on.
indexConfigs.dbName	string	The database the index applies to.
indexConfigs.collectionName	string	The collection the index applies to.

## Examples

### Get the Automation Configuration

#### Request

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/533c5895b9103...
```

#### Response

```
HTTP/1.1 200 OK
```

```
{
  <automation configuration>
}
```

### Update the Automation Configuration

**Request** The following replaces the automation configuration with the updated configuration saved in Users/admin/updated-conf.json. For an example of an updated configuration document, see [Example Automation Configuration](#).

```
curl -u "username:apiKey" -H "Content-Type: application/json" --digest -i "https://cloud.mongodb.com/...
```

#### Response

```
HTTP/1.1 200 OK
```

```
{}
```

**Example Automation Configuration** The following is an example entity passed through the --data @<configuration> option. The exact fields included in an automation configuration depend on the Ops Manager group.

The example uses ... when a field has multiple entries and also in place of some field values. For detailed information on a field, see the [Automation Configuration Entity](#) section on this page.

```
{  
    "options" : {  
        "downloadBase" : "/var/lib/mongodb-mms-automation",  
        "downloadBaseWindows" : "C:\\MMSAutomation\\versions"  
    },  
    "mongoDbVersions" : [  
        {  
            "name" : "3.2.0",  
            "builds" : [  
                {  
                    "platform" : "windows",  
                    "url" : "https://fastdl.mongodb.org/win32/mongodb-win32-x86_64-2008plus-3.2.0.z  
                    "gitVersion" : "45d947729a0315accb6d4f15a6b06be6d9c19fe7",  
                    "bits" : 64,  
                    "win2008plus" : true  
                },  
                ...  
            ],  
            ...  
        },  
        ...  
    ],  
    "agentVersion" : {  
        "directoryUrl" : "https://s3.amazonaws.com/mongodb-mms-build-agent/releases/prod/",  
        "name" : "2.6.4.1612-1"  
    },  
    "monitoringVersions" : [  
        {  
            "name" : "4.1.0.251-1",  
            "hostname" : "example.net",  
            "baseUrl" : null  
        },  
        ...  
    ],  
    "backupVersions" : [  
        {  
            "name" : "4.1.0.347-1",  
            "hostname" : "example.net",  
            "baseUrl" : null  
        },  
        ...  
    ],  
    "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" : {}
},
},
...
],
"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
        }
    ]
},
...
],
"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"
        }
    ]
},
"balancer" : { ... },
"auth" : {
    "authoritativeSet" : false,
    "disabled" : true
    "usersDeleted" : [ ],
    "usersWanted" : [ ]
},
"ssl" : { ... },
"roles" : [
    {
        "role" : ... ,
        "db" : ... ,
        "privileges" : [ ... ],
        "roles" : [ ... ]
    },
    ...
],
"kerberos" : {
    "serviceName" : ...
},
"indexConfigs" : [ ]
}

```

## Automation Status

### On this page

- [Overview](#)
- [Endpoint](#)
- [Sample Entity](#)
- [Entity Fields](#)
- [Example](#)

### 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.

## Endpoint

**Get Status** Retrieve the status of each MongoDB process in the deployment.

GET /api/public/v1.0/groups/GROUP-ID/automationStatus

## 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	number	The version of the most recently submitted <i>automation configuration</i> . If there is a <i>conflict in submissions of automation configurations</i> , this field lists the winning configuration.
processes	array	The group's deployed MongoDB instances.
processes.hostname	string	The fully qualified domain name (retrieved by issuing <code>hostname -f</code> ) of the server on which the MongoDB process and Automation Agent are hosted.
processes.name	string	The process name as specified in the automation configuration.
processes.lastGoalVersion	number	The <i>hash version</i> of the automation configuration with which this process had been deployed as configured. If the <code>processes.lastGoalVersionAchieved</code> number is not equal to the <code>goalVersion</code> number, the process has not yet deployed according to the current configuration.
processes.plan	array	Describes how a process that is not yet up-to-date with the configuration will achieve the goal state.

## Example

**Get Status**

```
curl -u "username:apiKey" --digest -i "https://cloud.mongodb.com/api/public/v1.0/groups/533c5895b9103"
```

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
}
```

## 11.3 Enable the Public API

### On this page

- [Overview](#)
- [Procedure](#)

### Overview

You enable the Public API on a per-group basis. To enable the Public API, you must have the *Owner* role for the group or have the *Global Owner* role Ops Manager.

Once you have enabled the Public API, users who want to use the API must generate their own API keys and optionally configure access to whitelisted operations. See [Configure Public API Access](#). A user's *roles* determine which *API resources* the user can access.

For a description of the Public API, see the [Public API Principles](#).

### Procedure

For each group that will use the Public API, do the following:

**Step 1: Select the *Settings* tab and then select *Group Settings*.**

**Step 2: Toggle the *Enable Public API* field to *ON*.**

An API Key is like a password. Keep it secret.

To generate a key, select the *Settings* tab and then *Public API Access*. 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.

## 11.4 Configure Public API Access

### On this page

- [Overview](#)
- [Procedures](#)

## Overview

To access the *Public API*, you generate an API key and optionally configure access for whitelisted operations.

### Public API Key

Your Public API key gives you access to the Public API. You have the same level of access through the API as you do through the Ops Manager interface.

You can have up to ten Public API keys associated with your account. Each key can be either enabled or disabled but all count toward the ten-key limit.

### Whitelist

Address-based whitelists protect certain API operations. Only client requests that originate from a whitelisted IP address are permitted to perform the operations. If an operation is whitelisted, it is marked as such on its *Public API Resources* page. To perform whitelisted operations, first create the whitelist of addresses for the operations, as described on this page.

### Access Control

Your Ops Manager *roles* determine which *API resources* you can use.

## Procedures

### Generate Public API Keys

---

**Important:** When you generate a key, Ops Manager displays it **one time only**. You must copy it. Ops Manager will never display the full key again.

---

A Public API key is like a password. Keep it secret.

**Step 1: Click *Settings*, then *Public API Access*.**

**Step 2: Generate a new Public 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 3: 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*.

### Provide Access to Whitelisted Operations

**Step 1: Click the *Settings* tab, and then click *Public API Access*.**

**Step 2: In the *Whitelist*, click *Add* and enter an address.** Enter an IP address or CIDR range. To add multiple entries in the whitelist, repeat this step.

You can enter any of the following:

Entry	Grants
An IP address	Access to whitelisted operations from that address.
A CIDR-notated range of IP addresses	Access to whitelisted operations from those addresses.
0.0.0.0/0	Unrestricted access to whitelisted operations.

If you leave the whitelist empty, you have no access to whitelisted operations.

#### Delete an Address from the Whitelist

**Step 1: Select the *Settings* tab and then select *Public API Access*.**

**Step 2: In the *Whitelist*, select the gear icon for the address and select *Delete*.**

## 11.5 Public API Tutorials

[Deploy a Sharded Cluster](#) Create a cluster using the Public API.

[Update the Automation Configuration](#) Implement automated restores through the Public API

[Upgrade a Deployment's MongoDB Version](#) Upgrade the version of MongoDB used by the deployment's instances.

[Automate Backup Restoration through the API](#) Implement automated restores through the Public API

### Deploy a Cluster through the API

#### On this page

- Overview
- Prerequisites
- Examples
- Procedures
- Next Steps

#### 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 sufficient access to Ops Manager. A user with the *global owner* or *group owner* role has sufficient access.

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. If you are deploying the sharded cluster on a fresh install of Ops Manager, you must register the first user.

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

**Generate an 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 into the Ops Manager web interface.**

**Step 2: Click *Settings*, then *Public API Access*.**

**Step 3: Generate a new Public 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 your user credentials, <public\_api\_key> with your *Public API key*, <app-example.net> with the Ops Manager URL, and <group\_name> with the name of the new group. The group name must be globally unique within the Ops Manager platform:

```
curl -u "<user@example.net>:<api_key>" -H "Content-Type: application/json" "http://<app-example.net>/api/public/v1.0/groups"
{
    "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 <http://docs.opsmanager.mongodb.com//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"
{
    "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: If you used a global owner user to create the group, you can remove that user from the group. (Optional)** The user you use to create the group is automatically added to the group. If you used a *global owner* user, you can remove the user 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"
```

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>?_method=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 *Settings* 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 through the API to Ops Manager.

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 your user credentials, `<public_api_key>` with the previously retrieved *Public 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>/au
```

Confirm that the `version` field of the retrieved *automation configuration* matches the `version` field in the `mms-cluster-config-backup.json` file, which is found on any server running the Automation Agent.

**Step 2: Create the top level of the new automation configuration.** 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",  
    "downloadBaseWindows": "C:\\MMSAutomation\\versions"  
  },  
  "mongoDbVersions": [],  
  "monitoringVersions": []  
}
```

```

        "backupVersions": [],
        "processes": [],
        "replicaSets": [],
        "sharding": []
    }
}

```

**Step 3: Add MongoDB versions to the automation configuration.** 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 automation configuration.** 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 automation configuration.** 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,
        "args": {
            "setParameter": "replSetConfigVersion=1"
        }
    }
]

```

```

    "processType": "mongod",
    "args2_4": {
        "port": 27017,
        "replSet": "rs1",
        "dbpath": "/data/",
        "logpath": "/data/mongodb.log"
    }
},
...
]

```

**Step 6: Add the sharded cluster topology to the automation configuration.** 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 updated automation configuration.** Use the `groups/<group_id>/automationConfig` endpoint to send the automation configuration document to Ops Manager. Replace `<configuration_document>` with the configuration document you created in the previous steps. Replace the credentials, Public API key, URL, and group ID as in previous steps.

```
curl -u "<user@example.net>:<api_key>" -H "Content-Type: application/json" "http://<app-example.net>/groups/<group_id>/automationConfig"
'
```

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 and confirm it contains the changes. Also confirm that the `version` field equals 2.

To retrieve the automation configuration, issue a command similar to the following. Replace the credentials, 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"
```

**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, Public 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"
```

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 Automation Configuration

### On this page

- Overview
- Prerequisites
- Procedure

### Overview

To modify your deployment through the Public API you update your group's *automation configuration*. The automation configuration determines the goal state of the group's databases and agents. The Automation Agent on each server works to achieve the goal state for the items running on its server.

To modify the automation configuration you retrieve the current configuration, make your changes, and then use PUT to replace the entire configuration. You must take care to change only those items you want modified and to leave the rest as is.

To update, you must use PUT. **Do not use PATCH.**

### Prerequisites

You must have access to the Public API. For more information, see [Configure Public API Access](#).

### Procedure

**Step 1: Retrieve the automation configuration from Ops Manager.** Use the *automationConfig* resource to retrieve the configuration. Issue the following command, replacing <username> and <apiKey> with your *API credentials*, <url> with the URL of Ops Manager, and <group\_id> with the group ID from your *Group Settings*:

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationConfig" --di
```

Confirm that the *version* field of the retrieved *automation configuration* matches the *version* field in the *mms-cluster-config-backup.json* file, which is found on any server running the Automation Agent.

**Step 2: Edit the automation configuration.** Refer to *Automation Configuration* for detailed descriptions of settings.

**Step 3: Send the updated automation configuration.** Use the *automationConfig* resource to send the updated automation configuration.

Issue the following command, replacing <configuration> with path to the updated configuration document. Replace the username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" -H "Content-Type: application/json" "http://<url>/api/public/v1.0/group
```

Upon successful update of the configuration, the API returns the HTTP 200 OK status code to indicate the request has succeeded.

**Step 4: Confirm successful update of the automation configuration.** Retrieve the automation configuration from Ops Manager and confirm it contains the changes. To retrieve the configuration, issue the following command, replacing username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationConfig" --di
```

**Step 5: Check the deployment status to ensure goal state is reached.** Use the *automationStatus* resource to retrieve the deployment status. Issue the following command, replacing username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationStatus" --di
```

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*.

## Update the MongoDB Version of a Deployment

### On this page

- [Overview](#)
- [Consideration](#)
- [Prerequisite](#)
- [Procedure](#)

### 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 `<username>` and `<apiKey>` with your [API credentials](#), `<url>` with the URL of Ops Manager, and `<group_id>` with the group ID from your [Group Settings](#):

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationConfig" --di
```

Confirm that the `version` field of the retrieved `automation configuration` matches the `version` field in the `mms-cluster-config-backup.json` file, which is found on any server running the Automation Agent.

**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 updated automation configuration.** Use the `automationConfig` resource to send the updated automation configuration.

Issue the following command, replacing `<configuration>` with path to the updated configuration document. Replace the username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" -H "Content-Type: application/json" "http://<url>/api/public/v1.0/group/<groupID>/configuration/<configuration>"
```

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 and confirm it contains the changes. To retrieve the configuration, issue the following command, replacing username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationConfig" --di
```

**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 username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationStatus" --di
```

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](#).

## Automate Backup Restoration through the API

### On this page

- [Overview](#)
- [Prerequisites](#)
- [Procedure](#)

### Overview

Automating backup restorations involve manipulation of the *automation configuration*. You first request a restore link through the `restoreJobs` resource and then add the restore link to the automation configuration using the `PUT` method.

You must add the restore link to each `mongod` instance in the automation configuration's `processes` array. For example, for a sharded cluster you add the restore link to each member process, including each config server.

To automate backup restoration, you must use `PUT`. **Do not use** `PATCH`.

### Prerequisites

You must have access to the Public API. For more information, see [Configure Public API Access](#).

### Procedure

**Step 1: Retrieve the snapshot ID.** Use the *snapshots* resource to get the snapshot ID. Issue the following command, replacing `<username>` and `<apiKey>` with your [API credentials](#), `<url>` with the URL of Ops Manager, `<group_id>` with the group ID from your [Group Settings](#), and `<cluster_id>` with the cluster ID.

```
curl -i -u "username:apiKey" --digest "http://<url>/api/public/v1.0/groups/<group_id>/clusters/<clust
```

**Step 2: Create a restore job for the snapshot.** Use the *restoreJobs* resource to request an HTTP restore of the snapshot, which will provide a link to a downloadable tar.gz file. Issue the following command, replacing the username, API key, URL, group ID, and cluster ID as in the previous step. Replace <snapshot\_id> with the snapshot ID you retrieved.

```
curl -i -u "username:apiKey" -H "Content-Type: application/json" --digest -X POST "http://<url>/api/public/v1.0/groups/<group_id>/clusters/<cluster_id>/automationConfig" --data-binary '{ "snapshotId": "<snapshot_id>" }'
```

For an example, see [Create Restore Jobs](#).

**Step 3: Retrieve the restore link.** Use the *restoreJobs* resource to retrieve the restore link. Issue the following command, replacing the username, API key, URL, group ID, and cluster ID as in the previous step.

```
curl -i -u "username:apiKey" --digest "http://<url>/api/public/v1.0/groups/<group_id>/clusters/<cluster_id>/automationConfig"
```

Copy the restore link from the delivery.url field.

**Step 4: Retrieve the automation configuration.** Use the *automationConfig* resource to retrieve the configuration. Issue the following command, replacing <username> and <apiKey> with your *API credentials*, <url> with the URL of Ops Manager, and <group\_id> with the group ID from your *Group Settings*:

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationConfig" --digest
```

Confirm that the version field of the retrieved *automation configuration* matches the version field in the *mms-cluster-config-backup.json* file, which is found on any server running the Automation Agent.

**Step 5: Add the restore link to the automation configuration.** In the processes array, add the backupRestoreUrl field to each mongod to be restored. Enter the restore link as the field's value.

For example:

```
"processes" : [ { ... , "hostname" : "example.mongodbdns.com", "backupRestoreUrl" : "https://api-backup.mongodb.com/backup/restore/v2/pull/abc123abc123/def456def456" }, ... ]
```

**Step 6: Send the updated automation configuration.** Use the *automationConfig* resource to send the updated automation configuration.

Issue the following command, replacing <configuration> with path to the updated configuration document. Replace the username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" -H "Content-Type: application/json" "http://<url>/api/public/v1.0/groups/<group_id>/automationConfig" --data-binary { "processes" : [ { ... , "hostname" : "example.mongodbdns.com", "backupRestoreUrl" : "https://api-backup.mongodb.com/backup/restore/v2/pull/abc123abc123/def456def456" }, ... ] }
```

Upon successful update of the configuration, the API returns the HTTP 200 OK status code to indicate the request has succeeded.

**Step 7: Confirm successful update of the automation configuration.** Retrieve the automation configuration from Ops Manager and confirm it contains the changes. To retrieve the configuration, issue the following command, replacing username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationConfig" --di
```

**Step 8: Check the deployment status to ensure goal state is reached.** Use the *automationStatus* resource to retrieve the deployment status. Issue the following command, replacing username, API key, URL, and group ID as in previous steps.

```
curl -u "<username>:<apiKey>" "http://<url>/api/public/v1.0/groups/<group_id>/automationStatus" --di
```

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*.

**Step 9: Remove `processes.backupRestoreUrl`.** When Goal State is achieved, remove the `processes.backupRestoreUrl` field and resend the automation configuration.

## 12 Troubleshooting

### On this page

- [Getting Started Checklist](#)
- [Installation](#)
- [Monitoring](#)
- [Authentication](#)
- [Backup](#)
- [System](#)
- [Automation Checklist](#)

This document provides advice for troubleshooting problems with Ops Manager.

### 12.1 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 is 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 *Settings* 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 Ops Manager](#) 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. `api-agents.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.

## 12.2 Installation

### The monitoring server does not start up 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
```

## 12.3 Monitoring

### Alerts

For information on creating and managing alerts, see [Manage Alert Configurations](#) 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 a Process from Management or Monitoring](#).
- Disable or delete the alert configuration. See [Manage Alert Configurations](#).
- Turn off alerts for a specific host. See [Manage a Process's Alerts](#).

#### 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 the 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 the 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.

## Receive “Oplog Behind” alert

The [Oplog Behind](#) alert occurs when the Ops Manager backup service has not received an oplog from the Backup Agent for at least an hour. This can be caused by the following conditions:

- Backup Agent is down.
- Backup Agent is overloaded.
- Backup Agent can't reach the replica set.
- Backup Agent can't reach or successfully send data to Ops Manager.
- Backup Agent is continuously restarting.

Check the Backup Agent [logs](#) in Ops Manager and on the Backup Agent host to find out which of these occurred.

## 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 Changes](#) and then [Confirm & Deploy](#).

If you shut down the deployment and still cannot find a solution, [remove the deployment](#) from Ops Manager.

## 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 [Enable SSL for a Deployment](#) for more information.
- If your database is running with authentication. You must supply Ops Manager with the authentication credentials for the host. See [Configure MongoDB Authentication and Authorization](#).

## Hosts

### Hosts are not Visible

Problems with the Monitoring Agent detecting hosts can be caused by a few factors.

**Host not added:** 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 *Settings* 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](#) 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 Support](#) for help in deleting these hosts.

## Groups

### 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 [Create a Group](#).

## 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.

## **12.4 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 [Settings](#) 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 [Support](#) to rule out SMS message delivery delays.

## 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

### Forgot to Change MONGODB-CR Error

If your MongoDB deployment uses LDAP for authentication, and you find the following error message:

You forgot 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](#).

## 12.5 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](#).

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
```

## Insufficient Oplog Size Error

When using the Ops Manager interface to back up a MongoDB cluster, Ops Manager checks to see if the cluster's oplogs are, based on their recent usage, large enough to hold a minimum of 3 hours worth of data based on the last 24 hours of usage patterns. If the oplogs are not large enough, it may be possible that the oplogs may have turned over multiple times, creating a gap between where a backup ends and an oplog starts.

If the oplog size check fails, the user cannot enable backups and is shown the warning:

```
Insufficient oplog size: The oplog window must be at least 3 hours
over the last 24 hours for all members of replica set <name>.
Please increase the oplog.
```

If possible, wait to start a backup until the oplog has had sufficient time to meet the size requirement.

If this is not possible, the minimum oplog size value can be changed. See [mms.backup.minimumOplogWindowHours](#) for how to set this value.

**Warning:** Do not change the minimum oplog size unless you are certain smaller backups still provide useful backups.

---

**Important:** MongoDB recommends only changing this value temporarily to permit a test backup job to execute. The minimum oplog size value should be reset to the default as soon as possible. If an oplog is set to too small of a value, it can result in a gap between a backup job and an oplog which makes the backup unusable for restores. Stale backup jobs must be resynchronized before it can be used for restores.

Understanding the risks given, you can start backups using this changed minimum value. Once you pass the 24 hour mark, you should reset this minimum value to preserve the sanity check for the global Ops Manager installation going forward.

## 12.6 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 Database, see [Ops Manager Hardware and Software Requirements](#).

## Obsolete Config Settings

Ops Manager will fail to start if there are obsolete configuration settings set in the `conf-mms.properties` file. If there is an obsolete setting, the log lists an “Obsolete Setting” error as in the following:

```
[OBSOLETE SETTING] Remove "mms.multiFactorAuth.require" or replace "mms.multiFactorAuth.require" with
```

You will need to remove or replace the obsolete property in the `conf-mms.properties` file before you can start Ops Manager.

## 12.7 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.

### Automation Runs Only on 64-bit Architectures

Ops Manager provides only 64-bit downloads of the Automation Agent.

### Using Own Hardware

- If you deploy Automation manually, ensure that you have one Automation Agent on every server.
- If you deploy the agent manually, you must create MongoDB’s dbpath and the directory for the MongoDB binaries and ensure that the user running the agent owns these directories.

