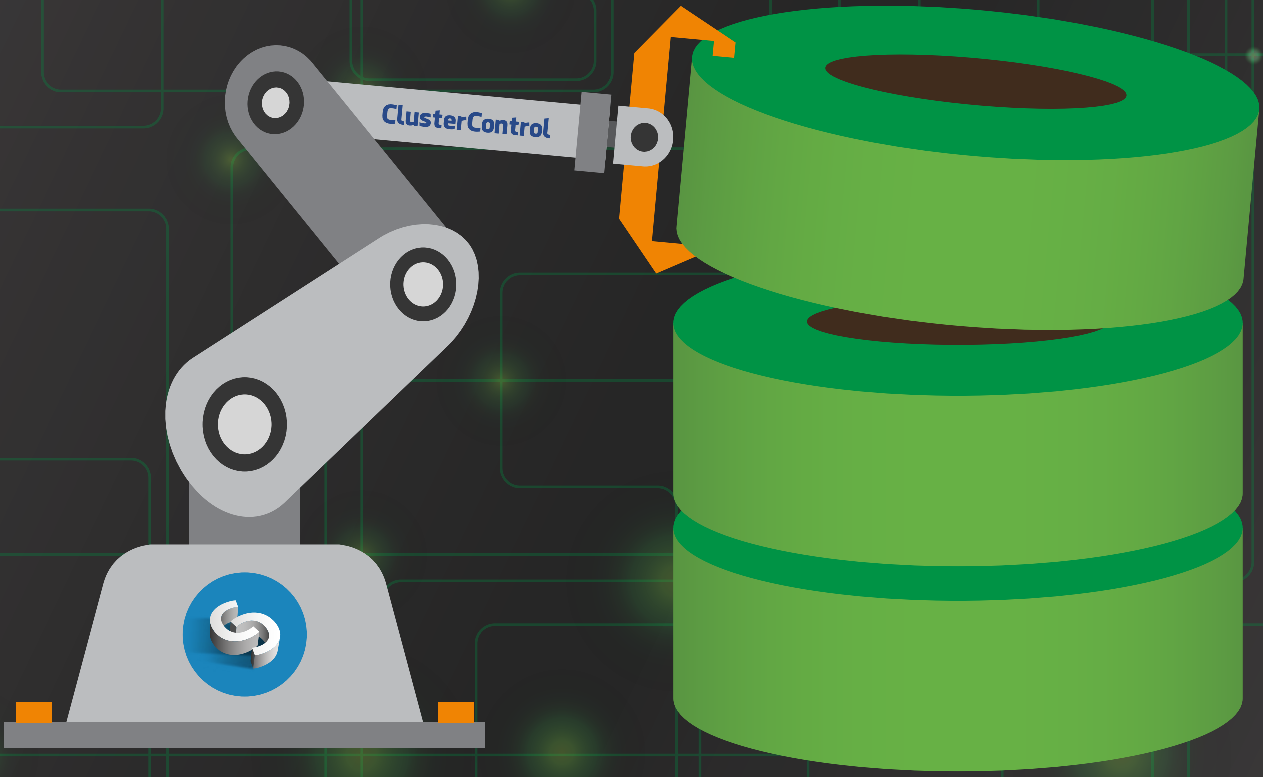
severalnines

MongoDB Management and Automation with ClusterControl



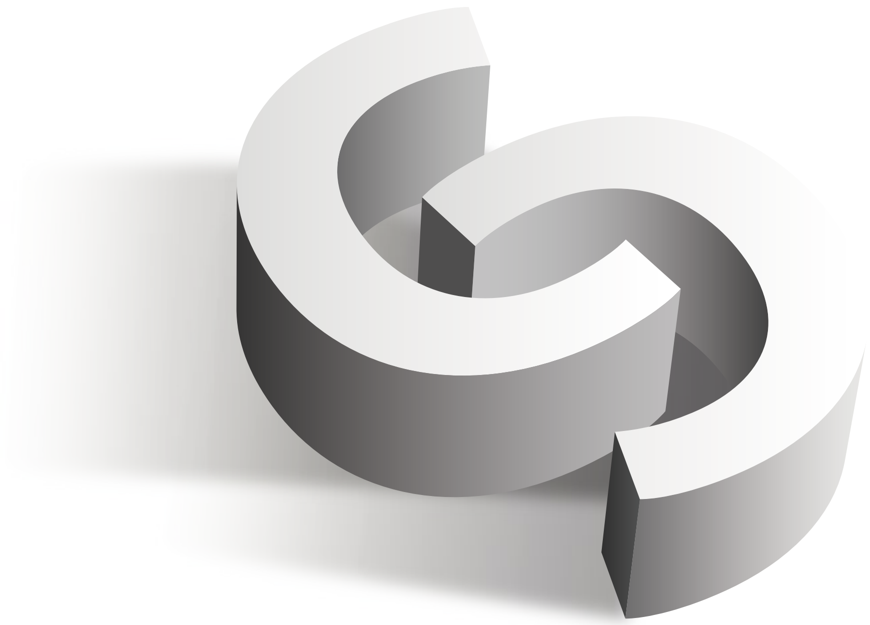
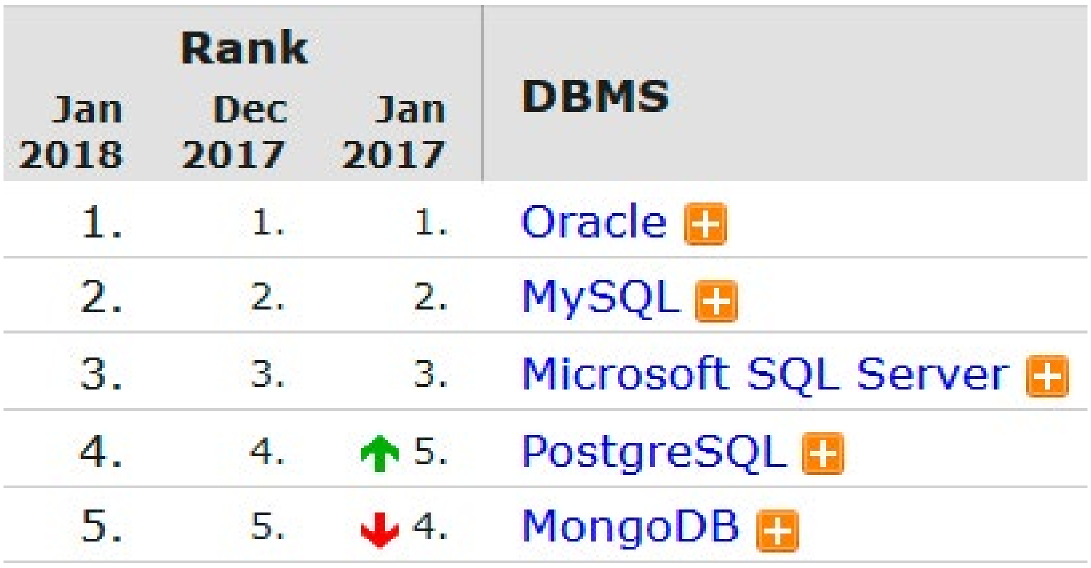


Table of Contents

Introduction

MongoDB is the world's leading NoSQL database server, and—per DBengine's ranking, the most widely-known ranking in the database industry—the 5th database server overall in terms of popularity.



Rank			DBMS
Jan 2018	Dec 2017	Jan 2017	
1.	1.	1.	Oracle
2.	2.	2.	MySQL
3.	3.	3.	Microsoft SQL Server
4.	4.	↑ 5.	PostgreSQL
5.	5.	↓ 4.	MongoDB

Figure 1. Source: http://db-engines.com/ranking

MongoDB is a document-oriented database server, using JSON-formatted documents for data rather than the columns and rows of the table structure known to any relational database administrator. This structure allows a flexibility that is difficult to obtain in current relational databases, and is behind the "schema-less" nature of MongoDB. A "collection", as the MongoDB equivalent of a relational database table is known, does not impose a specific structure on its documents. Practically speaking, this means—among other benefits—that each document in the collection can have differing fields, and a field that exists in one document in the collection need not exist in another. The JSON format also brings embedded arrays, and the ability to index on any attribute among other features.

Considerations for administering MongoDB

Built-in Redundancy

A key feature of MongoDB is its built-in redundancy, in the form of Replication. If you have two or more data nodes, they can be configured as a replica set, in which all data written to the Primary node, is replicated in near real time to the secondary nodes,