If you install using the `rpm` package, the agent runs as the `mongod` user; if using the `deb` package, the agent runs as the `mongodb` user. If you install using the `tar.gz` archive file, you can run the agent as any user.

## 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 `cloud.mongodb.com` on port 443 (i.e. `https`). For more information on access to ports and IP addresses, see [Security Overview](#).

## 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.

# 13 Frequently Asked Questions

## On this page

- [What versions of MongoDB does Ops Manager manage?](#)
- [Automation FAQs](#)
- [Monitoring FAQs](#)
- [Backup FAQs](#)
- [Administration FAQs](#)
- [Miscellaneous](#)

This document addresses common questions about Ops Manager and its use.

### 13.1 What versions of MongoDB does Ops Manager manage?

For specific Ops Manager functions and supported MongoDB versions, see [MongoDB Compatibility](#).

### 13.2 Automation FAQs

Ops Manager can automate management operations for your monitored MongoDB processes, allowing you to reconfigure, stop, and restart MongoDB through the Ops Manager interface.

Ops Manager Automation can run only on 64-bit architectures.

## How does Ops Manager manage MongoDB deployments?

After you deploy the Automation Agent in the environment of the MongoDB deployment, each agent periodically communicates with Ops Manager and performs any required work.

Agents constantly reassess their environment to adapt their work as necessary. If an agent encounters an issue, such as network connectivity problems or Ops Manager failure, the agents adjust their work to compensate and safely arrive at their goal state.

Agents create plans to move from their current state to a goal state. Plans execute in steps, where each step is autonomous and independent of other steps.

For example, for an installation, the plan involves downloading MongoDB, starting the process with the appropriate command line options, initializing the replica set, waiting for a healthy majority. The configuration reaches goal state when the replica set is active and has a healthy majority.

## How many Automation Agents do I need?

To use Automation, you must have an agent running on every host where a managed MongoDB instance runs.

## Is any MongoDB data transferred by the Automation Agent?

Agents do *not* transmit any data records from a MongoDB deployment. The agents only communicate deployment configuration information and MongoDB logs.

## Will Ops Manager handle failures during an upgrade, such as Ops Manager going down or a network partition?

Generally speaking, yes. The design of the management and automation components of Ops Manager do not account for *all* possible failures; however the architecture of the system can work around many types of failures.

## What types of deployment can I create in Ops Manager?

You can configure all MongoDB deployment types: sharded clusters, replica sets, and standalones.

The shards in a sharded cluster **must** be replica sets. That is, a shard cannot be a standalone mongod. If you must run a shard as a single mongod (which provides **no** redundancy or failover), run the shard as a single-member replica set.

## 13.3 Monitoring FAQs

### Host Configuration

#### How do I add a new host or server?

See [Add Existing MongoDB Processes to Ops Manager](#).

## **Can Ops Manager monitor itself?**

Yes. You can use Ops Manager to monitor the *Backing MongoDB instances* that hold data for Ops Manager and the Backup Daemon. Create a separate group for the backing instances.

You **cannot**, however, use Ops Manager to backup or automate the backing instances.

Ops Manager also provides built-in basic monitoring of all backing databases through *System alerts*.

## **Can I monitor Kerberos-enabled instances?**

Yes. Ops Manager 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, Ops Manager 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.

<sup>1</sup> 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 *Settings* 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. In your Ops Manager group, a single Monitoring Agent connects to all MongoDB databases. Configure firewalls to allow the Monitoring Agent to connect across data centers and servers. 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. If one Monitoring Agent fails, another starts monitoring. As long as a Monitoring Agent is running, Ops Manager will not trigger a *Monitoring Agent Down* alert. See the *Monitoring Agent Redundancy* section.

### **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 is 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 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.

---

<sup>1</sup> DB Stats will not appear until 30 minutes after you add the host to Ops Manager.

## **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 try to connect using the mongo shell:

```
mongo hostname:port/test
```

## **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. 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.

To create a preferred hostname, go to *Group Settings* and add a *Preferred Hostnames* entry. For details, see *Edit a Group's Configuration*

## **How do I download the Monitoring Agent?**

You can update the Monitoring Agent from the *Agents* page of the *Settings* 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?**

See *Remove Monitoring Agents from Ops Manager*.

## **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 of the *Settings* 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 64 days.
- Snapshots according to their *retention policy*.

## **13.4 Backup FAQs**

Ops Manager creates backups of MongoDB replica sets and sharded clusters. After an initial sync to MongoDB's data centers, Ops Manager 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.

### **Requirements**

#### **What version of MongoDB does the Backup feature require?**

For information on compatibility, see [MongoDB Compatibility](#).

#### **What MongoDB permissions does the Backup Agent require?**

If you are backing up a MongoDB instance that has authentication enabled, the Backup Agent requires elevated privileges, as described in [Required Access for Backup Agent](#).

#### **See also:**

[User Privilege Roles in MongoDB](#).

#### **Are there any limits to the types of deployments the Backup feature supports?**

Yes. Backup does not currently support standalone deployments. Backup has full support for replica sets and sharded clusters.

## Why doesn't the Backup feature support standalone deployments?

After an initial sync of your data, Ops Manager copies data from the *oplog* to provide a continuous backup with point-in-time recovery. Ops Manager does not support backup of 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 *Settings* tab.

### Why is my Backup Agent highlighted in red?

If your agent is highlighted in red on the *Agents* page of the *Settings* tab, your agent is out of date. For instructions on updating the agent, see [Install Backup Agent](#).

## Operations

### How does the Backup Feature 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 to 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/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 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 backups and have no effect on the source deployment. Each token is less than 100 bytes. See: [Checkpoints](#) for more information about checkpoints.

## **Will Backup impact my production databases?**

The Backup feature 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 backups, 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. Ops Manager does not currently support back up of 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 Ops Manager [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*.
- Rolling index builds.
- Using `compact` or `repairDatabase` to reclaim a significant amount of space.

Resync is not strictly necessary after `compact` or `repairDatabase` operations but will ensure that the Ops Manager copy of the data is resized, which means quicker restores.

#### See also:

[Maintenance Operations for Replica Set Members](#).

## Configuration

### How can I prevent Ops Manager from backing up a collection?

Ops Manager provides a *namespaces filter* that allows you to specify which collections or databases to back up.

### How can I change which namespaces are backed up?

To edit the filter, see [Edit a Backup's Settings](#). Changing the *namespaces filter* might necessitate a resync. If so, Ops Manager handles the resync.

### 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 because no daemon has space for a 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 produces a copy of your data files that you can use to seed a new deployment.

### **How does Ops Manager provide point-in-time restores?**

Ops Manager first creates a local restore of a snapshot preceding the point in time and then applies the stored oplog entries to reach the specified point in time.

The ability of Ops Manager to provide a given point-in-time restore depends on the retention policy of the snapshots and the configured point-in-time window.

For information on retention policy and the point-in-time window, see [Edit Snapshot Schedule and Retention Policy](#).

### **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. You can change the schedule through the *Edit Snapshot Schedule* menu option for a backed-up deployment. Administrators can change the *snapshot frequency and retention policy* through the *snapshotSchedule resource* in the API.

### **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 Ops Manager must apply to the preceding snapshot to roll forward to the requested point-in-time of the backup.

### **Does the Backup feature 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, Ops Manager 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 Ops Manager to deliver the restore. All restores require 2-factor authentication. If you have SMS set up, Ops Manager will send an authorization code via SMS. 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).

---

Ops Manager delivers restores as `tar.gz` archives of MongoDB data files.

For more information, see [Restore MongoDB Deployments](#).

### 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 public key, see [Generate a Key Pair for SCP Restores](#).

### How does Ops Manager handle a rollback of backed-up data?

If your MongoDB deployment experiences a [rollback](#), then Ops Manager also rolls back the backed-up data.

Ops Manager detects the rollback when a [\*tailing cursor\*](#) finds a mismatch in timestamps or hashes of write operations. Ops Manager 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 Ops Manager 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. Ops Manager then resumes normal backup 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 your backups\*](#).

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 Ops Manager 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 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 Ops Manager cannot find a common point in the oplog.
- If an oplog event tries to update a document that does not exist in the backup of the replica set, as might happen if syncing from a secondary that has inconsistent data with respect to the primary.

## 13.5 Administration FAQs

### Can I reset my password?

Yes, in the login screen, use the link to reset your password.

To change your password, log in with your current password and [access your user account](#).

### How do I add a user to my company/group?

For instructions on adding users to your company/group, see [Manage Ops Manager Users](#).

### 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.

### What do the alert conditions mean?

For a reference on the alert conditions, see [Alert Conditions](#).

### What alerts are configured by default?

See [Manage Alert Configurations](#) for the default alert configurations as well as steps to add new alerts or modify existing alerts, including modifying the alert frequency.

## 13.6 Miscellaneous

### 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](#)

## 14 Reference

[Ops Manager Configuration](#) Describes the Ops Manager configuration options.

[Automation Agent](#) Install and configure the automation agent.

[Monitoring Agent](#) Install and configure the Monitoring Agent.

**Backup Agent** Install and configure the Backup Agent.

**Database Commands Used by Monitoring Agent** A reference sheet for the monitoring service.

**Audit Events** An inventory of all audit events reported in the activity feed.

**MongoDB Compatibility** MongoDB compatibility with Ops Manager features.

**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 configuration file used to determine the desired state of the MongoDB deployment.

**Supported MongoDB Options for Automation** Supported options for a MongoDB process as specified in the automation configuration file.

**SNMP Traps and Ops Manager Severities** Explanation of how severe Ops Manager alerts are in SNMP Traps.

**Glossary** A glossary of common terms and concepts in Ops Manager.

## 14.1 Ops Manager Configuration

### On this page

- [Overview](#)
- [Web Server Settings](#)
- [Email Settings](#)
- [User Authentication Method](#)
- [Authentication through Ops Manager Application Database](#)
- [Authentication through LDAP](#)
- [Multi-Factor Authentication \(MFA\) Settings](#)
- [Other Authentication Options](#)
- [HTTP/HTTPS Proxy Settings](#)
- [Twilio Integration Settings](#)
- [MongoDB Version Management](#)
- [Backup Snapshots](#)
- [Public API](#)
- [Monitoring Agent Session Failover](#)
- [SNMP Heartbeat Settings](#)
- [Backup Settings](#)
- [Backup Daemon](#)
- [Ops Manager Application Database Connection String](#)
- [SSL Connection to the Application Database](#)
- [Kerberos Authentication to the Application Database](#)
- [Encrypt User Credentials](#)

### Overview

Ops Manager stores configuration settings both globally in the Ops Manager Application Database and locally on each server. Global settings apply to all your Ops Manager servers. Local settings apply to the server on which they are configured. Any local settings on a server override the global settings.

You configure global settings through the Ops Manager interface during installation. You can edit global settings at any time through the *Admin* interface by clicking the *General* tab and then clicking *Ops Manager Config*.

You configure local settings through a server's `conf-mms.properties` file. Each server's `conf-mms.properties` must contain the connection string and authentication settings for accessing the Ops Manager Application Database. The `conf-mms.properties` file also contains any overrides of global settings specific to that server.

The location of the `conf-mms.properties` file depends on how you installed Ops Manager, as described in the table below.

Install method	<code>conf-mms.properties</code> location
rpm or deb package	<code>/opt/mongodb/mms/conf/</code>
<code>tar.gz</code> archive	<code>&lt;install-directory&gt;/conf/</code>
msi file	<code>&lt;install-folder&gt;\Server\Config</code>
(Windows)	By default, this is: <code>C:\MMSData\Server\Config</code> .

## Web Server Settings

Configure global settings through the *Admin* interface. Ops Manager stores global settings in the Ops Manager Application database.

### URL to Access Ops Manager

Type: string

The fully qualified URL and port number of the Ops Manager Application. For example:

`http://mms.example.com:8080`

To use a port other than 8080, see [Manage Ops Manager Ports](#).

Corresponds to configuration file setting: `mms.centralUrl`

### HTTPS PEM Key File

Type: string

Absolute path to the PEM file that contains the Ops Manager Application's valid certificate and private key. The PEM file is required if the Ops Manager Application will 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, you must also change the port specified in the [URL to Access Ops Manager](#) setting.

Corresponds to configuration file setting: `mms.https.PEMKeyFile`

### HTTPS PEM Key File Password

Type: string

The password for the HTTPS PEM key file. This is required if the PEM file contains an encrypted private key. If storing this in the `conf-mms.properties` file, you can encrypt the password using the Ops Manager `credentialstool`. See [Encrypt User Credentials](#).

Corresponds to configuration file setting: `mms.https.PEMKeyFilePassword`

### Load Balancer Remote IP Header

Type: string

If you use a load balancer with the Ops Manager Application, set this to the HTTP header field the load balancer uses to identify the originating client's IP address to the application server. When you specify [Load](#)

`Balancer Remote IP Header`, do not allow clients to connect directly to any application server. A load balancer placed in front of the Ops Manager Application servers must not return cached content.

See [Configure a Highly Available Ops Manager Application](#) for more information.

Corresponds to configuration file setting: `mms.remoteIp.header`

## Email Settings

The following email address settings are mandatory. You **must** define them before Ops Manager will start.

### From Email Address

*Type:* string

The email address used for sending the general emails, such as Ops Manager alerts. You can include an alias with the email address.

Ops Manager Alerts <`mms-alerts@example.com`>

Corresponds to configuration file setting: `mms.fromEmailAddr`

### Reply To Email Address

*Type:* string

The email address from which to send replies to general emails.

Corresponds to configuration file setting: `mms.replyToEmailAddr`

### Admin Email Address

*Type:* string

The email address of the Ops Manager admin. This address receives emails related to problems with Ops Manager.

Corresponds to configuration file setting: `mms.adminEmailAddr`

### Email Delivery Method Configuration

*Type:* string

The email interface to use.

If you use the AWS Simple Email Service, set this to `com.xgen.svc.core.dao.email.AwsEmailDao`. See also the [AWS Access Key](#) and [AWS Secret Key](#) settings.

For JavaEmailDao, set this to `com.xgen.svc.core.dao.email.JavaEmailDao`.

Corresponds to configuration file setting: `mms.emailDaoClass`

### Transport

*Type:* string

*Default:* smtp

The transfer protocol, either smtp or smtps, as specified by your email provider.

Corresponds to configuration file setting: `mms.mail.transport`

### SMTP Server Hostname

*Type:* string

*Default:* localhost

Email hostname as specified by your email provider.

`mail.example.com`

Corresponds to configuration file setting: `mms.mail.hostname`

**SMTP Server Port**

*Type:* number

*Default:* 25

Port number for the transfer protocol as specified by your email provider.

Corresponds to configuration file setting: `mms.mail.port`

**Username**

*Type:* string

User name of the email account. If unset, defaults to disabled SMTP authentication.

Corresponds to configuration file setting: `mms.mail.username`

**Password**

*Type:* string

Password for the email account. If unset, defaults to disabled SMTP authentication.

Corresponds to configuration file setting: `mms.mail.password`

**Use SSL**

*Type:* boolean

*Default:* false

Set this to `true` if the transfer protocol runs on top of TLS.

Corresponds to configuration file setting: `mms.mail.tls`

**AWS Endpoint**

*Type:* string

Required if using the Amazon Simple Email Service. Sets the endpoint URL for the Amazon Simple Email Service.

Corresponds to configuration file setting: `aws.ses.endpoint`

**AWS Access Key**

*Type:* string

The access key ID for AWS. Required if using the Amazon Simple Email Service.

Corresponds to configuration file setting: `aws.accesskey`

**AWS Secret Key**

*Type:* string

The secret access key for AWS. Required if using the Amazon Simple Email Service.

Corresponds to configuration file setting: `aws.secretkey`

**User Authentication Method**

**User Authentication Method**

*Type:* string

Select whether to store authentication credentials in the Ops Manager Application Database or in an external authentication source.

Corresponds to configuration file setting: `mms.userSvcClass`

## Authentication through Ops Manager Application Database

### **Password Changes Before Reuse**

Type: number

The number of previous passwords to remember. You cannot reuse a remembered password as a new password.

Corresponds to configuration file setting: `mms.password.minChangesBeforeReuse`

### **Failed Login Attempts Before Account Lock**

Configuration file setting: `mms.password.maxFailedAttemptsBeforeAccountLock`

Type: number

The number of failed login attempts before an account becomes locked. Only an Ops Manager Administrator can unlock a locked account.

Corresponds to configuration file setting: `mms.password.maxFailedAttemptsBeforeAccountLock`

### **Days Inactive Before Account Lock**

Configuration file setting: `mms.password.maxDaysInactiveBeforeAccountLock`

Type: number

The maximum number of days with no visits to the Ops Manager website before Ops Manager locks an account.

Corresponds to configuration file setting: `mms.password.maxDaysInactiveBeforeAccountLock`

### **Days Before Password Change Required**

Type: number

The number of days a password is valid before the password expires.

Corresponds to configuration file setting: `mms.password.maxDaysBeforeChangeRequired`

## Authentication through LDAP

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*.

Settings that begin with “`mms.ldap.global.role`” 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 [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*.

### **LDAP URI**

Type: string

The URI for the LDAP or SSL LDAP server.

`mms.ldap.url=ldaps://acme-dc1.acme.example.com:3890`

Corresponds to configuration file setting: `mms.ldap.url`

### **LDAP SSL CA File**

Type: string

A file containing one or more trusted certificates in PEM format. Use this setting if you are using LDAPS and the server is using a certificate that is not from a well-known CA.

`mms.ldap.ssl.CAFile=/opt/CA.pem`

Corresponds to configuration file setting: `mms.ldap.ssl.CAFile`

#### **LDAP SSL PEM Key File**

*Type:* string

A file containing a client certificate and private key. Use this setting when your SSL LDAP server requires client certificates.

`mms.ldap.ssl.PEMKeyFile=/opt/keyFile.pem`

Corresponds to configuration file setting: `mms.ldap.ssl.PEMKeyFile`

#### **LDAP SSL PEM Key File Password**

*Type:* string

The password for `LDAP SSL PEM Key File`. Use this setting if the `PEMKeyFile` is encrypted.

`mms.ldap.ssl.PEMKeyFilePassword=<password>`

Corresponds to configuration file setting: `mms.ldap.ssl.PEMKeyFilePassword`

#### **LDAP Bind Dn**

*Type:* string

The LDAP user used to execute searches for other users.

`mms.ldap.bindDn=authUser@acme.example.com`

Corresponds to configuration file setting: `mms.ldap.bindDn`

#### **LDAP Bind Password**

*Type:* string

The password for the search user.

`mms.ldap.ssl.bindPassword=<password>`

Corresponds to configuration file setting: `mms.ldap.bindPassword`

#### **LDAP User Base Dn**

*Type:* string

The base Distinguished Name (DN) that Ops Manager uses to search for users. Escape the = sign with \.

`mms.ldap.user.baseDn=DC\=acme,DC\=example,DC\=com`

Corresponds to configuration file setting: `mms.ldap.user.baseDn`

#### **LDAP User Search Attribute**

*Type:* string

The LDAP field used for the LDAP search. This is typically a username or an email address. The value of this field is also used as the Ops Manager username.

`mms.ldap.user.searchAttribute=<myAccountName>`

Corresponds to configuration file setting: `mms.ldap.user.searchAttribute`

**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.

```
mms.ldap.user.group=memberOf
```

Corresponds to configuration file setting: mms.ldap.user.group

**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 **LDAP User Group** setting.

```
mms.ldap.global.role.owner=CN\=MMSGlobalOwner,OU\=MMS,OU\=acme Groups,DC\=acme,DC\=example,DC\=com
```

Corresponds to configuration file setting: mms.ldap.global.role.owner

**LDAP User First Name**

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.

```
mms.ldap.user.firstName="First"
```

Corresponds to configuration file setting: mms.ldap.user.firstName

**LDAP User Last Name**

Type: string

The LDAP user attribute that contains the user's last name such as sn. After successful LDAP authentication, Ops Manager synchronizes the specified LDAP attribute with the last name from the Ops Manager user record.

```
mms.ldap.user.lastName="Last"
```

Corresponds to configuration file setting: mms.ldap.user.lastName

**LDAP User Email**

Type: string

The LDAP user attribute that contains the user's email address such as mail. After successful LDAP authentication, Ops Manager synchronizes the specified LDAP attribute with the email address from the Ops Manager user record.

```
mms.ldap.user.email="user@example.com"
```

Corresponds to configuration file setting: mms.ldap.user.email

**LDAP Global Role Automation Admin**

Type: string

The LDAP group whose members have the *global automation admin role* in Ops Manager. Specify groups using the format used by the LDAP attribute specified in the **LDAP User Group** setting. You can specify multiple groups using the ; ; delimiter. To change the default delimiter, use the `mms.ldap.group.separator` setting.

```
mms.ldap.global.role.automationAdmin=CN\=MMS-AutomationAdmin,OU\=MMS,OU\=acme Groups,DC\=acme,DC\=example,DC\=com
```

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*.

Corresponds to configuration file setting: mms.ldap.global.role.automationAdmin

#### **LDAP Global Role Backup Admin**

Type: string

The LDAP group whose members have the *global backup admin role* in Ops Manager.

```
mms.ldap.global.role.backupAdmin=CN\=MMS-BackupAdmin,OU\=MMS,OU\=acme Groups,DC\=acme,DC\=example,DC\=com
```

Corresponds to configuration file setting: mms.ldap.global.role.backupAdmin

#### **LDAP Global Role Monitoring Admin**

Type: string

The LDAP group whose members have the *global monitoring admin role* in Ops Manager.

```
mms.ldap.global.role.monitoringAdmin=CN\=MMS-MonitoringAdmin,OU\=MMS,OU\=acme Groups,DC\=acme,DC\=example,DC\=com
```

Corresponds to configuration file setting: mms.ldap.global.role.monitoringAdmin

#### **LDAP Global Role User Admin**

Type: string

The LDAP group whose members have the *global user admin role* in Ops Manager.

```
mms.ldap.global.role.userAdmin=CN\=MMS-UserAdmin,OU\=MMS,OU\=acme Groups,DC\=acme,DC\=example,DC\=com
```

Corresponds to configuration file setting: mms.ldap.global.role.userAdmin

#### **LDAP Global Role Read Only**

Type: string

The LDAP group whose members have the *global read-only role* in Ops Manager.

```
mms.ldap.global.role.readOnly=CN\=MMS-ReadOnly,OU\=MMS,OU\=acme Groups,DC\=acme,DC\=example,DC\=com
```

Corresponds to configuration file setting: mms.ldap.global.role.readOnly

#### **mms.ldap.group.separator**

To set this, click *Config* and then click the *Custom* tab.

Type: string

Each of the global role values takes a delimited list of groups:

"dbas,sysadmins"

If a group value contains the delimiter, the delimiter must be set to another value.

---

#### **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 ; ; .

## Multi-Factor Authentication (MFA) Settings

### Multi-factor Auth Level

Type: string

Default: OFF

Configures the *two-factor authentication* “level”:

- OFF: Disables two-factor authentication. Ops Manager does not use two-factor authentication.
- OPTIONAL: Users can choose to set up two-factor authentication for their Ops Manager account.
- REQUIRED\_FOR\_GLOBAL\_ROLES: Users who possess a *global role* **must** set up two-factor authentication. Two factor authentication is optional for all other users.
- REQUIRED: **All** users must set up two-factor authentication for their Ops Manager account.

Two-factor authentication is recommended for the security of your Ops Manager deployment.

Corresponds to configuration file setting: `mms.multiFactorAuth.level`

### `mms.multiFactorAuth.require`

In Ops Manager 1.8 and later, `mms.multiFactorAuth.level` replaces `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.

### Multi-factor Auth Allow Reset

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 each Ops Manager group the user belongs to.

Corresponds to configuration file setting: `mms.multiFactorAuth.allowReset`

### Multi-factor Auth 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.

Corresponds to configuration file setting: `mms.multiFactorAuth.issuer`

## Other Authentication Options

### ReCaptcha Enabled

Type: boolean

Set to true to require reCaptcha validation when a new user registers. You must have a reCaptcha account.

Corresponds to configuration file setting: reCaptcha.enabled

### ReCaptcha Public Key

Type: string

The reCaptcha public key associated with your account.

Corresponds to configuration file setting: reCaptcha.public.key

### ReCaptcha Private Key

Type: string

The reCaptcha private key associated with your account.

Corresponds to configuration file setting: reCaptcha.private.key

### Session Max Hours

Type: number

The number of hours before a session on the Ops Manager website expires.

Corresponds to configuration file setting: mms.session.maxHours

### mms.monitoring.agent.session.timeoutMillis

To set this, click *Config* and then click the *Custom* tab.

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.

## HTTP/HTTPS Proxy Settings

Ops Manager can pass all outgoing HTTP and HTTPS requests through an HTTP or HTTPS proxy.

### Proxy Host

Type: string

Specify the hostname of the HTTP or HTTPS proxy to which you wish to connect.

proxy.example.com

Corresponds to configuration file setting: http.proxy.host

### Proxy Port

Type: integer

Specify the port on which you wish to connect to the host. You must specify both the `Proxy Host` and `Proxy Port` to use a proxy.

Corresponds to configuration file setting: http.proxy.port

### **Proxy Username**

*Type:* string

If the proxy requires authentication, use this setting to specify the username with which to connect to the proxy.

Corresponds to configuration file setting: `http.proxy.username`

### **Proxy Password**

*Type:* string

If the proxy requires authentication, use this setting to specify the password with which to connect to the proxy.

Corresponds to configuration file setting: `http.proxy.password`

## **Twilio Integration Settings**

To receive alert notifications via SMS, you must have a [Twilio](#) account.

### **Account SID**

*Type:* string

Twilio account ID.

Corresponds to configuration file setting: `twilio.account.sid`

### **Twilio Auth Token**

*Type:* string

Twilio API token.

Corresponds to configuration file setting: `twilio.auth.token`

### **Twilio From Number**

*Type:* string

Twilio phone number.

Corresponds to configuration file setting: `twilio.from.num`

## **MongoDB Version Management**

The following settings determine how Ops Manager knows what MongoDB releases exist and how the MongoDB binaries are supplied to the Ops Manager server. The Automation Agents and Backup Daemons use these binaries when deploying MongoDB.

### **Version Manifest Source**

*Type:* string

*Default:* `mongodb`

Set this to `Local` if your Automation Agents and Backup Daemons will not have internet access to download MongoDB binaries. If you set this to `Local`, an Ops Manager admin must manually provide the version manifest and the MongoDB binaries, as described in [Configure Local Mode if Servers Have No Internet Access](#).

Corresponds to configuration file setting: `automation.versions.source`

### **Versions Directory**

*Type:* string

Specify the directory on the Ops Manager Application server where Ops Manager stores the MongoDB binaries. The Automation Agent accesses the binaries when installing or changing versions of MongoDB on your deployments. If you set `Version Manifest Source` to run in `Local` mode, the Backup Daemons also access

the MongoDB binaries from this directory. See [Configure Local Mode if Servers Have No Internet Access](#) for more information.

#### **Backup Versions Auto Download**

Type: boolean

Indicates whether the Backup Daemons automatically install the versions of MongoDB they run. If set to `true`, the daemons retrieve the binaries either over the internet from MongoDB Inc. or, if Ops Manager is running in [Local Mode](#), from the [Versions Directory](#) on the Ops Manager servers.

Set this to `false` only if Ops Manager does not use Automation **and** the Backup Daemons do not have internet access. If you set this to `false`, an Ops Manager admin must place the MongoDB binaries on the Backup Daemon servers.

Corresponds to configuration file setting: `mongodb.release.autoDownload`

#### **Backup Versions Auto Download Enterprise Builds**

Type: string

If [Backup Versions Auto Download](#) is set to `true`, specify whether the Daemon should download binaries for the Enterprise Edition.

**Warning:** If you will run [MongoDB Enterprise](#) and provision your own Linux servers, then you must manually install a set of dependencies to each server **before installing MongoDB**. See [Configure Local Mode if Servers Have No Internet Access](#).

Corresponds to configuration file setting: `mongodb.release.directory`

## **Backup Snapshots**

#### **File System Store Gzip Compression Level**

Type: int

Determines the compression level. Available levels are from 0 to 9:

- 0 for no compression.
- 1 to 9 increases the compression level, starting with level 1 which provides the fastest but less compression and up to level 9 which provides the best compression but is the slowest.

The default compression level is 6.

---

**Note:** The setting affects snapshots going forward and does not affect existing snapshots.

---

Corresponds to configuration file setting: `backup.fileSystemSnapshotStore.gzip.compressionLevel`

## **Public API**

You can modify certain default behaviors of the [Public API](#). To add the following settings, click the [Admin](#) link, then the [General](#) tab, then the [Ops Manager Config](#) page, and then the [Custom](#) section.

#### **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 this setting in the [Custom](#) section and set its value to `false`.

```
mms.publicApi.whitelistEnabled
```

Type: boolean

Certain API calls require that requests originate from a whitelisted IP address. To turn off this requirement, add this setting and set its value to `false`.

## Monitoring Agent Session Failover

If you have multiple monitoring agents for a group, only one Monitoring Agent 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. The following settings tune the frequency with which the standby agents poll Ops Manager to determine if they have become the primary agent, and the interval that Ops Manager uses to determine if the primary agent is unaccessible.

To add the following settings, click the *Admin* link, then the *General* tab, then the *Ops Manager Config* page, and then the *Custom* section.

```
mms.monitoring.agent.session.timeoutMillis
```

Type: integer

Default: 90000

The interval that Ops Manager uses to determine if a secondary agent should be promoted to primary. If Ops Manager does not hear from the primary agent for the duration specified in `mms.monitoring.agent.session.timeoutMillis`, Ops Manager promotes a secondary agent to primary. The minimum supported value is 90000.

```
mms.monitoring.agent.standbyCollectionFactor
```

Type: Integer

Default: 4

Specifies how frequently standby agents check in with Ops Manager to see if they have become the primary agent. The following values are permitted:

- 1: the standby agents check every 55 seconds.
- 2: the standby agents check in at twice the rate as 1, or approximately every 27 seconds.
- 3: the standby agents check approximately every 18 seconds
- 4: the standby agents check approximately every 14 seconds.

## 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, first download the Management Information Base (MIB) file at <http://downloads.mongodb.com/on-prem-monitoring/MMS-MONGODB-MIB.txt>. Then add the following settings as custom settings. To do so, click the *Admin* link, then the *General* tab, then the *Ops Manager Config* page, and then the *Custom* section.

```
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.

## Non-Uniform Memory Access (NUMA) Settings

`mongodb.disable.numa`

*Type:* boolean

To disable NUMA for the *head databases*:

1. Click the *Admin* link, then the *General* tab, then the *Ops Manager Config* page, and then the *Custom* section.
2. Add `mongodb.disable.numa` as a *Key* and set its *Value* to `true`.
3. Click *Save*.

See *MongoDB and NUMA Hardware* in the MongoDB Production Notes to learn more about NUMA.

---

**Important:** Each Ops Manager instance with Backup Daemons enabled must have the `numactl` service installed. If `numactl` is not installed and this setting is set to `true`, backup jobs fail.

---

## Backup Settings

To add the following settings, click the *Admin* link, then the *General* tab, then the *Ops Manager Config* page, and then the *Custom* section.

`mms.alerts.LowHeadFreeSpace.minimumHeadFreeSpaceGB`

*Type:* integer

*Default:* 10

Specifies the minimum amount of free disk space in GB required on the partition where the Backup Daemon stores the *head databases* that back up your data. You can view a daemon’s available head space on the Ops Manager *Daemons Page*. If the amount of free space drops below this minimum, Ops Manager triggers the following *system alert*:

`System detects backup daemon has low free head space`

`mms.alerts.BackupAgentConfCallFailure.maximumFailedConfCalls`

*Type:* integer

*Default:* 10

If the Backup Agent experiences more than this number of consecutive failed conf calls, Ops Manager triggers the following *global alert*:

```
Backup Agent has too many conf call failures  
mms.backup.minimumOplogWindowHours
```

To set this, click *Config* and then click the *Custom* tab.

Type: float

Default: 3

This sets the minimum number of hours that the oplog should record.

**Warning:** MongoDB recommends only changing this value temporarily to permit a test backup job to execute. The minimum oplog size value should be reset to the default as soon as possible. If an oplog is set to too small of a value, it can result in a gap between a backup job and an oplog which makes the backup unusable for restores. Stale backup jobs must be resynchronized before it can be used for restores. See also [Insufficient Oplog Size Error](#)

## Backup Daemon

The following settings are specific to a Backup Daemon and are set through the *Admin* interface, through the *Backup* tab's [Daemons page](#). These settings **are not** global but are specific to the daemon being configured. For a given daemon, you can set these locally through the [conf-mms.properties](#) configuration file.

### Head directory

If the directory is already configured, the path is listed in the *Server* column.

Type: string

The dedicated disk partition on the Backup Daemon's server where the daemon stores the [head databases](#). The daemon maintains a head database for each shard or replica set it backs up. 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 Database.

---

Corresponds to configuration file setting: `rootDirectory`

### Number of Workers

Type: number

The number of replica sets that should be processed at a time.

Corresponds to configuration file setting: `numWorkers`

## Ops Manager Application Database Connection String

The following settings configure the Ops Manager connection to the [Ops Manager Application Database](#). You must configure this setting in the [conf-mms.properties](#) file on each Ops Manager server. To encrypt authentication information, see [Encrypt User Credentials](#).

### mongo.mongoUri

Type: string

The [connection string](#) used to access the Ops Manager Application Database. The connection string **must** include the following if applicable:

- All members of the replica set, if the Ops Manager Application database is a replica set.

- Authentication credentials for the authentication mechanism used on the Ops Manager Application database.

See the following example connection strings:

- **Replica Sets:** If you use a *replica set* for the database's *backing instance*, specify all members of the replica set, as shown in the example below for a replica set named appdbRS. If you omit the port number, Ops Manager uses the default 27017 port for all hosts.

```
mongo.mongoUri=mongodb://db1.example.com:40000,db2.example.com:40000,db3.example.com:40000/?authMechanism=SCRAM-SHA-1&replicaSet=appdbRS
```

- **Default MongoDB Authentication:** For a MongoDB instance using the MongoDB SCRAM-SHA-1 or MONGODB-CR challenge-response mechanisms, the connection string must include authentication credentials. The Ops Manager Application must authenticate as a MongoDB user with the following roles:

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

Prefix the hostname with the MongoDB username and password in the form <username>:<password>@

```
mongo.mongoUri=mongodb://mongodbuser1:password@mydb1.example.com:40000
```

- **x.509 Certificate Authentication:** 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.com:40000/?authMechanism=MONGODBX509&ssl=true&sslPEMKeyFile=/path/to/cert.pem
```

- **LDAP Authentication:** 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.com:40000/?authMechanism=PLAIN&authSource=$external
```

- **Kerberos Authentication:** 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`.

This is an example of Kerberos authentication:

```
mongo.mongoUri=mongodb://username%40REALM.EXAMPLE.COM@mydb1.example.com:40000/?authMechanism=GSSAPI
```

To enable Kerberos authentication between the Ops Manager Application and the *Backup Data Storage*, see [Kerberos Authentication to the Application Database](#).

#### See also:

`authMechanism` and `authSource` in the MongoDB manual.

### `mongo.encryptedCredentials`

Type: boolean

To use encrypted credentials in `mongo.mongoUri`, encrypt the credentials using the Ops Manager `credentialstool`, enter them in the `mongo.mongoUri` setting, and set this to `true`:

```
mongo.encryptedCredentials=true
```

## SSL Connection to the Application Database

The following settings configure Ops Manager to use SSL to encrypt connections to the dedicated MongoDB instances that host the *Ops Manager Application Database* and *Backup Data Storage*. You must configure this setting in the `conf-mms.properties` file on each Ops Manager server.

### `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 or `net.ssl.CAFile` setting.

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 password using the Ops Manager `credentialstool`. See [Encrypt User Credentials](#).

## Kerberos Authentication to the Application Database

To enable Kerberos authentication between Ops Manager and the *Ops Manager Application Database*, configure the following settings in the `conf-mms.properties` file on each Ops Manager server. 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 User Credentials

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. If you choose to store credentials in plain text, reduce the permissions on the `conf-mms.properties` file on each server.

---

### Note: Protect Plain Text Passwords

If you choose to store credentials in plain text, reduce the permissions on the `conf-mms.properties` file on each server.

Operating System	Permission Changes
Linux	<code>sudo chmod 600 &lt;install_dir&gt;/conf/conf-mms.properties</code>
Windows	Restrict access to only the users and/or groups that need to modify <code>conf-mms.properties</code> .

---

**Important:** When installed with `rpm` or `deb` packages, the `credentialstool` tool requires root (`sudo`) privileges, because it modifies the `/etc/mongodb-mms/gen.key` file. Ops Manager uses the `gen.key` to encrypt sensitive data in the database and configuration files.

---

Use the `credentialstool` to generate encrypted credentials for the MongoDB deployments:

### Step 1: Run the shell command to create a pair of encrypted credentials.

Operating System	Command
Linux / Mac OS X	<code>sudo &lt;install_dir&gt;/bin/credentialstool.sh --username &lt;username&gt; --password &lt;password&gt;</code>
Windows	<code>&lt;install_dir&gt;\bin\credentialstool.bat --username &lt;username&gt; --password &lt;password&gt;</code>

Table 3: Substitutions

<code>&lt;username&gt;</code>	Your MongoDB username
<code>&lt;install_dir&gt;</code>	Path where Ops Manager was installed.

## **Step 2: Enter the password when prompted.**

The `credentialstool` then outputs the encrypted credential pair.

## **Step 3: Add the encrypted credentials to the `conf-mms.properties` file.**

1. Enter the encrypted credential pair in the `mongo.mongoUri` settings where needed.
2. Add the `mongo.encryptedCredentials` setting and set it to `true`.

---

### **Example**

```
mongo.mongoUri=mongodb://da83ex3s:a4fbef3a1@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.

---

## **14.2 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**

If a Linux server has an existing MongoDB deployment that was installed with a package manager, use the same package manager to install the Automation Agent. If a Linux server has an existing MongoDB deployment that was installed without a package manager, use the archive install.

**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 on Windows** Install and start the Automation Agent on Windows.

### **Install the Automation Agent with `rpm` Packages**

---

#### **On this page**

- [Overview](#)
- [Prerequisites](#)
- [Procedures](#)

**Overview** Ops Manager Automation relies on an Automation Agent, which must be installed on every server that runs a monitored MongoDB deployment. The Automation Agents periodically poll Ops Manager to determine the goal configuration, deploy changes as needed, and report deployment status back to Ops Manager.

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.

If you are installing to a server that has an existing MongoDB deployment that was **not** installed with the `rpm` package, then instead *install the agent using an archive* and ensure the agent runs as the same user as the MongoDB process.

## Prerequisites

**Server Networking Access** The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server's FQDN by issuing `hostname -f` in a shell connected to the server. 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.

**Installing to a Server that Already Runs MongoDB** If you install the Automation Agent to a server that is already running a MongoDB process, the agent must have:

- Permission to stop the MongoDB process. The Automation Agent will restart the process using the agent's own set of MongoDB binaries. If you had installed MongoDB with a package manager, use the same package manager to install the Automation Agent. This gives the agent the same owner as MongoDB.
- Read and Write permissions on the MongoDB data directory and log directory.
- Permission to stop, start, and update any existing Monitoring and Backup Agents.

**Installing to a Server Before Installing MongoDB** If you deploy the Automation Agent to a server that does not have MongoDB installed, ensure the user that owns the Automation Agent has Read and Write permissions on the MongoDB data and log directories you plan to use.

**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
```

### 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 the group's *agent API key*.

```
mmsGroupId=<Group ID>
mmsApiKey=<agent API key>
```

For mmsBaseUrl, enter the URL of the Ops Manager Application. Include the port number.

```
mmsBaseUrl=<application URL>
```

For SUSE 11+ deployments only, configure the sslTrustedMMServerCertificate setting. All other users should omit this step.

```
sslTrustedMMServerCertificate=/etc/ssl/certs/UTN_USERFirst_Hardware_Root_CA.pem
```

**Step 4: Optional: Configure the Automation Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the httpProxy environment variable. In the /etc/mongodb-mms/automation-agent.config file, set the httpProxy value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 5: Prepare the data directory.** The data directory stores MongoDB data and must be owned by the `mongod` user. For an existing MongoDB deployment, ensure the directory has the mongod user as owner. If no MongoDB deployment exists, create the directory and set the owner.

The following commands create a data directory and set the owner as the mongod user:

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

**Step 6: Start the Automation Agent.** Issue the following command:

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

#### Update the Automation Agent with an rpm Package

**Important:** The preferred way to update the Automation Agent is through the Ops Manager UI. If an Automation Agent is out of date, Ops Manager displays a warning on the *Deployment* page and provides a link to perform the update automatically.

---

If you use this procedure, you do **not** need to stop the agent. 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
```

#### Step 2: Install the Automation Agent Package

```
sudo rpm -U mongodb-mms-automation-agent-manager-latest.x86_64.rpm
```

**Step 3: Prepare the data directory.** The data directory stores MongoDB data and must be owned by the ‘mongod’ user. For an existing MongoDB deployment, ensure the directory has the mongod user as owner. If no MongoDB deployment exists, create the directory and set the owner.

The following commands create a data directory and set the owner as the mongod user:

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

## Install the Automation Agent with deb Packages

### On this page

- [Overview](#)
- [Prerequisites](#)
- [Procedures](#)

**Overview** Ops Manager Automation relies on an Automation Agent, which must be installed on every server that runs a monitored MongoDB deployment. The Automation Agents periodically poll Ops Manager to determine the goal configuration, deploy changes as needed, and report deployment status back to Ops Manager.

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.

To install the Automation Agent to a server that has an existing MongoDB deployment that was **not** installed with a deb package, use the [Install the Automation Agent from an Archive](#) procedure.

### Prerequisites

**64-Bit Architecture** The Automation Agent can run only on a 64-bit architecture.

**Server Networking Access** The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server’s FQDN by issuing `hostname -f` in a shell connected to the server. 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.

**Installing to a Server that Already Runs MongoDB** If you install the Automation Agent to a server that is already running a MongoDB process, the agent must have:

- Permission to stop the MongoDB process. The Automation Agent will restart the process using the agent’s own set of MongoDB binaries. If you had installed MongoDB with a package manager, use the same package manager to install the Automation Agent. This gives the agent the same owner as MongoDB.
- Read and Write permissions on the MongoDB data directory and log directory.
- Permission to stop, start, and update any existing Monitoring and Backup Agents.

**Installing to a Server Before Installing MongoDB** If you deploy the Automation Agent to a server that does not have MongoDB installed, ensure the user that owns the Automation Agent has Read and Write permissions on the MongoDB data and log directories you plan to use.

**Ubuntu 12.04** If you install the Automation Agent to an Ubuntu 12.04 machine, and you plan to deploy MongoDB instances that use *LDAP authentication*, then you must either:

- manually change the primary group of the `mongodb` user to `sasl`, or
- *install the agent from the tar.gz archive*.

Alternatively, you can use Ubuntu 14.04.

**Server Networking Access** The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server's FQDN by issuing `hostname -f` in a shell connected to the server. 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.

**Root Access** To install the Automation Agent using a `deb` package, you must have root access.

**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 the group's *agent API key*.

```
mmsGroupId=<Group ID>
mmsApiKey=<agent API Key>
```

For `mmsBaseUrl`, enter the URL of the Ops Manager Application. Include the port number.

```
mmsBaseUrl=<application URL>
```

**Step 4: Optional: Configure the Automation Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `/etc/mongodb-mms/automation-agent.config` file, set the `httpProxy` value to the URL of your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 5: Prepare the data directory.** The data directory stores MongoDB data. For an existing MongoDB deployment, ensure that the directory is owned by the `mongodb` user. If no MongoDB deployment exists, create the directory and set the owner.

```
sudo mkdir /data
sudo chown mongodb:mongodb /data
```

**Step 6: Start the Automation Agent.** Issue the following command:

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

#### Update the Automation Agent with a deb Package

**Important:** The preferred way to update the Automation Agent is through the Ops Manager UI. If an Automation Agent is out of date, Ops Manager displays a warning on the *Deployment* page and provides a link to perform the update automatically.

If you use this procedure, you do **not** need to stop the agent. 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
```

#### Step 2: Install the Automation Agent Package.

```
sudo dpkg -i mongodb-mms-automation-agent-manager_latest_amd64.deb
```

**Step 3: Prepare the data directory.** The data directory stores MongoDB data. For an existing MongoDB deployment, ensure that the directory is owned by the `mongodb` user. If no MongoDB deployment exists, create the directory and set the owner.

```
sudo mkdir /data
sudo chown mongodb:mongodb /data
```

#### Install the Automation Agent from an Archive

##### On this page

- Overview
- Prerequisites
- Procedure

**Overview** Ops Manager Automation relies on an Automation Agent, which must be installed on every server that runs a monitored MongoDB deployment. The Automation Agents periodically poll Ops Manager to determine the goal configuration, deploy changes as needed, and report deployment status back to Ops Manager.

This procedure installs the Automation Agent on a Linux server using an `tar.gz` archive file.

Automation Agents can run only on 64-bit architectures.

## Prerequisites

**Server Networking Access** The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server's FQDN by issuing `hostname -f` in a shell connected to the server. 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.

**Installing to a Server that Already Runs MongoDB** If you install the Automation Agent to a server that is already running a MongoDB process, the agent must have:

- Permission to stop the MongoDB process. The Automation Agent will restart the process using the agent's own set of MongoDB binaries. If you had installed MongoDB with a package manager, use the same package manager to install the Automation Agent. This gives the agent the same owner as MongoDB.
- Read and Write permissions on the MongoDB data directory and log directory.
- Permission to stop, start, and update any existing Monitoring and Backup Agents.

**Installing to a Server Before Installing MongoDB** If you deploy the Automation Agent to a server that does not have MongoDB installed, ensure the user that owns the Automation Agent has Read and Write permissions on the MongoDB data and log directories you plan to use.

**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 agent 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 the group's *agent API key*.
- For `mmsBaseUrl`, enter the URL of the Ops Manager Application. Include the port number.

```
mmsGroupId=<Group ID>
mmsApiKey=<agent API key>
mmsBaseUrl=<application URL>
```

**Step 4: Optional: Configure the Automation Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/local.config` file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 5: Create the automation directories and data directory.** Create the following directories and ensure that the user running the agent owns the directories:

- `/var/lib/mongodb-mms-automation`
- `/var/log/mongodb-mms-automation`
- `/data`

**Step 6: 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

**Important:** The preferred way to update the Automation Agent is through the Ops Manager UI. If an Automation Agent is out of date, Ops Manager displays a warning on the *Deployment* page and provides a link to perform the update automatically.

**Step 1: Stop any currently running Automation Agents.** Issue the following command:

```
pskill -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 agent 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 the group's *agent API key*.
- For `mmsBaseUrl`, enter the URL of the Ops Manager Application. Include the port number.

```
mmsGroupId=<Group ID>
mmsApiKey=<agent API key>
mmsBaseUrl=<application URL>
```

**Step 5: Optional: Configure the Automation Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/local.config` file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 6: Create the automation directories and data directory.** Create the following directories and ensure that the user running the agent owns the directories:

- `/var/lib/mongodb-mms-automation`
- `/var/log/mongodb-mms-automation`
- `/data`

**Step 7: 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](#)
- [Prerequisites](#)
- [Procedure](#)

**Overview** Ops Manager Automation relies on an Automation Agent, which must be installed on every server that runs a monitored MongoDB deployment. The Automation Agents periodically poll Ops Manager to determine the goal configuration, deploy changes as needed, and report deployment status back to Ops Manager.

Automation Agents can run only on 64-bit architectures.

### Prerequisites

**Server Networking Access** The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server's FQDN by issuing `hostname -f` in a shell connected to the server. 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.

**Installing to a Server that Already Runs MongoDB** If you install the Automation Agent to a server that is already running a MongoDB process, the agent must have:

- Permission to stop the MongoDB process. The Automation Agent will restart the process using the agent's own set of MongoDB binaries. If you had installed MongoDB with a package manager, use the same package manager to install the Automation Agent. This gives the agent the same owner as MongoDB.
- Read and Write permissions on the MongoDB data directory and log directory.
- Permission to stop, start, and update any existing Monitoring and Backup Agents.

**Installing to a Server Before Installing MongoDB** If you deploy the Automation Agent to a server that does not have MongoDB installed, ensure the user that owns the Automation Agent has Read and Write permissions on the MongoDB data and log directories you plan to use.

**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 agent 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 the group's *agent API key*.
- For `mmsBaseUrl`, enter the URL of the Ops Manager Application. Include the port number.

```
mmsGroupId=<Group ID>
mmsApiKey=<agent API key>
mmsBaseUrl=<application URL>
```

**Step 4: Optional: Configure the Automation Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/local.config` file, set the `httpProxy` value to the URL of your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 5: Create the automation directories and data directory.** Create the following directories and ensure that the user running the agent owns the directories:

- `/var/lib/mongodb-mms-automation`
- `/var/log/mongodb-mms-automation`
- `/data`

**Step 6: Start the Automation Agent.** Issue the following command:

```
nohup ./mongodb-mms-automation-agent --config=local.config >> /var/log/mongodb-mms-automation/automation.log
```

#### Update the Automation Agent on OS X

**Important:** The preferred way to update the Automation Agent is through the Ops Manager UI. If an Automation Agent is out of date, Ops Manager displays a warning on the *Deployment* page and provides a link to perform the update automatically.

**Step 1: Stop any currently running Automation Agents.** Issue the following command:

```
ps aux | grep mongo | grep -v grep | awk '{print $2}' | xargs kill -9
```

**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 4: Edit the `local.config` file to include your Group ID and agent 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 the group's *agent API key*.
- For `mmsBaseUrl`, enter the URL of the Ops Manager Application. Include the port number.

```
mmsGroupId=<Group ID>
mmsApiKey=<agent API key>
mmsBaseUrl=<application URL>
```

**Step 5: Optional: Configure the Automation Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/local.config` file, set the `httpProxy` value to the URL of your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 6: Create the automation directories and data directory.** Create the following directories and ensure that the user running the agent owns the directories:

- `/var/lib/mongodb-mms-automation`
- `/var/log/mongodb-mms-automation`
- `/data`

**Step 7: Start the Automation Agent.** Issue the following command:

```
nohup ./mongodb-mms-automation-agent --config=local.config >> /var/log/mongodb-mms-automation/automation.log
```

## Install the Automation Agent on Windows

### On this page

- [Overview](#)
- [Prerequisites](#)
- [Procedure](#)

**Overview** Ops Manager Automation relies on an Automation Agent, which must be installed on every server that runs a monitored MongoDB deployment. The Automation Agents periodically poll Ops Manager to determine the goal configuration, deploy changes as needed, and report deployment status back to Ops Manager.

Automation Agents can run only on 64-bit architectures.

### Prerequisites

**Server Networking Access** The servers that host the MongoDB processes must have full networking access to each other through their fully qualified domain names (FQDNs). You can view a server's FQDN by issuing `hostname -f` in a shell connected to the server. 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.

**Installing to a Server that Already Runs MongoDB** If you install the Automation Agent to a server that is already running a MongoDB process, the agent must have:

- Permission to stop the MongoDB process. The Automation Agent will restart the process using the agent's own set of MongoDB binaries. If you had installed MongoDB with a package manager, use the same package manager to install the Automation Agent. This gives the agent the same owner as MongoDB.
- Read and Write permissions on the MongoDB data directory and log directory.

- Permission to stop, start, and update any existing Monitoring and Backup Agents.

**Installing to a Server Before Installing MongoDB** If you deploy the Automation Agent to a server that does not have MongoDB installed, ensure the user that owns the Automation Agent has Read and Write permissions on the MongoDB data and log directories you plan to use.

---

**Important:** Windows Firewall stealth mode should be disabled on the MongoDB servers on which the Automation Agent is installed for best performance.

---

**Procedure** This section includes procedures for both installing and updating the Automation Agent.

### Install the Automation Agent on Windows

#### Step 1: Login to Ops Manager.

**Step 2: Begin or Manage a Deployment** When you reach the *Install an Automation Agent on each server* page, each server is listed.

1. Click on the *Install Agent* menu of the first MongoDB server.
2. Select *Windows - MSI*.

#### Step 3: Install the Automation Agent

1. The *Automation Agent Installation Instructions* box displays with three important values that are needed during installation:
  - Base URL
  - Group ID
  - API Key

These values identify which server manages this agent and are needed for Step e, so keep this box open.

2. After the MSI downloads, double-click the *mongodb-mms-automation-agent-<version>.windows\_x86\_64.msi*.
3. If a security warning appears, click *Run*.
4. At the *Configuration/Log Folder* step, provide the directory into which these files are saved.
5. At the *MMS Base URL and Agent API Key* step, enter the values from the dialog box in the corresponding fields.
6. Specify the Log and Backup directories.
7. Click *Finish* once installation is complete.

**Step 4: Verify the Automation Agent is running.** The Automation Agent starts automatically after installation. This step verifies that it is running and can communicate with Ops Manager.

#### Step 5: Repeat installation procedure for each MongoDB server.

## **Update the Automation Agent on Windows**

---

**Important:** The preferred way to update the Automation Agent is through the Ops Manager UI. If an Automation Agent is out of date, Ops Manager displays a warning on the *Deployment* page and provides a link to perform the update automatically.

---

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

**Step 2: Shut down Ops Manager Automation Agent.** To shutdown the Ops Manager Automation Agent:

1. Click the *Start* button.
2. Click *Administrative Tools*.
3. Click *Services*.
4. Right-click on the *MMS Automation Agent* and select *Stop*.

**Step 3: Download the latest agent.** From a system shell, issue the following command, where <OpsManagerURI> is the URI of your Ops Manager installation (for example, `onprem.example.net`):

```
curl -OL <OpsManagerURI>/download/agent/automation/mongodb-mms-automation-agent-latest.windows_x86_64.msi
```

## **Step 4: Install the Automation Agent**

1. After the MSI downloads, double-click the *mongodb-mms-automation-agent-<version>.windows\_x86\_64.msi*
2. If a security warning appears, click *Run*.
3. Verify that the *Destination Folder* installation path and *Configuration/Log Folder* paths are correct.
4. At the *MMS Base URL and Agent API Key* step, verify the provided values.
5. At the *Automation Paths* step, verify the provided paths.
6. Click *Install*.
7. Click *Finish* once installation is complete.

## **Step 5: Login to Ops Manager.**

**Step 6: Verify the Automation Agent is running.** The Deployment page should display after login.

1. Click on the stacked box icon next to *Servers*.
2. Verify that the *Automation Agent* is showing the correct version.

## **Step 7: Repeat installation procedure for each MongoDB server.**

## Automation Agent Configuration

### On this page

- Configuration File
- Settings

### 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 *Settings* 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 *Settings* 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 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
```

**MongoDB Kerberos Settings** Specify these settings if the Automation Agent authenticates to hosts using Kerberos. See [Configure the Backup Agent for Kerberos](#).

### **krb5ConfigLocation**

Type: string

The *absolute* path to an non-system-standard location for the Kerberos configuration file. For example:

```
krb5ConfigLocation=/etc/krb_custom.conf
```

## MongoDB SSL Settings

### **sslTrustedMMServerCertificate**

Type: string

The path on disk that contains the trusted certificate authority certificates in PEM format. Because the Automation agent can pass `sslTrustedMMServerCertificate` to the other agents, which may have different paths relative to the automation agent, specify the **absolute path** to the certificate. This certificate verifies that the agent is talking to the Ops Manager servers.

```
sslTrustedMMServerCertificate=/etc/ssl/ca.pem
```

### **sslRequireValidMMServerCertificates**

Type: boolean

Use this option to disable certificate verification by setting this value to `false`. This configuration is recommended only for testing purposes as it makes connections susceptible to man-in-the-middle attacks.

```
sslRequireValidMMServerCertificates=true
```

## 14.3 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 Ops Manager, which processes and renders the data.

**Install or Update with RPM Packages** Install or update the Monitoring Agent using an `xpm` 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 `xpm` Packages

### On this page

- Overview
- Considerations
- 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 Ops Manager, which processes and renders this data. The agent initiates all connections between the agent and Ops Manager, and communications between the agent and Ops Manager 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 the Monitoring Agent on your system. You must install the Ops Manager 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 Ops Manager on port 443 (i.e. `https`.)

Ops Manager does not make *any* outbound connections to the agents or to MongoDB instances. If [Exposed DB Host Check is enabled](#), Ops Manager 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. 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.

`mms.monitoring.agent.standbyCollectionFactor` configures the frequency at which the standby agents check to see if they have become the primary agent. By default, the standby agents check every 14 seconds. See the `mms.monitoring.agent.standbyCollectionFactor` reference for details.

Ops Manager promotes a standby agent to primary after not hearing from the current primary for at least the interval specified by `mms.monitoring.agent.session.timeoutMillis`. The default delay is 90 seconds (90000 milliseconds), which is also the minimum.

You can tune `mms.monitoring.agent.standbyCollectionFactor` and `mms.monitoring.agent.session.timeoutMillis` by editing *Ops Manager Configuration*.

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 Ops Manager. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Ops Manager 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 *Settings* 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 agent API key.** In the `/etc/mongodb-mms/monitoring-agent.config` file, set the `mmsApiKey` property to your API key.

**Step 5: Optional: Configure the Monitoring Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `/etc/mongodb-mms/monitoring-agent.config` file, set the `httpProxy` value to the URL of your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 6: Optional: For SUSE deployments only, configure the `sslTrustedMMServerCertificate` property.** If you're deploying on SUSE, you must configure the `sslTrustedMMServerCertificate` setting. All other users should omit this step.

Enter the following property and value in the `/etc/mongodb-mms/monitoring-agent.config` file:

```
sslTrustedMMServerCertificate=/etc/ssl/certs/UTN_USERFirst_Hardware_Root_CA.pem
```

Save and close the file.

**Step 7: 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 <mmmsUri>/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
- Considerations
- 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 Ops Manager, which processes and renders this data. The agent initiates all connections between the agent and Ops Manager, and communications between the agent and Ops Manager 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 the Monitoring Agent on your system. You must install the Ops Manager 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 Ops Manager on port 443 (i.e. https.)

Ops Manager does not make *any* outbound connections to the agents or to MongoDB instances. If [Exposed DB Host Check is enabled](#), Ops Manager 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. 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.

`mms.monitoring.agent.standbyCollectionFactor` configures the frequency at which the standby agents check to see if they have become the primary agent. By default, the standby agents check every 14 seconds. See the `mms.monitoring.agent.standbyCollectionFactor` reference for details.

Ops Manager promotes a standby agent to primary after not hearing from the current primary for at least the interval specified by `mms.monitoring.agent.session.timeoutMillis`. The default delay is 90 seconds (90000 milliseconds), which is also the minimum.

You can tune `mms.monitoring.agent.standbyCollectionFactor` and `mms.monitoring.agent.session.timeoutMillis` by editing [Ops Manager Configuration](#).

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 Ops Manager. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Ops Manager 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 *Settings* 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 agent API key.** In the /etc/mongodb-mms/monitoring-agent.config file, set the `mmsApiKey` property to your API key.

**Step 5: Optional: Configure the Monitoring Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the /etc/mongodb-mms/monitoring-agent.config file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 6: 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
- Considerations
- 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 Ops Manager, which processes and renders this data. The agent initiates all connections between the agent and Ops Manager, and communications between the agent and Ops Manager 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 the Monitoring Agent on your system. You must install the Ops Manager 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 Ops Manager on port 443 (i.e. `https`.)

Ops Manager does not make *any* outbound connections to the agents or to MongoDB instances. If [Exposed DB Host Check is enabled](#), Ops Manager 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. 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.

`mms.monitoring.agent.standbyCollectionFactor` configures the frequency at which the standby agents check to see if they have become the primary agent. By default, the standby agents check every 14 seconds. See the `mms.monitoring.agent.standbyCollectionFactor` reference for details.

Ops Manager promotes a standby agent to primary after not hearing from the current primary for at least the interval specified by `mms.monitoring.agent.session.timeoutMillis`. The default delay is 90 seconds (90000 milliseconds), which is also the minimum.

You can tune `mms.monitoring.agent.standbyCollectionFactor` and `mms.monitoring.agent.session.timeoutMillis` by editing [Ops Manager Configuration](#).

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 Ops Manager. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Ops Manager 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 *Settings* 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 Monitoring Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the <install-directory>/monitoring-agent.config file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**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 *Settings* 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: Optional: Configure the Monitoring Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the <install-directory>/monitoring-agent.config file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 7: 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](#)
- [Considerations](#)
- [Prerequisites](#)
- [Procedures](#)
- [Additional Information](#)

**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 Ops Manager, which processes and renders this data. The agent initiates all connections between the agent and Ops Manager, and communications between the agent and Ops Manager 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 the Monitoring Agent on your system. You must install the Ops Manager 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 Ops Manager on port 443 (i.e. `https`.)

Ops Manager does not make *any* outbound connections to the agents or to MongoDB instances. If [Exposed DB Host Check is enabled](#), Ops Manager 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. 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.

`mms.monitoring.agent.standbyCollectionFactor` configures the frequency at which the standby agents check to see if they have become the primary agent. By default, the standby agents check every 14 seconds. See the `mms.monitoring.agent.standbyCollectionFactor` reference for details.

Ops Manager promotes a standby agent to primary after not hearing from the current primary for at least the interval specified by `mms.monitoring.agent.session.timeoutMillis`. The default delay is 90 seconds (90000 milliseconds), which is also the minimum.

You can tune `mms.monitoring.agent.standbyCollectionFactor` and `mms.monitoring.agent.session.timeoutMillis` by editing [Ops Manager Configuration](#).

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 Ops Manager. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Ops Manager 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 [Settings](#) 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 *Settings* 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 Monitoring Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/monitoring-agent.config` file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**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 *Settings* 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: Optional: Configure the Monitoring Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the <install-directory>/monitoring-agent.config file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 7: 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 <http://docs.opsmanager.mongodb.com//tutorial/rotate-agent-log-files> to configure log rotation.

## Install or Update the Monitoring Agent on Windows

### On this page

- [Overview](#)
- [Considerations](#)
- [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 Ops Manager, which processes and renders this data. The agent initiates all connections between the agent and Ops Manager, and communications between the agent and Ops Manager 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 the Monitoring Agent on your system. You must install the Ops Manager 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 Ops Manager on port 443 (i.e. https.)

Ops Manager does not make *any* outbound connections to the agents or to MongoDB instances. If *Exposed DB Host Check is enabled*, Ops Manager 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. 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.

`mms.monitoring.agent.standbyCollectionFactor` configures the frequency at which the standby agents check to see if they have become the primary agent. By default, the standby agents check every 14 seconds. See the `mms.monitoring.agent.standbyCollectionFactor` reference for details.

Ops Manager promotes a standby agent to primary after not hearing from the current primary for at least the interval specified by `mms.monitoring.agent.session.timeoutMillis`. The default delay is 90 seconds (90000 milliseconds), which is also the minimum.

You can tune `mms.monitoring.agent.standbyCollectionFactor` and `mms.monitoring.agent.session.timeoutMillis` by editing *Ops Manager Configuration*.

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 Ops Manager. If you have multiple agents, this delay permits other agents in your infrastructure to become the primary agent and permits Ops Manager 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 *Settings* 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 agent API key, the hostname of the Monitoring server and, optionally, the URL for a proxy server.** In the C:\MMSData\Monitoring\monitoring-agent.config file, set these properties:

Property	Value
mmsApiKey	Your API key
mmsBaseUrl	Hostname of the Monitoring server
httpProxy (optional)	The URL of to your proxy server

**Step 4: 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 *Settings* 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 agent API key and, optionally, the URL for a proxy server.** In the `C:\MMSData\Monitoring\monitoring-agent.config` file, set these properties:

Property	Value
<code>mmsApiKey</code>	Your API key
<code>httpProxy</code> (optional)	The URL of to your proxy server

**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

- Configuration File
- Settings

**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

Ops Manager provides default values for many of the Monitoring Agent Configuration settings.

---

**Important:** You must set the `mmsApiKey` value.

---

**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 *Settings* 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 Ops Manager URL. For example:

```
mmsBaseUrl=http://example.com:8080
```

## HTTP Proxy Settings

### `httpProxy`

Type: string

Specifies the URL of an HTTP proxy server the Monitoring Agent can use.

```
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
```

#### **krb5ConfigLocation**

Type: string

The *absolute* path to an non-system-standard location for the Kerberos configuration file. For example:

```
krb5ConfigLocation=/etc/krb_custom.conf
```

#### **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.

**sslTrustedMMServerCertificate**

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 `sslTrustedMMServerCertificate`.

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:

```
sslTrustedMMServerCertificate=/etc/mongodb-mms/mms-certs.pem
```

**sslRequireValidMMServerCertificates**

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 want 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

- Considerations
- Prerequisites
- 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:

- [Configure Monitoring Agent 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.

---

**Important:** The Monitoring Agent user must be defined consistently for all processes in your Ops Manager deployment.

---

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<sup>2</sup> and [database profiling](#) information<sup>3</sup>, the monitoring agent must authenticate to the database as a user with the following access:

Required Role	
clusterMonitor role on the admin database	

<sup>2</sup> Monitoring without dbStats excludes database storage, records, indexes, and other statistics.

<sup>3</sup> Profiling captures in-progress read and write operations, cursor operations, and database command information about the database.

For *mixed* MongoDB versions that use Monitoring Agents older than version 3.7.0, the specified access is inadequate to monitor deployments 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 pre-3.7.0 Monitoring Agent can recover from this error but causes the extra log entries. You can safely ignore these extra entries.

## 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	
<code>clusterAdmin</code> role on the <code>admin</code> database	
<code>readAnyDatabase</code> role on the <code>admin</code> 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 <sup>1</sup>, the agent must authenticate as a user with the following access:

Required Roles	
<code>clusterAdmin</code> role on the <code>admin</code> database	
<code>readAnyDatabase</code> role on the <code>admin</code> database	
<code>dbAdminAnyDatabase</code> roles in the <code>admin</code> database	

**Monitor without dbStats** To monitor MongoDB 2.4 databases *without* `dbStats` <sup>2</sup>, the agent must authenticate as a user with the following access:

Required Role	
<code>clusterAdmin</code> role on the <code>admin</code> 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.

**Configure for x.509** Procedure to configure the Monitoring Agent for MongoDB deployments using x.509 Client Certificate authentication.

## Configure Monitoring Agent for MONGODB-CR

### On this page

- Procedures

In MongoDB 3.0 and later, MongoDB's default authentication mechanism is a challenge and response mechanism (SCRAM-SHA-1). Previously, MongoDB used MongoDB Challenge and Response (MONGODB-CR) as the default.

The Monitoring Agent can use MONGODB-CR or SCRAM-SHA-1 to authenticate to hosts that enforce access control.

To authenticate using SCRAM-SHA-1 or MONGODB-CR, create a user in the `admin` database with the appropriate roles in MongoDB.

---

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage agent authentication for you if you use Automation to manage the agents. With Automation, Ops Manager creates the users for each agent and configures the agent appropriately. See: [Enable SCRAM-SHA-1 / MONGODB-CR Authentication for your Ops Manager Group](#) for more information.

---

### Procedures

**Create MongoDB User for the Agent** 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.

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 2.6 and Later** 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"
  ]
}
)
```

Refer to the [Access Control for MongoDB 2.4](#) reference to choose which MongoDB roles to provide for the Monitoring Agent. For example, if you wish to monitoring with database profiling, you will need to include the `dbAdminAnyDatabase` role.

**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

- [Considerations](#)
- [Procedures](#)

If your MongoDB deployment enforces access control, the Monitoring Agent must authenticate to MongoDB as a user with the proper access.

Starting with version 2.6, [MongoDB Enterprise](#) for Linux provides support for proxy authentication of users. This allows administrators to configure a MongoDB cluster to authenticate users by proxying authentication requests to a specified Lightweight Directory Access Protocol (LDAP) service. Monitoring Agents support authenticating to MongoDB instances using LDAP.

MongoDB Enterprise for Windows does **not** include LDAP support for authentication.

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.

---

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage agent authentication for you if you use Automation to manage the agents. With Automation, Ops Manager creates the users for each agent and configures the agent appropriately. See: [Enable LDAP Authentication for your Ops Manager Group](#) for more information.

---

**Considerations** You must configure LDAP authentication separately for each agent. See [Configure Backup Agent for LDAP Authentication](#) for configuration instructions for the Backup Agent.

You can configure LDAP authentication when adding a host or by editing an existing host. See [Enable LDAP Authentication for your Ops Manager Group](#) for instructions.

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.

## Procedures

**Create User in MongoDB** To monitor MongoDB 2.6+ instances that are using LDAP authentication, add a user that possess the required roles to the `$external` database in MongoDB. The `$external` database allows `mongod` to consult an external source, such as an LDAP server, to authenticate.

Use the following commands to create the users from a mongo shell connected to your MongoDB deployment:

```
db.getSiblingDB("$external").createUser(  
  {  
    user : "<username>",  
    roles: [ { role: "clusterMonitor", db: "admin" } ]  
  }  
)
```

See [Required Access for Monitoring Agent](#) 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

- [Prerequisites](#)
- [Procedures](#)

*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.

---

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage agent authentication for you if you use Automation to manage the agents. With Automation, Ops Manager creates the users for each agent and configures the agent appropriately. See: [Enable Kerberos Authentication for your Ops Manager Group](#) for more information.

---

**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.

## Procedures

### Create 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** 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.

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. If each agent is going to use its own Kerberos Principal, then you must create a user in the \$external database for each Kerberos Principal.

Use the following commands to create the users from a mongo shell connected to your MongoDB deployment:

```
use $external
db.createUser(
{
  user: "<Kerberos Principal>",
  roles: [ { role: "clusterMonitor", db: "admin" } ]
}
)
```

See [Required Access for Monitoring Agent](#) for more information on the required access.

If you are using the same Kerberos Principal for both the Monitoring and Backup Agents, the user must possess the required roles for both 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/krb5.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 the Monitoring Agent User for x.509 Client Certificate Authentication

### On this page

- Considerations
- Procedures

Ops Manager enables you to configure the Authentication Mechanisms that the Ops Manager Agents use to connect to your MongoDB deployments from within the Ops Manager interface. You can enable multiple authentication mechanisms for your group, but you must choose a single mechanism for the Agents to use to authenticate to your deployment.

MongoDB supports x.509 certificate authentication for use with a secure [TLS/SSL](#) connection. The x.509 client authentication allows clients to authenticate to servers with certificates rather than with a username and password.

In Ops Manager, x.509 Client Certificate (MONGODB-X509) is only available on MongoDB Enterprise builds. If you have existing deployments running on a MongoDB Community build, you must [upgrade them to MongoDB Enterprise](#) before you can enable x.509 Client Certificate (MONGODB-X509) for your Ops Manager group.

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage agent authentication for you if you use Automation to manage the agents. With Automation, Ops Manager creates the users for each agent and configures the agent appropriately. See: [Enable x.509 Authentication for your Ops Manager Group](#) for more information.

### Considerations

**Important:** A full description of TLS/SSL, PKI (Public Key Infrastructure) certificates, in particular x.509 certificates, and Certificate Authority is beyond the scope of this document. This tutorial assumes prior knowledge of TLS/SSL as well as access to valid x.509 certificates.

In order to enable x.509 Authentication for Ops Manager, you must obtain valid certificates generated and signed by a single certificate authority. Refer to the [Client x.509 Certificate](#) in the MongoDB Manual for more about the certificate requirements.

**Important:** x.509 Client Certificate Authentication requires that SSL be enabled and configured for the deployment.

**Procedures** This tutorial assumes that you have already configured your MongoDB deployment to use x.509 certificate authentication and SSL. If you have not done so, refer to the [Use x.509 Certificates to Authenticate Clients](#) and [Configure mongod and mognos for TLS/SSL](#) tutorials.

**Create MongoDB User for the subject** In order for the Monitoring Agent to connect to your MongoDB deployment, you must create a user for the Monitoring Agent in the \$external database.

For x.509 certificate authentication, use the `subject` value of your client certificate as the username.

Use the following commands to create the users from a mongo shell connected to your MongoDB deployment:

```
use $external
db.createUser(
{
    user: "<x.509 subject>",
    roles: [ { role: "clusterMonitor", db: "admin" } ]
})
```

See [Required Access for Monitoring Agent](#) for more information on the required access.

You can only associate an x.509 client certificate with a single user: each user must have its own certificate. However, you may use the same user for both the Monitoring and Backup agents. If you choose to use the same user for both agents, ensure that the user possesses the required permissions for both the monitoring agent **and** the [backup agent](#).

**Edit Agent Configuration File** x.509 requires that you configure the agent for 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 the agent.** [Configure Monitoring Agent for SSL](#) provides more details about configuring the Monitoring Agent for SSL.

Once you have configured the Monitoring agent, you still need to configure the x.509 Authentication mechanism in the Ops Manager interface, as described in [Enable x.509 Authentication for your Ops Manager Group](#).

## Configure Monitoring Agent for SSL

### On this page

- [Overview](#)
- [Prerequisite](#)
- [Procedures](#)

### 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](#).

---

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage SSL for you if you use Automation for the deployment. With Automation, Ops Manager prompts you for the certificates to use to connect to the deployment when you enable SSL and then configures the agents appropriately. See: [Enable SSL for a Deployment](#) for more information.

---

**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 the 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](#)
- [Install the munin-node Package](#)
- [Configure munin-node](#)
- [Additional Considerations for munin-node](#)

### 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.

For details on installing `munin-node` beyond the information provided here, please see <http://guide.munin-monitoring.org/en/latest/installation/install.html>.

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 appropriate command to start it, depending on your operating system.

---

**Note:** If you use `systemctl`, and if the `munin-node` service is not shown, try to reload the SystemD configuration and unit files.

- 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
```

Then restart `munin-node`.

---

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](#)
- [Procedures](#)

### 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 MongoDB 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 MongoDB Monitoring Agent. Select the Action menu and select Stop.

## Remove Monitoring Agents from Ops Manager

### On this page

- [View Active Agents](#)
- [Remove from Ops Manager](#)
- [Delete from the Server](#)

### View Active Agents

To view your active Monitoring Agents, click *Deployment*, then the *Agents* tab, then *All Agents*.

Active agents are those that have pinged Ops Manager in the last 5 minutes.

## Remove from Ops Manager

To remove a Monitoring Agent from Ops Manager, [stop the agent](#) and then wait 5 minutes.

When an agent fails to ping Ops Manager for 5 minutes, Ops Manager views the agent as inactive and distributes the agent's assignments to other agents (if available).

---

**Note:** The *All Agents* page displays an inactive agent for 24 hours before Ops Manager removes it from the page.

---

## 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 MongoDB Monitoring Agent program.

## 14.4 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 the Monitoring Agent, 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 `xpm` Packages

### On this page

- Overview
- Considerations
- Prerequisites
- Procedures
- Next Steps
- Additional Information

**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 the Monitoring Agent, please refer to the [Install Monitoring Agent](#) documentation.

### Considerations

**MongoDB Requirements** Ops Manager only supports backing up replica sets and sharded clusters, 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 2.4.3 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. You can change the default window by adding the `mms.backup.minimumOplogWindowHours` setting to the *Custom* tab on the [Ops Manager Configuration page](#).

**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 Agent, 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 the Backup feature 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 *Settings* 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: Configure the Backup Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `/etc/mongodb-mms/backup-agent.config` file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 6: Optional: For SUSE 11+ deployments only, configure the `sslTrustedMMServerCertificate` property.** If you are deploying on SUSE, you must configure the `sslTrustedMMServerCertificate` setting. All other users should omit this step.

Enter the following property and value in the `/etc/mongodb-mms/backup-agent.config` file:

```
sslTrustedMMServerCertificate=/etc/ssl/certs/UTN_USERFirst_Hardware_Root_CA.pem
```

Save and close the file.

**Step 7: 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 <mmUri>/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 [Back up a Deployment](#) 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](#)
- [Considerations](#)
- [Prerequisites](#)
- [Procedures](#)
- [Next Steps](#)
- [Additional Information](#)

**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 the Monitoring Agent, please refer to the [Install Monitoring Agent](#) documentation.

### **Considerations**

**MongoDB Requirements** Ops Manager only supports backing up replica sets and sharded clusters, 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 2.4.3 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. You can change the default window by adding the `mms.backup.minimumOplogWindowHours` setting to the *Custom* tab on the [Ops Manager Configuration page](#).

**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 Agent, 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 the Backup feature 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 *Settings* tab on the *Agents* page, select your appropriate 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: Optional: Configure the Backup Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `/etc/mongodb-mms/backup-agent.config` file, set the `httpProxy` value to the URL of your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 6: 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 than 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 [Backup a Deployment](#) 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](#)
- [Considerations](#)
- [Prerequisites](#)
- [Procedures](#)
- [Next Steps](#)
- [Additional Information](#)

**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 the Monitoring Agent, please refer to the [Install Monitoring Agent](#) documentation.

## Considerations

**MongoDB Requirements** Ops Manager only supports backing up replica sets and sharded clusters, 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 2.4.3 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. You can change the default window by adding the `mms.backup.minimumOplogWindowHours` setting to the *Custom* tab on the [Ops Manager Configuration page](#).

**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 Agent, 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 the Backup feature 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 *Settings* 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 *Settings* tab on the *Agents* page, select your the appropriate 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 Backup Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/local.config` file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**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 *Settings* tab on the *Agents* page, select your the appropriate 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: Optional: Configure the Backup Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/local.config` file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**Step 7: 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 [Back up a Deployment](#) to enable backup for a replica set.

**Additional Information** If you installed the Backup Agent from the `.tar.gz` archives, see <http://docs.opsmanager.mongodb.com//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](#)
- [Considerations](#)
- [Prerequisites](#)
- [Procedures](#)
- [Next Steps](#)
- [Additional Information](#)

**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 the Monitoring Agent, please refer to the [Install Monitoring Agent](#) documentation.

## Considerations

**MongoDB Requirements** Ops Manager only supports backing up replica sets and sharded clusters, 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 2.4.3 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. You can change the default window by adding the `mms.backup.minimumOplogWindowHours` setting to the *Custom* tab on the [Ops Manager Configuration page](#).

**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 Agent, 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 the Backup feature 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 *Settings* 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 Backup Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/local.config` file, set the `httpProxy` value to the URL of to your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**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 *Settings* tab on the *Agents* page, select your appropriate 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 Backup Agent to use a proxy server.** To configure the agent to connect to Ops Manager via a proxy server, you must specify the server in the `httpProxy` environment variable. In the `<install-directory>/local.config` file, set the `httpProxy` value to the URL of your proxy server:

```
httpProxy="http://proxy.example.com:9000"
```

**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 *Back up a Deployment* 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](#)
- [Considerations](#)
- [Prerequisites](#)
- [Procedures](#)
- [Next Steps](#)
- [Additional Information](#)

**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 the Monitoring Agent, please refer to the [Install Monitoring Agent](#) documentation.

## Considerations

**MongoDB Requirements** Ops Manager only supports backing up replica sets and sharded clusters, 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 2.4.3 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. You can change the default window by adding the `mms.backup.minimumOplogWindowHours` setting to the *Custom* tab on the [Ops Manager Configuration page](#).

**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 Agent, 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 the Backup feature 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 *Settings* tab on the *Agents* page, select your the appropriate 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 MongoDB 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 MongoDB 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 *Settings* tab on the *Agents* page, select your the appropriate 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 MongoDB Backup Agent service. Select the Action menu and select Start.

**Next Steps** After you have successfully installed the Backup Agent, see *Back up a Deployment* 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

- Configuration File
- Settings

**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 *Settings* 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

Type: string

To connect to Ops Manager 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
```

**krb5ConfigLocation**

Type: string

The *absolute* path to an non-system-standard location for the Kerberos configuration file. For example:

```
krb5ConfigLocation=/etc/krb_custom.conf
```

**gssapiServiceName**

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 with Ops Manager.

**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

- Considerations
- MongoDB 2.6
- MongoDB 2.4
- [Authentication Mechanisms](#)

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.

**Important:** The Backup Agent user must be defined consistently for all processes in your Ops Manager deployment.

**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	
<code>backup</code> 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	
<code>clusterAdmin</code> role on the admin database	
<code>readAnyDatabase</code> role on the admin database	
<code>userAdminAnyDatabase</code> role on the admin database	
<code>readWrite</code> role on the admin database	
<code>readWrite</code> 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	
<code>clusterAdmin</code> role on the <code>admin</code> database	
<code>readAnyDatabase</code> role on the <code>admin</code> database	
<code>userAdminAnyDatabase</code> role on the <code>admin</code> database	

And the following otherDBRoles:

Required Role	
<code>readWrite</code> role on the <code>local</code> database	
<code>readWrite</code> role on the <code>admin</code> database	
<code>readWrite</code> role on the <code>config</code> 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 for x.509** Procedure to configure the Backup Agent for MongoDB deployments using x.509 Client Certificate authentication.

## Configure Backup Agent for MONGODB-CR

### On this page

- Procedures

In MongoDB 3.0 and later, MongoDB's default authentication mechanism is a challenge and response mechanism (SCRAM-SHA-1). Previously, MongoDB used MongoDB Challenge and Response (MONGODB-CR) as the default.

The Backup Agent can use MONGODB-CR or SCRAM-SHA-1 to authenticate to hosts that enforce access control.

To authenticate using SCRAM-SHA-1 or MONGODB-CR, create a user in the `admin` database with the appropriate roles in MongoDB.

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage agent authentication for you if you use Automation

to manage the agents. With Automation, Ops Manager creates the users for each agent and configures the agent appropriately. See: [Enable SCRAM-SHA-1 / MONGODB-CR Authentication for your Ops Manager Group](#) for more information.

---

## Procedures

**Create MongoDB User for the Agent** 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.

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 back up 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 back up 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 back up 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

- [Considerations](#)
- [Procedures](#)

If your MongoDB deployment enforces access control, the Backup Agent must authenticate to MongoDB as a user with the proper access.

Starting with version 2.6, [MongoDB Enterprise](#) for Linux provides support for proxy authentication of users. This allows administrators to configure a MongoDB cluster to authenticate users by proxying authentication requests to a specified Lightweight Directory Access Protocol (LDAP) service. Backup Agents support authenticating to MongoDB instances using LDAP.

MongoDB Enterprise for Windows does **not** include LDAP support for authentication.

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.

---

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage agent authentication for you if you use Automation to manage the agents. With Automation, Ops Manager creates the users for each agent and configures the agent appropriately. See: [Enable LDAP Authentication for your Ops Manager Group](#) for more information.

---

**Considerations** You must configure LDAP authentication separately for each agent. See [Configure Monitoring Agent for LDAP](#) for configuration instructions for the Monitoring Agent.

You can configure LDAP authentication when activating backup or by editing the an existing host's configuration. [Enable LDAP Authentication for your Ops Manager Group](#) for instructions.

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.

## Procedures

**Create User in MongoDB** To back up MongoDB 2.6+ instances that are using LDAP authentication, add a user that possess the required roles to the `$external` database in MongoDB. The `$external` database allows `mongod` to consult an external source, such as an LDAP server, to authenticate.

Use the following commands to create the users from a mongo shell connected to your MongoDB deployment:

### 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 [Required Access for Backup Agent](#) 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

- Prerequisites
- Procedures

*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.

---

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage agent authentication for you if you use Automation

to manage the agents. With Automation, Ops Manager creates the users for each agent and configures the agent appropriately. See: [Enable Kerberos Authentication for your Ops Manager Group](#) for more information.

---

**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.

## Procedures

### Create 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** 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.

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. If each agent is going to use its own Kerberos Principal, then you must create a user in the \$external database for each Kerberos Principal.

Use the following commands to create the users from a mongo shell connected to your MongoDB deployment:

#### MongoDB 3.0 or Later

```
use $external
db.createUser(
{
    user: "<Kerberos Principal>",
    roles: [
        { role: "backup", db: "admin" }
    ]
})
```

#### MongoDB 2.6

```
use $external
db.createUser(
{
    user: "<Kerberos Principal>",
    roles: [
        "clusterAdmin",
        "readAnyDatabase",
        "userAdminAnyDatabase",
        { role: "readWrite", db: "admin" },
        { role: "readWrite", db: "local" },
        { role: "readWrite", db: "config" }
    ]
})
```

```
        ]  
    }  
}
```

See [Required Access for Backup Agent](#) for more information on the required access.

If you are using the same Kerberos Principal for both the Monitoring and Backup Agents, the user must possess the required roles for both the backup agent, **and** the *monitoring 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/krb5.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 User for x.509 Client Certificate Authentication

### On this page

- Considerations
- Procedures

Ops Manager enables you to configure the Authentication Mechanisms that the Ops Manager Agents use to connect to your MongoDB deployments from within the Ops Manager interface. You can enable multiple authentication mechanisms for your group, but you must choose a single mechanism for the Agents to use to authenticate to your deployment.

MongoDB supports x.509 certificate authentication for use with a secure [TLS/SSL](#) connection. The x.509 client authentication allows clients to authenticate to servers with certificates rather than with a username and password.

In Ops Manager, x.509 Client Certificate (MONGODB-X509) is only available on MongoDB Enterprise builds. If you have existing deployments running on a MongoDB Community build, you must *upgrade them to MongoDB Enterprise* before you can enable x.509 Client Certificate (MONGODB-X509) for your Ops Manager group.

---

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage agent authentication for you if you use Automation to manage the agents. With Automation, Ops Manager creates the users for each agent and configures the agent appropriately. See: *Enable x.509 Authentication for your Ops Manager Group* for more information.

---

## Considerations

**Important:** A full description of TLS/SSL, PKI (Public Key Infrastructure) certificates, in particular x.509 certificates, and Certificate Authority is beyond the scope of this document. This tutorial assumes prior knowledge of TLS/SSL as well as access to valid x.509 certificates.

In order to enable x.509 Authentication for Ops Manager, you must obtain valid certificates generated and signed by a single certificate authority. Refer to the [Client x.509 Certificate](#) in the MongoDB Manual for more about the certificate requirements.

---

**Important:** x.509 Client Certificate Authentication requires that SSL be enabled and configured for the deployment.

---

**Procedures** This tutorial assumes that you have already configured your MongoDB deployment to use x.509 certificate authentication and SSL. If you have not done so, refer to the [Use x.509 Certificates to Authenticate Clients](#) and [Configure mongod and mognos for TLS/SSL](#) tutorials.

**Create MongoDB User for the subject** In order for the Backup Agent to connect to your MongoDB deployment, you must create a user for the Monitoring Agent in the \$external database.

For x.509 certificate authentication, use the `subject` value of your client certificate as the username.

Use the following commands to create the users from a mongo shell connected to your MongoDB deployment:

### MongoDB 3.0 or Later

```
use $external
db.createUser(
{
  user: "<x.509 subject>",
  roles: [
    { role: "backup", db: "admin" }
  ]
})
```

### MongoDB 2.6

```
use $external
db.createUser(
{
  user: "<x.509 subject>",
  roles: [
    "clusterAdmin",
    "readAnyDatabase",
    "userAdminAnyDatabase",
  ]
})
```

```

        { role: "readWrite", db: "admin" },
        { role: "readWrite", db: "local" },
    ]
}
)

```

See [Required Access for Backup Agent](#) for more information on the required access.

You can only associate an x.509 client certificate with a single user: each user must have its own certificate. However, you may use the same user for both the Backup and Monitoring agents. If you choose to use the same user for both agents, ensure that the user possesses the required permissions for both the backup agent **and** the [monitoring agent](#).

**Edit Agent Configuration File** x.509 requires that you configure the agent for 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 the agent.** [Configure Monitoring Agent for SSL](#) provides more details about configuring the Monitoring Agent for SSL.

Once you have configured the Backup agent, you still need to configure the x.509 Authentication mechanism in the Ops Manager interface, as in [Enable x.509 Authentication for your Ops Manager Group](#).

## Configure Backup Agent for SSL

### On this page

- Overview
- Prerequisite
- Procedures

## 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](#).

---

**Note:** In Ops Manager 1.8 and later, Ops Manager can manage SSL for you if you use Automation for the deployment. With Automation, Ops Manager prompts you for the certificates to use to connect to the deployment when you enable SSL and then configures the agents appropriately. See: [Enable SSL for a Deployment](#) for more information.

**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](#)
- [Procedures](#)

### Overview

For maintenance or troubleshooting purposes, you may want to temporarily shut down or restart the 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 MongoDB 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 MongoDB 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 the *Settings* 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 lmmsl**, *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 MongoDB Backup Agent program.

## 14.5 Database Commands Used by Monitoring Agent

The Monitoring Agent uses a set of MongoDB diagnostic, administrative, and other database commands to report on the status of your MongoDB deployment. The agent uses these commands:

- `_isSelf`
- `buildInfo`
- `collStats` on the following local database collections, which support replication operations:
  - `local.oplog.rs`
  - `local.oplog.$main`. Only for master-slave replication
- `connPoolStats`
- `dbStats`
- find on the following config database collections, which support sharding operations:
  - `config.chunks`
  - `config.collections`
  - `config.databases`
  - `config.lockpings`
  - `config.mongos`
  - `config.settings`
  - `config.shards`
- find on the following local database collections, which support replication operations:
  - `local.system.replset`
  - `local.sources`. Only for master-slave replication
- find on the `system.profile` collection. Only if *database profiling* is enabled. Database profiling is disabled by default.
- `findOne` on the `local.oplog.rs` collection. Only for MongoDB v2.6 and earlier.
- `getCmdLineOpts`
- `getLog` issued with:
  - A value of `global`. Only if you have enabled *log collection*.
  - A value of `startupWarnings`.
- `getParameter` issued with a value of `*`
- `getShardVersion`
- `hostInfo`
- `isMaster`
- `listDatabases`
- `netstat`
- `ping`
- `profile`. Only if *database profiling* is enabled. Database profiling is disabled by default.

- replSetGetStatus
- serverStatus

## 14.6 Audit Events

### On this page

- User Audits
- Host Audits
- Alert Config Audits
- Backup Audits
- Group Audits

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. You can access the *Events* page through the *Admin* link.

### 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.

**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 deleted 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 namespaces filter for a replica set.

### **CLUSTER\_BLACKLIST\_UPDATED\_AUDIT**

A user updates the namespaces filter 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\_CHECKPOINT\_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.

## 14.7 MongoDB Compatibility

### On this page

- Automation and MongoDB
- Monitoring and MongoDB
- Backup and MongoDB
- MongoDB Deployment Types

This page describes compatibility between Ops Manager features and MongoDB.

---

### MongoDB 2.4 reached its End of Life date in March 2016

- Ops Manager will remove support for Automation and Backup of MongoDB 2.4 in Ops Manager 3.6 (estimated date: late 2017). Ops Manager groups must upgrade their MongoDB 2.4 deployments before the Ops Manager 3.6 release. Starting in Ops Manager 3.6, Automation will no longer support upgrades from MongoDB 2.4.
  - Ops Manager will continue to support Monitoring of MongoDB 2.4 deployments.
- 

### Automation and MongoDB

Automation supports MongoDB 2.4 and later.

Automation has these additional compatibility requirements:

To deploy:	You must run:
MongoDB 3.2 or later	Ops Manager 2.0 or later
MongoDB 3.0 or later	Ops Manager 1.6 or later

### Monitoring and MongoDB

Monitoring supports MongoDB 2.4 or later. However, to monitor MongoDB 3.0 or later, you must use Monitoring Agent version 2.7.0 or later.

Monitoring has these additional compatibility requirements:

To monitor:	You must run:
MongoDB 3.2 or later	Ops Manager 2.0 or later
MongoDB 3.0 or later	Ops Manager 1.6 or later

## Backup and MongoDB

Ops Manager can back up:

- MongoDB 2.4.3 or later for sharded clusters.
- MongoDB 2.0 or later for replica sets.

Backup has these additional compatibility requirements:

To back up:	You must run:
MongoDB 3.2 or later	Ops Manager 2.0 or later
MongoDB 3.0 or later	Ops Manager 1.6 or later

## MongoDB Deployment Types

You can configure all MongoDB deployment types: sharded clusters, replica sets, and standalones.

The shards in a sharded cluster **must** be replica sets. That is, a shard cannot be a standalone mongod. If you must run a shard as a single mongod (which provides **no** redundancy or failover), run the shard as a single-member replica set.

## 14.8 Supported Browsers

To use Ops Manager, ensure that your browser is one of the following supported browsers, with Javascript enabled:

- Chrome latest stable
- Firefox latest stable
- Internet Explorer 10 (IE10) and greater
- Safari latest stable version on MacOS 10.10.5+

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

## 14.9 Advanced Options for MongoDB Deployments

### On this page

- [Overview](#)
- [Advanced Options](#)

### 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 and parameters described here.

## **auditLog**

- *auditLogDestination*: auditLog.destination
- *auditLogFormat*: auditLog.format
- *auditLogPath*: auditLog.path
- *auditLogFilter*: auditLog.filter

## **basisTech**

- *rootDirectory*: basisTech.rootDirectory

## **net**

- *bind\_ip*: net.bindIp  
If you set this, you must include 127.0.0.1 as well. For example: “198.51.100.0,127.0.0.1“
- *maxConns*: net.maxIncomingConnections

## **net.http**

- *nohttpinterface*: net.http.enabled
- *jsonp*: net.http.JSONPEnabled
- *rest*: net.http.RESTInterfaceEnabled

## **net.ssl**

- *sslOnNormalPorts*: net.ssl.sslOnNormalPorts
- *sslPEMKeyFile*: net.ssl.PEMKeyFile
- *sslPEMKeyPassword*: net.ssl.PEMKeyPassword
- *sslMode*: net.ssl.mode

## **operationProfiling**

- *slowms*: operationProfiling.slowOpThresholdMs
- *profile*: operationProfiling.mode

## **processManagement**

- *pidfilepath*: processManagement.pidFilePath

## replication

- *oplogSize*: replication.oplogSizeMB
- *enableMajorityReadConcern*: replication.enableMajorityReadConcern

## security

- *noscripting*: noscripting
- *clusterAuthMode*: security.clusterAuthMode
- *enableEncryption*: security.enableEncryption
- *encryptionCipherMode*: security.encryptionCipherMode
- *encryptionKeyFile*: security.encryptionKeyFile

## security.kmip

- *keyIdentifier*: security.kmip.keyIdentifier
- *serverName*: security.kmip.serverName
- *kmipPort*: security.kmip.port
- *clientCertificateFile*: security.kmip.clientCertificateFile
- *clientCertificatePassword*: security.kmip.clientCertificatePassword
- *serverCAFile*: security.kmip.serverCAFile

## setParameter

- *connPoolMaxConnectionsPerHost*: connPoolMaxConnectionsPerHost
- *connPoolMaxShardedConnsPerHost*: connPoolMaxShardedConnectionsPerHost
- *wiredTigerConcurrentReadTransactions*: wiredTigerConcurrentReadTransactions
- *wiredTigerConcurrentWriteTransactions*: wiredTigerConcurrentWriteTransactions
- *releaseConnectionsAfterResponse*: releaseConnectionsAfterResponse
- *newCollectionsUsePowerOf2Sizes*: newCollectionsUsePowerOf2Sizes
- *enablelocalhostAuthBypass*: enablelocalhostAuthBypass
- *enableTestCommands*: enableTestCommands
- *ttlMonitorEnabled*: ttlMonitorEnabled
- *failIndexKeyTooLong*: failIndexKeyTooLong
- *logLevel*: logLevel
- *[other]*: specify a **custom option** and its value. You cannot use the other field to specify an option that is available in the drop-down menu.

## sharding

- *moveParanoia*: sharding.archiveMovedChunks
- *noAutoSplit*: sharding.autoSplit set to false

## storage

- *syncdelay*: storage.syncPeriodSecs
- *noprealloc*: storage.mmapv1.preallocDataFiles
- *smallfiles*: storage.mmapv1.smallFiles
- *nssize*: storage.mmapv1.nsSize
- *quota*: quota
- *quotaFiles*: quotaFiles
- *directoryperdb*: storage.directoryPerDB
- *engine*: storage.engine

## storage.journal

- *nojournal*: storage.journal.enabled
- *journalCommitInterval*: storage.mmapv1.journal.commitIntervalMs

## storage.wiredTiger.collectionConfig

- *blockCompressor*: storage.wiredTiger.collectionConfig.blockCompressor

## storage.wiredTiger.engineConfig

- *cacheSizeGB*: storage.wiredTiger.engineConfig.cacheSizeGB
- *checkpointDelaySecs*: storage.syncPeriodSecs
- *journalCompressor*: storage.wiredTiger.engineConfig.journalCompressor
- *directoryForIndexes*: storage.wiredTiger.engineConfig.directoryForIndexes
- *statisticsLogDelaySecs*: storage.wiredTiger.engineConfig.statisticsLogDelaySecs

## storage.wiredTiger.indexConfig

- *prefixCompression*: storage.wiredTiger.indexConfig.prefixCompression

## systemLog

- *logappend*: `systemLog.logAppend`:
- *quiet*: `systemLog.quiet`:
- *syslog*: `systemLog.destination`:
- *logTimestampFormat*: `systemLog.timeStampFormat`
- *logRotate*: `systemLog.logRotate`
- *verbosity*: `systemLog.verbosity`

## 14.10 Automation Configuration

### On this page

- Overview
- Configuration Version
- Download Base
- MongoDB Versions Specifications
- Automation Agent
- Monitoring Agent
- Backup Agent
- MongoDB Instances
- Replica Sets
- Sharded Clusters
- Cluster Balancer
- Authentication
- SSL
- MongoDB Roles
- Kerberos
- Indexes

### 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 manually.

Optional fields are marked as such.

A field that takes a <number> as its value can take integers and floating point numbers.

### Configuration Version

This lists the version of the automation configuration.

```
"version" : <integer>
```

Name	Type	Description
version	integer	The version of the configuration.

## Download Base

The download base is the path to the directory where automatic version downloads will be targeted and scripts for starting processes will be created.

```
"options" : {
    "downloadBase" : <string>,
    "downloadBaseWindows" : <string>
}
```

Name	Type	Description
options	object	The options object is required and must contain both the downloadBase and downloadBaseWindows fields.
options.downloadBase	string	The directory on Linux and Unix (including Mac OS X) platforms for automatic version downloads and startup scripts.
options.downloadBaseWindows	string	The directory on Windows platforms for automatic version downloads and startup scripts.

## MongoDB Versions Specifications

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>,
                "modules" : [ <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.
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.
mongoDbVersions.builds	array of objects	Objects that define the builds for this MongoDB instance.
mongoDbVersions.buildPlatform	string	The platform for this MongoDB instance.
mongoDbVersions.buildUrl	string	The URL from which to download MongoDB for this instance.
mongoDbVersions.buildCommit	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.
mongoDbVersions.buildModules	array	The list of modules for this version. Corresponds to the modules field returned by MongoDB 3.2+ buildInfo command.
mongoDbVersions.buildProcessorWidth	integer	The processor's bus width. Specify either 64 or 32.
mongoDbVersions.buildWindowsOptional	Boolean	Set to true if this is a Windows build that requires either Windows 7 later or Windows Server 2008 R2 or later.
mongoDbVersions.buildVisualCplusOptional	string	The URL from which the required version of the Microsoft Visual C++ redistributable can be downloaded.
mongoDbVersions.buildVisualCplusOptionalValues	array	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.
mongoDbVersions.buildVisualCplusOptionalName	string	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.
mongoDbVersions.buildVisualCplusOptionalMinVersion	string	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.

## Automation Agent

The agentVersion object is optional and specifies the version of Automation Agent.

```
"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.
agentVersion.name	string	The desired version of the Automation Agent (e.g. “1.8.1.1042-1”).
agentVersion.directoryUrl	string	The URL from which to download Automation Agent.

## Monitoring Agent

The monitoringVersions array is optional and specifies the version of the Monitoring Agent.

```
"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	<i>Optional.</i> Objects that define version information for each Monitoring Agent.
monitoringVersions.name	string	The desired version of the Monitoring Agent (e.g. “2.9.1.176-1”). For MongoDB compatibility with Automation, see <a href="#">MongoDB Compatibility</a> .
monitoringVersions.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.
monitoringVersions.urls	object	The platform- and build-specific URLs from which to download the Monitoring Agent.
monitoringVersions.urlplat	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.
monitoringVersions.baseUrl	string	The base URL used for the mmsBaseUrl setting in the <a href="#">Monitoring Agent Configuration</a> .
monitoringVersions.logPath	string	<i>Optional.</i> The directory where the agent stores its logs. The default is to store logs in /dev/null.
monitoringVersions.logRotate	object	<i>Optional.</i> Enables log rotation for the MongoDB logs for a process.
monitoringVersions.logRotate.sizeMB	number	The maximum size in MB for an individual log file before rotation.
monitoringVersions.logRotate.timeInHours	number	The maximum time in hours for an individual log file before rotation.
monitoringVersions.logRotate.maxCompressed	number	<i>Optional.</i> The maximum number of total log files to leave uncompressed, including the current log file. The default is 5. In earlier versions of Ops Manager, this field was named maxUncompressed. The earlier name is still recognized, though the new version is preferred.
monitoringVersions.logRotate.percent	number	<i>Optional.</i> The maximum percentage of total disk space all log files should take up before deletion. The default is .02.

## Backup Agent

The backupVersions array is optional and specifies the version of the Backup Agent.

```
"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	<i>Optional.</i> Objects that define version information for each Backup Agent.
backupVersions.name	string	The desired version of the Backup Agent (e.g. “3.1.1.263-1”).
backupVersions.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.
backupVersions.urls	object	The platform- and build-specific URLs from which to download the Backup Agent.
backupVersions.urls.os	object for each operating system	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.
backupVersions.baseUrl	string	The base URL used for the mothership and https settings in the <a href="#">Backup Agent Configuration</a> . For example, for baseUrl="https://cloud.mongodb.com, the backup configuration fields would have these values:
backupVersions.logPath	string	<i>Optional.</i> The directory where the agent stores its logs. The default is to store logs in /dev/null.
backupVersions.logRotater	object	<i>Optional.</i> Enables log rotation for the MongoDB logs for a process.
backupVersions.logRotater.size	number	The maximum size in MB for an individual log file before rotation.
backupVersions.logRotater.time	number	The maximum time in hours for an individual log file before rotation.
backupVersions.logRotater.number	number	<i>Optional.</i> The maximum number of total log files to leave uncompressed, including the current log file. The default is 5.
backupVersions.logRotater.percentage	number	<i>Optional.</i> The maximum percentage of total disk space all log files should take up before deletion. The default is .02.

## MongoDB Instances

The processes array determines the configuration of your MongoDB instances. You can also use the array to restore an instance.

```

"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>,
"backupRestoreUrl" : <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.
processes.name	string	A unique name to identify the instance.
processes.processType	string	Either mongod or mongos.
processes.version	string	The name of the mongoDbVersions specification used with this instance.
processes.<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 <a href="#">supported configuration options</a> .
processes.disabled	Boolean	<i>Optional.</i> Set to true to shut down the process.
processes.manualMode	Boolean	<i>Optional.</i> Set to true to operate this process in manual mode. The Automation Agent will take no actions on the process.
processes.hostname	string	<i>Optional.</i> The name of the host this process should run on. This defaults to localhost.
processes.cluster	string	<i>Optional.</i> Required for a mongos. The name of the cluster. This must correspond to the sharding.name field in the sharding array for the mongos.
processes.numCores	integer	<i>Optional.</i> The number of cores the process should be bound to. The Automation Agent will spread processes out across the cores as evenly as possible.
processes.logRotate	object	<i>Optional.</i> Enables log rotation for the MongoDB logs for a process.
processes.logRotate.sizeThresholdMB	integer	The maximum size in MB for an individual log file before rotation. The file rotates immediately if the file meets either this sizeThresholdMB or the processes.logRotate.timeThresholdHrs limit.
processes.logRotate.timeThresholdHrs	integer	The maximum runtime in hours for an individual log file before the next rotation. The time is since the last rotation. The log file rotates immediately if the file meets either this timeThresholdHrs or the processes.logRotate.sizeThresholdMB limit.
processes.logRotate.uncompressedLogCount	integer	<i>Optional.</i> The maximum number of total log files to leave uncompressed, including the current log file. The default is 5.
processes.logRotate.uncompressedLogPercentage	number	<i>Optional.</i> The maximum percentage of total disk space that can be used to store the log files. If this limit is exceeded, the compressed log files are deleted to meet this limit, starting with the oldest log files first. The default is .02.
processes.authSchemaVersion	integer	<i>Optional.</i> 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.
processes.alias	string	<i>Optional.</i> 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.
processes.backupRestoreUrl	string	<i>Optional.</i> This is used only when creating a restore and specifies the delivery url for the restore. See <a href="#">Automate Backup Restoration through the API</a> .

## Replica Sets

The `replicaSets` array is optional and defines each replica set's configuration.

```
"replicaSets" : [
    {
        "_id" : <string>,
        "version" : <integer>
        "members" : [
            {
                "_id" : <integer>,
                "host" : <string>
            },
            ...
        ],
        "force" : {
            "currentVersion" : <integer>
        }
    },
    ...
]
```

Name	Type	Description
replicaSets	array of objects	<i>Optional.</i> Objects that define the configuration of each <a href="#">replica set</a> . The Automation Agent uses the values in this array to create valid <a href="#">replica set configuration documents</a> . 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: <code>_id</code> ; <code>version</code> ; and <code>members</code> . For more information on the configuration documents, see <a href="#">replicaSetGetConfig</a> in the MongoDB manual.
replicaSets	string	The name of the replica set.
replicaSets	integer	The version of the replica set configuration.
replicaSets	array of objects	Objects that define each member of the replica set. The <code>members.host</code> field must specify the host's name as listed in <code>processes.name</code> . The Automation Agent expands the <code>host</code> field to create a valid replica set configuration. For more information on <code>members</code> objects, see <a href="#">replicaSetGetConfig</a> in the MongoDB manual.
replicaSets	object	<i>Optional.</i> An object that contains the <code>currentVersion</code> field set to a version number. Automation will force a reconfiguration of the replica set if and only if the value of <code>currentVersion</code> equals the current version of the replica set. You can use <code>force</code> to reconfigure a replica set that has lost members and can't reach a majority of votes.

## Sharded Clusters

The sharding array is optional and defines the configuration of each sharded cluster.

```
"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	<i>Optional.</i> Objects that define the configuration of each <i>sharded cluster</i> . 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.
sharding.name	string	The name of the cluster. This must correspond with the value in processes.cluster for a mongos.
sharding.configServer	array of servers	String values that provide the names of each <i>config server</i> 's hosts. The host names are the same names as are used in each host's processes.name field.
sharding.collections	array of objects	Objects that define the sharded <i>collections</i> and their <i>shard keys</i> .
sharding.collections	string	The <i>namespace</i> of the sharded collection. The namespace is the combination of the database name and the name of the collection. For example, testdb.testcoll.
sharding.collections	array of arrays	The collection's <i>shard keys</i> . 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.
sharding.shards	array of objects	Objects that define the cluster's <i>shards</i> .
sharding.shard	string	The name of the shard.
sharding.shard	string	The name of the shard's replica set, as specified in the replicaSets._id field.

## Cluster Balancer

The balancer object is optional and defines balancer settings for each cluster.

```

"balancer": {
    "<clusterName1>": <object>,
    "<clusterName2>": <object>,
    ...
}

```

Name	Type	Description
balanceObject	object	<i>Optional.</i> 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 <a href="#">in this tutorial</a> in the MongoDB manual.

## Authentication

The auth object is optional and defines authentication-related settings.

```
"auth" : {  
    "autoUser": <string>,  
    "autoPwd": <string>,  
    "disabled": <Boolean>,  
    "deploymentAuthMechanisms": [ <string>, <string>, ... ],  
    "key" : <string>,  
    "keyfile" : <string>,  
    "usersDeleted" : [  
        {  
            "user" : <string>,  
            "dbs" : [ <string>, ... ]  
        }  
    ],  
    "usersWanted" : [  
        {  
            "db" : <string>,  
            "user" : <string>,  
            "roles" : [ <string>, ... ],  
            "pwd" : <32-character hex string>,  
            "initPwd" : <string>,  
            "userSource" : <string>,  
            "otherDBRoles" : {  
                <string> : [ <string>, ... ]  
            }  
        }  
    ]  
}
```

Name	Type	Description
auth	object	<i>Optional.</i> Defines authentication-related settings.
auth.autoUser	string	The username that the Automation agent uses when connecting to an instance.
auth.autoPwd	string	The password that the Automation agent uses when connecting to an instance.
auth.disabled	Boolean	Specifies whether authentication is enabled or disabled. Set to <code>true</code> to disable authentication, or <code>false</code> to enable authentication.
auth.deploymentMechanisms	array	List the supported authentication mechanisms for the processes in the deployment. Specify MONGODB-CR for MONGODB-CR / SCRAM-SHA-1 authentication, MONGODB-X509 for x.509 Client Certificate authentication, PLAIN for LDAP authentication, and GSSAPI for authenticating with Kerberos.
auth.disabled	boolean	<i>Optional.</i> Indicates if auth is disabled. If not specified, disabled defaults to <code>false</code> .
auth.key	string	The contents of the key file that Ops Manager uses to authenticate to the MongoDB processes. The <code>key</code> is not required if <code>disabled</code> is <code>true</code> .
auth.keyfile	string	The path and name of the key file that Ops Manager uses to authenticate to the MongoDB processes. The <code>keyfile</code> is not required if <code>disabled</code> is <code>true</code> .
auth.usersDeleted	array of objects	<i>Optional.</i> Objects that define the authenticated users to be deleted from specified databases or from all databases. This array must contain two fields: the <code>auth.usersDeleted.user</code> field and the <code>auth.usersDeleted.dbs</code> field.
auth.usersDeleted.name	string	The user's name.
auth.usersDeleted.dbs	array	String values that list the names of the databases from which the authenticated user is to be deleted.
auth.usersWanted	array of objects	<i>Optional.</i> Contains objects that define authenticated users to add to specified databases. Each object must have the <code>auth.usersWanted.db</code> , <code>auth.usersWanted.user</code> , and <code>auth.usersWanted.roles</code> fields, and then have exactly one of the following fields: <code>auth.usersWanted.pwd</code> , <code>auth.usersWanted.initPwd</code> , or <code>auth.usersWanted.userSource</code> .
auth.usersWanted.db	string	The database to which to add the user.
auth.usersWanted.user	string	The name of the user.
auth.usersWanted.roles	array	String values that list the <code>roles</code> to be assigned the user from the user's database, which is specified in <code>auth.usersWanted.db</code> .
auth.usersWanted.pwd	character hex string	The <b>MONGODB-CR</b> hash of the password assigned to the user. If you set this field, <b>do not</b> set the <code>auth.usersWanted.initPwd</code> or <code>auth.usersWanted.userSource</code> fields.
auth.usersWanted.initPwd	string	An initial cleartext password assigned to the user. If you set this field, <b>do not</b> set the <code>auth.usersWanted.pwd</code> or <code>auth.usersWanted.userSource</code> fields.
auth.usersWanted.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 <a href="#">Privilege Documents page in the MongoDB 2.4 manual</a> . If you set this field, <b>do not</b> set the <code>auth.usersWanted.pwd</code> or <code>auth.usersWanted.initPwd</code> fields.
auth.usersWanted.otherDB	object	<i>Optional.</i> If the <code>auth.usersWanted.db</code> field specifies <code>admin</code> 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

The `ssl` object is optional and enables SSL for encrypting connections.

```
"ssl" : {  
    "CAFilePath" : <string>  
}
```

Name	Type	Description
<code>ssl</code>	object	<i>Optional.</i> Enables SSL for encrypting connections. To use SSL, be sure to choose a package that supports SSL. Starting in MongoDB 3.0, most MongoDB distributions now include support for SSL. All <a href="#">MongoDB Enterprise</a> supported platforms include SSL support.

## MongoDB Roles

The `roles` array is optional and describes user-defined roles.

```
"roles" : [  
    {  
        "role" : <string>,  
        "db" : <string>,  
        "privileges" : [  
            {  
                "resource" : { ... },  
                "actions" : [ <string>, ... ]  
            },  
            ...  
        ],  
        "roles" : [  
            {  
                "role" : <string>,  
                "db" : <string>  
            }  
        ]  
    },  
    ...  
]
```

Name	Type	Description
<code>roles</code>	array of objects	<i>Optional.</i> The <code>roles</code> 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 <code>:manual:` system roles collection &lt;/reference/system-roles-collection&gt;</code> , except for the <code>_id</code> field, which is not included here.

## Kerberos

The `kerberos` object is optional and defines a kerberos service name used in authentication.

```
"kerberos": {  
    "serviceName": <string>  
}
```

Name	Type	Description
kerberos	object	<i>Optional.</i> A key-value pair that defines the kerberos service name agents use to authenticate via kerberos.
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 <a href="#">MongoDB Server Parameters</a> page in the MongoDB manual.

## Indexes

The indexConfigs array is optional and defines indexes to be built for specific replica sets.

```
"indexConfigs" : [
  {
    "key" : [
      [ <string> : <val> ],
      ...
    ],
    "rsName" : <string>,
    "dbName" : <string>,
    "collectionName" : <string>
  },
  ...
]
```

Name	Type	Description
indexConfigs	array of objects	<i>Optional.</i> Objects that define specific indexes to be built for specific replica sets.
indexConfigs.key	array of arrays	The index's keys. This “array of arrays” contains a single array if the index has just one key.
indexConfigs.rsName	string	The replica set that the index is build on.
indexConfigs.dbName	string	The database the index applies to.
indexConfigs.collectionName	string	The collection the index applies to.

## 14.11 Supported MongoDB Options for Automation

### On this page

- [Overview](#)
- [MongoDB 2.6 and Later Configuration Options](#)
- [MongoDB 2.4 and Earlier Configuration Options](#)

### Overview

The processes.<args> object in an [automation configuration file](#) specifies the configuration options for each MongoDB instance. The supported settings 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 and supports the following MongoDB settings and parameters. The object uses the [MongoDB configuration format](#).

The processes.args2\_6 object supports the following:

- auditLog.destination
- auditLog.format
- auditLog.path
- auditLog.filter
- basisTech.rootDirectory
- cpu
- diaglog
- net.bindIp

If you set `net.bindIp`, you must include 127.0.0.1 as well.

---

### **Example**

```
net:  
    bindIp: 198.51.100.0,127.0.01
```

---

- net.http.JSONPEnabled
- net.http.RESTInterfaceEnabled
- net.http.enabled
- net.maxIncomingConnections
- net.port
- net.ssl.clusterFile
- net.ssl.clusterPassword
- net.ssl.disabledProtocols
- noscripting
- notableScan
- operationProfiling.mode
- operationProfiling.slowOpThresholdMs
- processManagement.fork
- processManagement.pidFilePath
- replication.enableMajorityReadConcern
- replication.oplogSizeMB
- replication.replSet
- replication.replSetName
- replication.secondaryIndexPrefetch
- security.authorization
- security.clusterAuthMode
- security.keyFile
- security.enableEncryption
- security.encryptionCipherMode

- security.encryptionKeyFile
- security.kmip.keyIdentifier
- security.kmip.serverName
- security.kmip.port
- security.kmip.clientCertificateFile
- security.kmip.clientCertificatePassword
- security.kmip.serverCAFfile
- setParameter.connPoolMaxConnsPerHost
- setParameter.connPoolMaxShardedConnsPerHost
- setParameter.enableTestCommands
- setParameter.enablelocalhostAuthBypass
- setParameter.failIndexKeyTooLong
- setParameter.logLevel
- setParameter.newCollectionsUsePowerOf2Sizes
- setParameter.releaseConnectionsAfterResponse
- setParameter.textSearchEnabled
- setParameter.ttlMonitorEnabled
- sharding.clusterRole
- sharding.configDB
- storage.dbPath
- storage.directoryPerDB
- storage.engine
- storage.indexBuildRetry
- storage.repairPath
- storage.journal.commitIntervalMs
- storage.journal.enabled
- storage.mmapv1.journal.debugFlags
- storage.mmapv1.journal.commitIntervalMs
- storage.mmapv1.preallocDataFiles
- storage.mmapv1.nsSize
- storage.nsSize
- storage.preallocDataFiles
- storage.quota.maxFilesPerDB
- storage.quota.enforced
- storage.smallFiles
- storage.syncPeriodSecs

- `storage.wiredTiger.collectionConfig.blockCompressor`
- `storage.wiredTiger.engineConfig.cacheSizeGB`
- `storage.wiredTiger.engineConfig.directoryForIndexes`
- `storage.wiredTiger.engineConfig.journalCompressor`
- `storage.wiredTiger.indexConfig.prefixCompression`
- `systemLog.destination`
- `systemLog.logAppend`
- `systemLog.path`
- `systemLog.quiet`
- `systemLog.timeStampFormat`
- `systemLog.verbosity`

Ops Manager 1.8.0 adds the ability to specify a **custom option** using the `setParameter.[other]` option.

With `setParameter.[other]`, you specify both the parameter, and its value, as in the following MongoDB configuration document:

```
systemLog:
  destination: file
  path: "/var/log/mongodb/mongodb.log"
  logAppend: true
setParameter:
  enableLocalhostAuthBypass: false
  customParameter: "customValue"
...
```

## 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`
- `configsrv`
- `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
- vvv

## 14.12 SNMP Traps and Ops Manager Severities

This page explains how severe specific SNMP trap alerts are.

The severities are:

1. debug
2. info
3. warning
4. error
5. critical

The SNMP Alerts have the following severities in the SNMP traps:

Alert	Severity
<ul style="list-style-type: none"><li>• Host Down</li><li>• Monitoring Agent Down</li><li>• Backup Agent Down</li></ul>	critical (5)
<ul style="list-style-type: none"><li>• Host Recovering</li><li>• Host Metric outside of configured threshold</li></ul>	warning (3)
<p><b>Example</b> Connections exceeded user-specified threshold of 1000.</p>	
<ul style="list-style-type: none"><li>• Any informational alert</li></ul>	info (2)
<p><b>Example</b></p> <ul style="list-style-type: none"><li>– host is now primary</li><li>– host is now secondary</li><li>– host restarted</li></ul>	
<ul style="list-style-type: none"><li>• Host version outdated</li><li>• Monitoring Agent version outdated</li><li>• Backup Agent version outdated</li></ul>	
Any other alert not explicitly mentioned in this table.	warning (3)

## 14.13 Glossary

**agent** One of several lightweight programs that run within your network to monitor, manage, and back up your MongoDB databases.

See [Automation Agent](#), [Monitoring Agent](#) and [Backup Agent](#).

**agent API key** A unique identifier that authenticates a group's agents to Ops Manager. Each group has one agent API key.

**Ops Manager Application** The main Ops Manager component. The Ops Manager Application provides the user interface for managing MongoDB deployments and provides endpoints for Ops Manager [agents](#) to transmit data.

See [Ops Manager Application](#).

**Ops Manager Application Database** The dedicated MongoDB database that stores metadata for the Ops Manager installation and the managed MongoDB deployments.

See [Ops Manager Application Database](#).

**automation** The assisted management of MongoDB processes through the Ops Manager interface. [Automation Agents](#) installed on your MongoDB servers allow you to deploy, configure, and update MongoDB processes directly from Ops Manager.

See [Automation](#).

**Automation Agent** A lightweight component that automates common management tasks. The Automation Agent runs on every server that will have a mongod or mongos.

See [Automation Agent](#).

**Backup Agent** A lightweight component that runs within your data center and backs up MongoDB processes via the MongoDB wire protocol. No direct file system access is needed.

See [Backup Agent](#).

**Backup Blockstore Database** The database that stores your [snapshots](#). The database is also referred to simply as the *blockstore*. The blockstore uses a storage format that parses a snapshot into smaller chunks that allow Ops Manager to manage snapshot changes incrementally. You can administer blockstores from the [Snapshot Storage Page](#). The blockstore is one type of [Backup Database](#).

**Backup Daemon** The Ops Manager component that creates and manages backups by maintaining [head databases](#) and [snapshots](#).

See [Backup Daemon Service](#).

**Backup Database** The set of databases where Ops Manager stores backup data. This includes the [Olog Store Database](#) and includes the [Backup Blockstore Database](#) if used.

See [Backup Data Storage](#) and [Backup Flows](#).

**backup job** A process run by the [Backup Daemon](#) to apply the most recent changes to its backup of a [replica set](#). The daemon stores backups locally as [head databases](#). A sharded cluster will have a different head database for each shard. You can re-assign backup jobs among Backup Daemons.

See [Jobs Page](#).

**blockstore** See [Backup Blockstore Database](#).

**checkpoint** A point in time between snapshots to which you can restore a sharded cluster. Ops Manager must stop the [balancer](#) each time it creates a checkpoint. Ops Manager does not require checkpoints, and they are disabled by default.

See [Checkpoints](#).

**cluster** In Ops Manager, *cluster* can refer to either a [replica set](#) or [sharded cluster](#).

**custom snapshot** A backup of the state of your MongoDB deployment at a point in time between [stored snapshots](#). Ops Manager builds a custom snapshot by applying oplog data to a stored snapshot.

See [Restore Overview](#).

**deployment** *Deployment* usually refers to all the MongoDB processes that run within an Ops Manager [group](#). Deployment can also refer to a specific set of MongoDB processes, such as a specific [sharded cluster](#) or [replica set](#).

**excluded namespace** A database or collection that Ops Manager will not back up, as designated by its [namespace](#).

See [Namespaces Filter](#).

**File System Store** A directory on a server that stores your database backup [snapshots](#) as files. You can administer file system storage from the [Snapshot Storage Page](#).

**groom** A job that removes unused blocks on a [blockstore](#) and that can move blocks from one blockstore to another. You can view and manage grooms from [Grooms Page](#) and [Groom Priority](#).

**group** A distinct set of MongoDB processes and Ops Manager users. Each Ops Manager group must have a globally unique name within Ops Manager.

See [Create a Group](#).

**head** See [head database](#).

**head database** The copy of a backed-up deployment stored on the *Backup Daemon’s* server. The daemon maintains a head database for each *shard* or *replica set* it backs up and creates periodic *snapshots*. The daemon stores the head databases in the *head directory*.

**head directory** The dedicated disk partition on the *Backup Daemon’s* server where the Backup Daemon stores the *head databases*. The daemon writes to this directory as the `mongodb-mms` user.

See [rootDirectory](#).

**HTTP Service** The interface through which the *Monitoring Agent* communicates with Ops Manager.

See [Ops Manager Application](#).

**initial sync** The MongoDB operation that replicates data from an existing *replica set* member to a new member. Ops Manager uses initial sync when creating a new *head database*.

See [Initial Sync](#).

See also [Replica Set Data Synchronization](#) in the MongoDB manual.

**job** See [backup job](#).

**monitoring** The real-time reporting, visualization, and alerting of the state of your MongoDB processes. See [Monitoring](#).

**Monitoring Agent** A lightweight component that runs within your data center and monitors your MongoDB processes via the MongoDB wire protocol. No direct file system access is needed. See [Monitoring Agent](#).

**namespace** The combination of the database name and collection name:

`[database-name] . [collection-name]`

**oplog slice** A compressed batch of entries for the tailed *oplog* of a backed-up shard or replica set. The *Backup Agent* creates an oplog slice and sends it to the HTTP Service, which stores it in the *Oplog Store Database*.

The *Backup Daemon* retrieves the slice and applies it to the associated *head database*.

See [Oplog Stores Page](#).

**Oplog Store Database** The database where Ops Manager stores *oplog slices* before applying them to a deployment’s backup.

See [Oplog Stores Page](#).

**ping** A data transmission sent by the *Monitoring Agent* to Ops Manager to confirm that the agent and its MongoDB processes are running and reachable.

**point-in-time restore** A database restoration that captures the state of your data at a moment in-between *snapshots*. Point-in-time restores take longer to perform than snapshot restores.

See [Restore Overview](#).

**process** An instance of MongoDB running on a given host and port. The MongoDB database process is `mongod`. MongoDB also uses the `mongos` process to route operations in the *sharded clusters*.

See [MongoDB Package Components](#) in the MongoDB manual.

**Public API key** A unique identifier that authenticates an Ops Manager user through the *Public API*. The key belongs to the user, as opposed to the *agent API key*, which belongs to the group.

See [Enable the Public API](#).

**role** The access given to an Ops Manager or MongoDB user.

- For descriptions of Ops Manager user roles, see [Ops Manager Roles](#).
- For descriptions of MongoDB user roles, see [Built-in Roles](#) in the MongoDB manual.

**server** A physical or virtual machine that hosts one or more MongoDB processes.

**snapshot** A backup of your data captured at a specific interval and stored in either a [blockstore](#) or a file system.

Ops Manager creates snapshots from the backups kept on the [head databases](#). The [Snapshot Frequency and Retention Policy](#) determines the interval for taking snapshots and how long to store them.

See also [custom snapshot](#).

**snapshot frequency and retention policy** The schedule for how often to take [snapshots](#) and how long to store them.

See [Snapshot Frequency and Retention Policy](#).

**storage engine** The database storage engine manages how data is stored on disk. MongoDB versions 3.0 and higher offer multiple storage engines.

See [Storage and FAQ: MongoDB Storage](#) in the MongoDB manual.

**sync store** During [initial sync](#) of a backed-up deployment, Ops Manager briefly stores slices of the backed-up deployment in a temporary sync store on the Application Database. Ops Manager uses the storage while streaming slices to the Backup Daemon.

**version manifest** The list of all released MongoDB versions. Ops Manager uses this list if running in [local mode](#).

See [Version Manifest](#).

## 15 Release Notes

[Ops Manager Server Changelog](#) A record of changes to Ops Manager.

[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.

### 15.1 Ops Manager Server Changelog

#### Ops Manager Server 2.0.6

*Released on 2016-08-18*

- Agent Upgrades: [Automation Agent 2.5.20.1755](#)
- Fixed case where acknowledged alerts could be opened again.
- Fixed issue where DNS failures on the hostname(s) of the Ops Manager application database cause Ops Manager to shutdown.
- File system snapshot stores can be used in group specific snapshot store filters.
- Fixed issue where an unconfigured Backup Daemon could be assigned a backup job.
- Upgrade to JDK8u102.

#### Ops Manager Server 2.0.5

*Released on 2016-07-14*

- Agent Upgrades: [Automation Agent 2.5.19.1732](#), [Monitoring Agent 3.9.1.326](#)
- Fixed credentialstool on Windows, which is used to encrypt passwords in the config file.

- Fixed Backup Daemon auto-download of RHEL platform specific builds.
- Added support for LDAP referrals for Ops Manager user authentication.
- Added support for changing LDAP search attribute for Ops Manager user authentication.
- Fixed index creation UI in Firefox and IE11.

## **Ops Manager Server 2.0.4**

*Released on 2016-05-20*

- Agent Upgrades: *Automation Agent 2.5.18.1647, Backup Agent 3.9.0.336*
- Fixed failure to generate diagnostics archive due to large amount of log data.
- Validate Automation sslMode changes at publish time instead of draft.
- Allow Automation to transition from sslMode disabled to not having a sslMode.
- Fixed false positive auth mechanism validation failures when starting backup.
- Fixed issue with processing some types of aggregation queries when calculating suggested indexes.
- Fixed exception during a backup restore if that data previously was on a blockstore that has since been deleted.
- Removed Ubuntu 14.04 enterprise builds for MongoDB 2.4.X that were erroneously in the version manifest.
- Ability to edit LDAP Groups for Automation Admin was accidentally hidden.
- Fixed javascript error on empty profiler view.
- Upgrade to JDK8u92.

## **Ops Manager Server 2.0.3**

*Released on 2016-03-24*

- Agent Upgrades: *Automation Agent 2.5.17.1604*
- Fixed critical bug in conversion to config server replica sets. Conversions to config server replica sets should be not performed with Ops Manager 2.0.2.
- Fixed Ops Manager not recording HTTP access logs.
- Fixed LDAP PEM settings from failing pre-flight checks even when LDAP wasn't in use.
- Fixed Automated Point-In-Time restores of Sharded Cluster with config server replica sets.
- Fixed removing SSL for a Deployment
- Email configuration changes no longer require a restart of the Ops Manager service.
- Allow specifying a temporary port for use during conversion to config server replica sets.
- Added Automation support for *net.ssl.disabledProtocols*.
- Allow control over the compression level of the File System Snapshot Store.

## **Ops Manager Server 2.0.2**

*Released on 2016-03-01*

- Agent Upgrades: [Automation Agent 2.5.16.1552](#)
- Added support rolling upgrades to config servers as a replica set (requires MongoDB 3.2.4+).
- Added support for running Agents, the Ops Manager server, and MongoDB on SUSE12.
- Added support for Slack and Flowdock notifications as system alerts.
- Fixed Automation Admin Role missing from group LDAP configuration.
- Fixed charting problem on Chrome 48+.
- Fixed issue deleting processes that were part of a config server replica set.
- Fixed issue where deployment drafts could prevent Ops Manager from starting in local mode.
- Fixed issue where disabling 2FA in Ops Manager still required 2FA for users that had it configured.
- Ops Manager upgrades on Windows require an uninstall of the previous version. This restriction was added to prevent issues that could occur on upgrade that are still unresolved without uninstall.
- Upgrade to JDK8u74.

## **Ops Manager Server 2.0.1**

*Released 2016-01-21*

- Agent Upgrades: [Automation Agent 2.5.15.1526](#).
- Stability and performance improvements for restores via automation.
- Support restores via automation for shared clusters with config server replica sets.
- Fixed editing of managed users not promoting to re-enter password (Relevant only to imported SCRAM-SHA1 users.)
- Fixed old errors from imports to automation impacting new imports.
- Automation now updates the location of the keyfile according to the defined downloadBase.
- Fixed cases where suggested indexes did not handle unexpected profiling data.
- Fixed issue with filesystem snapshots failing when trying to resume a snapshot after restart.
- Fixed LDAP form validation not allowing “ldaps”.
- Fixed the Backup Daemon not recognizing Windows MongoDB Enterprise builds.
- Fixed cases where global diagnostic archive would fail if it was too large.
- Fixed importing into automation with SSL always requiring client certificates.

## **Ops Manager Server 2.0.0**

*Released 2015-12-08*

## New Features

### General

- Adds support to monitor, back up, and automate MongoDB 3.2 deployments.
- Single Ops Manager Package: there is no longer a separate package for the Backup Daemon. The single Ops Manager package installs both the Ops Manager Application and the Backup Daemon. You can configure any server with Ops Manager to handle backups through the Backup Admin interface.
- Configuration in the Database: the Ops Manager application configuration is now stored in the application database rather than in configuration files. This allows for central configuration management.

Each Ops Manager instance, on each server, must be configured with information on how to connect to the Application Database. Local config files override the information in the database: as such, switching to configuration in the database is not required, but is recommended.

- Backup agent port change: Ops Manager no longer requires a separate port for backup traffic. All HTTP traffic is now over a single port. By default, Ops Manager uses port 8080.
- Ops Manager 2.0 updates the admin interface to show the topology of Ops Manager software, the application database, and any backup databases.
- Added support to convert to LDAP authentication for Ops Manager users at any time, with no downtime.
- Upgraded to JDK8u66.

### Automation

- Added support for x.509 member authentication.
- Improved handling of adding members to replica sets: to avoid disrupting majority writes, new members are now added to *replica sets* as priority=0, votes=0 until they are ready.
- Added the ability to manage indexes from the Ops Manager UI.
- Improved index creation: indexes are now created in a rolling fashion.
- Added Automation support for Windows MongoDB instances.

### Monitoring

- Added a new profiler with Suggested Indexes.
- Added support for maintenance windows during which time Ops Manager does not send alert notifications.

### Backup

- Filesystem snapshot storage: added the ability to store snapshots on a plain shared file system instead of a MongoDB instance. With filesystem storage, Ops Manager stores snapshots in a directory hierarchy and the data files themselves are compressed using gzip.
- Backup agent port change: Ops Manager no longer requires a separate port for backup traffic. All HTTP traffic is now over a single port. By default, Ops Manager uses port 8080

Ops Manager will automatically update any Backup Agents managed by Automation to use the new port. You will need to manually update any Backup Agents set up manually after upgrading Ops Manager. The upgrade instructions describe how to configure the `mothership` field in the configuration files of non-automated Backup Agents.

- Sync store no longer required: a dedicated sync store is no longer required: Backup *initial syncs* are “streamed” to the Backup Daemon and only use a small amount of temporary space in the *oplog store*.
- Automated restores: added a new option to automatically restore a backup to a running *replica set* or *sharded cluster*.
- Added support for namespace whitelisting, which allows you to back up only a subset of your data.
- Added the ability to manage HTTP restore link expiration from the Ops Manager UI and through the API for each individual restore request.
- Added support for the Backup Daemon to download required MongoDB binaries from the Ops Manager web server when they are not available locally.

### **Associated Agent Updates**

- *Automation Agent 2.5.11.1484*
- *Monitoring Agent 3.9.1.238*
- *Backup Agent 3.9.0.336*

### **Considerations for Upgrade**

**Backup Database** There are data migrations that touch the various backup data stores that make up the *Backup Data Storage*. The data stores must all be online when you upgrade. Any data stores that are no longer in use should be deleted through the Ops Manager UI before upgrading.

**Backup Daemon** Beginning with Ops Manager 2.0, there is no separate Backup Daemon package. The Ops Manager package also installs the Backup Daemon. When started, the Ops Manager package automatically starts two services: the Ops Manager Application and the Backup Daemon. You choose on which servers to “activate” the Backup Daemon. The daemon always runs, but it performs no operations unless activated.

After upgrade, a server that runs **only** the Ops Manager Application continues to do so but now also runs a “dormant” Backup Daemon service. The Backup Daemon remains dormant as long as you do not activate it.

A server that runs **only** a Backup Daemon runs Ops Manager with an “activated” Backup Daemon and a “dormant” Ops Manager Application. The Ops Manager Application remains dormant as long as you do not direct HTTP traffic to it.

**Backup HTTP Service** Beginning with Ops Manager 2.0, there is no Backup HTTP Service on port 8081. Any Backup Agents that are managed by Automation will be automatically updated to use the new port, 8080. For Backup Agents that were installed **manually**, you must edit the agent’s configuration file, as described in the procedure below. You must have access to the servers running any manually installed Backup Agents.

**Warning:** You must configure the new port for any **manually installed** Backup Agents, or the agents will have no access to Ops Manager.

**Agent Updates** Do not update the agents before upgrade. If you use Automation, Ops Manager prompts you to update the agents after you upgrade. Follow the prompts to update the agents through the Ops Manager UI. Do *not* update the agents manually.

**conf-mms.properties** Beginning in 2.0, Ops Manager stores global configuration settings in the Ops Manager Application Database and stores only local settings in the Ops Manager server’s `conf-mms.properties` file. The upgrade procedure uses the existing `conf-mms.properties` file to connect to the Ops Manager Application Database before replacing the existing file with the new, smaller 2.0 file.

**Restore properties** The following properties no longer apply and are replaced by settings specified when initiating a restore:

- mms.backup.restore.linkExpirationHours
- mms.backup.restore.linkUnlimitedUses

**mms.conf** If you have modified the `mms.conf` file in your current installation (which is not typical), back up the file. You must use the new `mms.conf` file installed by the upgrade. You can redo any modifications once the upgrade is complete.

See: [Upgrade Ops Manager](#) for upgrade instructions for your operating system.

## Ops Manager Server 1.8.3

*Released 2015-12-15*

- Fixed issue where monitoring settings for existing servers were not always editable.
- Support for additional Amazon Simple Email Server regions. To specify regions other than the default US-EAST, see `aws.ses.endpoint`.
- Fixed SNMP notification mechanism for System Alerts.
- Fixed user privileges for MongoDB 3.0 missing from UI on Users & Roles page.
- Upgraded to JDK8u66.

## Ops Manager Server 1.8.2

*Released 2015-10-20*

- Agent Updates: [Automation Agent 2.0.14.1398](#), [Monitoring Agent 3.7.1.227](#), and [Backup Agent 3.4.2.314](#).
- MONGODB-X509 authentication mechanism no longer requires MongoDB Enterprise.
- Fixed system alerts failing to connect to Application Database and Backup Databases running with SSL.
- Fixed issue where Backup resync of a Config Server could cause the Backup Job to get stuck.

## Ops Manager Server 1.8.1

*Released 2015-08-17*

- Agent Updates: [Automation Agent 2.0.12.1238](#), [Monitoring Agent 3.7.0.212](#)
- Updated Backup seedSecondary script for MongoDB 3.0.
- Fixed adding users with GLOBAL roles to individual groups.
- Fixed Host Down alerts not firing correctly for arbiters.
- Fixed error when trying to enable x.509 authentication for Monitoring only (without Automation).
- Fixed error when trying to enable host log collection.
- Fixed case where an acknowledged Alert can be re-opened when Alert processing is behind.
- Fixed case where monitoring classified a Config Server as a Standalone when there were no mongos services.

## Ops Manager Server 1.8.0

*Released 2015-06-23*

## New Features

### Security

- Automation now supports [SSL](#) and MongoDB Enterprise authentication mechanisms: [Kerberos](#), [LDAP](#), and [x.509](#).  
Ops Manager 1.8 can start new MongoDB instances using SSL and enterprise authentication mechanisms and import existing instances using SSL and enterprise authentication for management.
- Added the ability to [specify a proxy server](#) for Ops Manager to use to access external services.
- Added [support for self-signed CAs and client certificates](#) when using SSL LDAP for Ops Manager user authentication.

### Alerts

- [System Alerts](#): system alerts allow an Ops Manager Administrator to receive alerts when the state of the software itself is unhealthy.
- [Global Alerts](#): global alerts allow an Ops Manager administrator to monitor any set of Ops Manager groups without needing to configure the alerts on a group-by-group basis.
- Added the ability to [deliver Group, Global, and System alerts via an HTTP webhook](#).
- Lowered the alerting check frequency from five minutes to one minute, allowing for more responsive alerts.

### Automation

- Automation now uses distribution-specific builds for MongoDB Community Edition when one is available for the operating system and version in use. Previously, Automation used the generic MongoDB Community Edition build.

Upgrading the Automation Agent and Ops Manager to the new version will not automatically change your MongoDB deployments to a distribution-specific build: if you wish to use the distribution-specific build, you will need to [update the MongoDB version](#).

- Added support to [change the storage engine for a MongoDB deployment using Automation](#).
- **Beta:** Added Automation support for Windows MongoDB instances. This feature must be enabled for an Ops Manager group for it to be available.

### Monitoring

- Standby Monitoring Agents now check in with Ops Manager more frequently. You can now configure the Monitoring Agent session timeout to allow for faster failover. See: [Monitoring Agent Redundancy](#) for more information.

### Backup

- Added the ability to configure the Backup Database's block size. The [Configure the Size of the Blocks in the Blockstore](#) tutorial describes how to configure the size of the blocks in the Backup Database's blockstore.
- Added the ability to initiate backup SCP restores through the Public API. See: [Restore Jobs](#).

## **Associated Agent Updates (v1.8)**

- *Automation Agent 2.0.9.1201*
- *Monitoring Agent 3.3.1.193*
- *Backup Agent 3.4.1.283*

## **Considerations for Upgrade (v1.8)**

- Ops Manager 1.8 requires that the *Ops Manager Application Database* and *Backup Data Storage* run MongoDB 2.6 or later. Ops Manager will not start after upgrade if your backing databases are using an earlier version of MongoDB. The MongoDB Manual provides upgrade tutorials with each release. To upgrade from MongoDB 2.4 to 2.6, see: [Upgrade MongoDB to 2.6](#).
- When you upgrade to Ops Manager 1.8, Ops Manager disables all Automation Agents until they are upgraded to *Automation Agent 2.0.9.1201*. You can upgrade the Automation Agents by clicking the link that appears in the *Please upgrade your agents* banner that will appear on the *Deployment* page in the Ops Manager interface.
- Direct upgrade is only allowed from Ops Manager 1.5 and Ops Manager 1.6. To upgrade to Ops Manager 1.8 from an earlier version of MongoDB, you must first upgrade to Ops Manager 1.6, and then to 1.8.
- In Ops Manager 1.8, `mms.multiFactorAuth.level` replaces the deprecated `mms.multiFactorAuth.require` setting. `mms.multiFactorAuth.level` supports more values than its predecessor.

Ops Manager will not start with `mms.multiFactorAuth.require` in the properties file, but will report an error indicating that the setting has been deprecated, and that you must update your configuration.

- Ops Manager 1.8 does not include the Backup HTTP Service: its functionality is now part of *System Alerts* and *Global Alerts*.
- System Alerts give new insight into the health of Ops Manager and may immediately trigger on upgrade if Ops Manager is not in the expected state. For example, if your Application or Backup databases have startup warnings or if the connection strings to those databases point to any unreachable MongoDB instances, Ops Manager will issue an alert.
- The Ops Manager Deployment user interface has been streamlined such that the *View Mode* and *Edit Mode* dual views have been merged into a unified view

## **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](#).
- For Backup HTTP pull restore, the link expiration and the number of allowed uses of a link are now configurable.

## 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 Microsoft 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 issue ([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 Data Storage*. 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: [View Diagnostics](#) 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*.
- For Backup HTTP pull restore, the link expiration and the number of allowed uses of a link are now configurable.

## 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 be 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 [Assign Snapshot Stores to Specific 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 using LDAP with 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.

## **15.2 Automation Agent Changelog**

### **Automation Agent 2.5.20.1755**

*Released with Ops Manager 2.0.6 on 2016-08-18*

- Improve logging on authentication failures.
- Fixed setting `clusterAuthMode` on sharded clusters.

### **Automation Agent 2.5.19.1732**

*Released with Ops Manager 2.0.5 on 2016-07-14*

- Substantial optimization in state-gathering.
- Configurable timeout for connections to MongoDB processes.
- Fixed problem verifying success when creating text indexes in rolling index builds.

## **Automation Agent 2.5.18.1647**

*Released with Ops Manager 2.0.4 on 2016-05-20*

- Agent no longer downloads restore data for arbiters.
- Fixed some cases where CSRS conversion could get stuck.
- Fixed agent unable to restart a config server if all config servers were down.
- Fixed issue validating MongoDB versions when a cluster was on mixed operating systems.

## **Automation Agent 2.5.17.1604**

*Released with Ops Manager 2.0.3 on 2016-03-24*

- Fixed import of arbiter using a different keyfile than existing configuration.
- Allow specifying a temporary port for use during a CRSR upgrade.

## **Automation Agent 2.5.16.1552**

*Released with Ops Manager 2.0.2 on 2016-03-01*

- Added support for managing MongoDB on SUSE12.
- Added support for rolling upgrades to config servers as a replica set. See [Convert Config Servers to a Replica Set](#).

## **Automation Agent 2.5.15.1526**

*Released with Ops Manager 2.0.1 on 2016-01-21*

- Stability and performance improvements for restores via automation.
- Added optimization to prioritize replica set reconfiguration actions over index builds.
- Improved index building mechanism: index builds are no longer performed in a rolling fashion for 2-node replica sets, but instead are built in the background.
- Added optimization to not compare unsupported index options when determining whether or not an index already exists.
- Fixed issue with importing existing deployments that include arbiters running with authentication.
- Fixed issue with rolling storage engine conversion for replica sets to ensure a super majority is always up.
- Fixed issue with creating custom roles on sharded clusters running MongoDB 3.2 with config server replica sets.

## **Automation Agent 2.5.11.1484**

*Released with Ops Manager 2.0.0 on 2015-12-08*

- Added support for MongoDB 3.2.0 clusters with config servers as replica sets.
- Added support for automated restores via the Automation Agent.
- Added support for rolling index builds.
- Added support for configuring WiredTiger encrypted storage for MongoDB 3.2.

- Added support for rolling conversion to X-509 member authentication.
- Improved handling of sharded clusters with members running on both Linux and Windows-based operating systems.
- Added optimization when starting a new Monitoring or Backup Agent to ensure that the process is running before achieving Goal State.
- Fixed `glibc` incompatibility issue on RHEL5 and RHEL6.
- Fixed bug where failed Automation Agent automatic updates can cause surge in configuration calls from the Automation Agent.

### **Automation Agent 2.0.14.1398**

*Released with Ops Manager 1.8.2 on 2015-10-20*

- Fixed Agent from not recognizing RHEL Workstations as RHEL.

### **Automation Agent 2.0.12.1238**

*Released with Ops Manager 1.8.1 on 2015-08-17*

- Fixed managing an existing deploy with user that has “root” privileges.
- Fixed case where storage engine conversions could get stuck if replica set contained an arbiter.
- Fixed updating credentials after failed attempt to manage an existing deployment.

### **Automation Agent 2.0.9.1201**

*Released with Ops Manager 1.8 on 2015-06-23*

- Added support for managing SSL-enabled deployments.
- Added support for managing deployment using Kerberos, LDAP, and x.509 Client Certificate authentication.
- Added support to import an existing `mongos` with a config file.
- Added support for importing an existing deployment that contains authenticated `arbiter`s on which the hostname does not resolve locally to the loopback interface.
- Added the ability to upgrade the `authSchemaVersion` when auth is not enabled.
- Added support to change the storage engine for `replica sets` with more than one data node.
- Enabled storage engine conversions for single-node replica sets and `standalones`.
- Added more detailed logging of when MongoDB, the Monitoring Agent, or the Backup Agent rotate their logs.
- Added support for distribution-specific MongoDB Community Edition builds.
- Added up-front validation to ensure that MongoDB processes are running as the same user as the Automation Agent.
- Added functionality to delete MongoDB binaries on disk that are not used by a managed process.
- Added optimization where Ops Manager assumes success when starting a forked MongoDB process, rather than waiting for EOF.
- Improved algorithm for balancing `mongod` processes across cores.
- When deleting directories, symlinks are no longer deleted.

- Fixed issue importing credentials for MONGODB-CR users from SCRAM-SHA-1 deployments. See: [MMS-2612](#) for more details.
- Fixed issue with deriving the default port for config servers started with the `--configsvr` option but with no port specified. See: [MMS-2489](#).
- Fixed issue with configuring `oplog` sizes greater than 1TB.
- Fixed issue where the Automation Agent interfered with manually-created replica set tags.
- Ensured that the Automation Agent fails gracefully when an expected user does not exist during an initial import.

## **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.

## **15.3 Monitoring Agent Changelog**

### **Monitoring Agent 3.9.1.326**

*Released with Ops Manager 2.0.5 on 2016-07-14*

- Fixed expired code-signing certificate in MSI. This is a packaging change only, the agent version did not change.

## **Monitoring Agent 3.9.1.238**

*Released with Ops Manager 2.0.0 on 2015-12-08*

- Added support for MongoDB 3.2.0 config servers as replica sets.

## **Monitoring Agent 3.7.1.227**

*Released with Ops Manager 1.8.2 on 2015-10-20*

- Fixed potential memory leak when profiler is enabled.

## **Monitoring Agent 3.7.0.212**

*Released with Ops Manager 1.8.1 on 2015-08-17*

- Avoid harmless authentication in mongod log files when reading oplog stats using the clusterMonitor role.

## **Monitoring Agent 3.3.1.193**

*Released with Ops Manager 1.8 on 2015-06-23*

- Added support for x.509 Client Certificate authentication. For configuration details, see: [Configure the Monitoring Agent User for x.509 Client Certificate Authentication](#).
- The Kerberos credentials cache now uses a fixed name.
- Improved support for collecting database statistics from secondaries.
- Adds an optimization to ensure the Monitoring Agent's database stats collection tasks do not synchronize.

## **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 cluster-Monitor 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.

## **15.4 Backup Agent Changelog**

### **Backup Agent 3.9.0.336**

*Released with Ops Manager 2.0.4 on 2016-05-20*

- Fixed crash that could occur if a collection was deleted during an initial sync.

### **Backup Agent 3.9.0.336**

*Released with Ops Manager 2.0.0 on 2015-12-08*

- Added support for streaming initial syncs.
- Added support for MongoDB 3.2.0 config servers as replica sets.
- Added the ability to only backup selected namespaces (whitelist).
- Fixed issue with initial sync failing due to collections being deleted during the sync.
- Fixed issue with collection names with trailing spaces.

### **Backup Agent 3.4.2.314**

*Released with Ops Manager 1.8.2 on 2015-10-20*

- Fixed issue where initial syncs would fail if a namespace was deleted during the sync.

### **Backup Agent 3.4.1.283**

*Released with Ops Manager 1.8 on 2015-06-23*

- Added support for x.509 Client Certificate authentication. For configuration details, see: [Configure Backup Agent User for x.509 Client Certificate Authentication](#).
- The Kerberos credentials cache now uses a fixed name.
- Fixed a race condition which could result in inconsistent cluster snapshots for MongoDB 3.0+ sharded clusters using the `backup` role.

### **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.