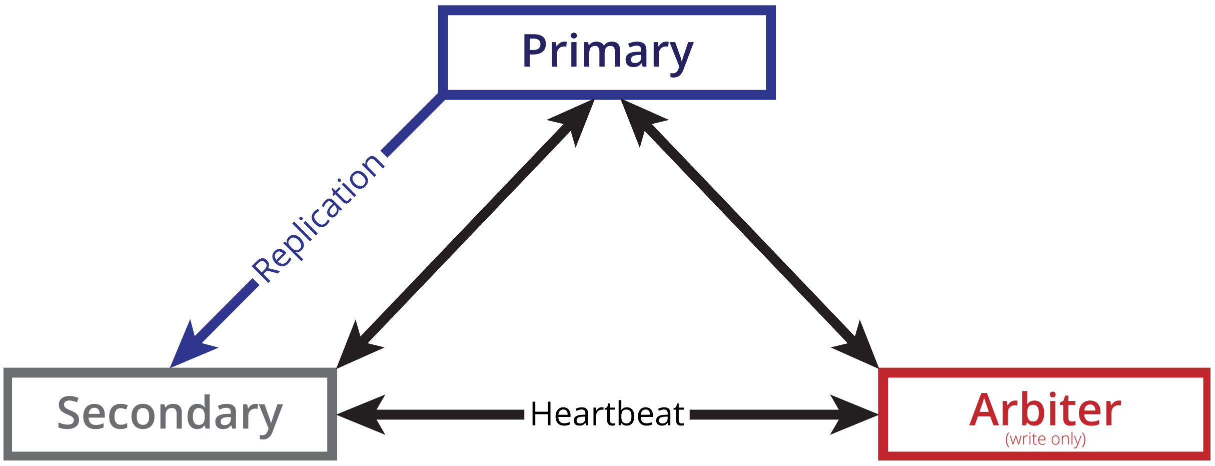


Figure 2. MongoDB Replica Set

ensuring multiple copies of the data. In the case of Primary failover, the remaining nodes in the replica set conduct an election and promote the winner to be Primary, a process that typically takes 2-3 seconds, and writes to the replica set can resume. MongoDB also uses a journal for faster, safer writes to the server or replica set, and also employs a “write concern” method through which the level of write redundancy is configured.

To manually deploy a replica set, the high-level steps are as follows:

1. Allocate a single physical or virtual host for each database node, and install the MongoDB command line client on your desktop. For a redundant replica set configuration, a minimum of three nodes are required, at least two of which will be data nodes. One node in the replica set may be configured as an arbiter: this is a mongod process configured only to make up a quorum by providing a vote in the election of a Primary when required. Data is not replicated to arbiter processes.
2. Install MongoDB on each node. Some Linux distributions include MongoDB Community Edition, but be aware that these may not include the latest versions. MongoDB Enterprise is available only by download from MongoDB’s website. Similar functionality to MongoDB Enterprise is also available via Percona Server for MongoDB, a drop-in replacement for MongoDB Enterprise or Community Edition.
3. Configure the individual mongod.conf configuration files for your replica set,

using the “replication parameter”. If you will use a key file for security, configure this now also. Note that using key file security also enables role-based authentication, so you will also need to add users and roles to use the servers. Restart the mongod process on each server.

4. Ensure connectivity between nodes. You must ensure that MongoDB replica set nodes can communicate with each other on port 27017, and also that your client(s) can connect to each of the replica set nodes on the same port.
5. Using the MongoDB command line client, connect to one of the servers, and run rs.initiate() to initialise your replica set, followed by rs.add() for each additional node. rs.conf() can be used to view the configuration.

While these steps are not as complex as deploying and configuring a MongoDB sharded cluster, or sharding a relational database, they can be onerous and prone to error, especially in larger environments.

Scalability

MongoDB is frequently referred to as “web scale” database software, due to its capacity for scaling horizontally. Like relational databases, it is possible to scale MongoDB vertically, simply by upgrading the physical host on which is resides with more CPU cores, more RAM, faster disks, or even increased bus speed. Vertical scaling has its limits however, both in terms of cost-benefit ratio and diminishing returns, and of technical limitation. To address this, MongoDB has an “auto-sharding” feature, that allows databases to be split across many hosts (or replica sets, for redundancy). While sharding is also possible on relational platforms, unless designed for at database inception, this requires major schema and application redesign, as well as client application redesign, making this a tedious, time-consuming, and error-prone process.

MongoDB sharding works by introducing a router process, through which clients connect to the sharded cluster, and configuration servers, which store the cluster meta-data, the location in the cluster of each document. When a client submits a query to the router process, it first refers to the config servers to obtain the locations of the documents, and then obtains the query results directly from the individual servers or replica sets (shards). Sharding is carried out on a per collection basis.

A critically important parameter here, for performance purposes, is the “shard key”, an indexed field or compound field that exists in each document in a collection. It is this that defines the write distribution across shards of a collection. As such, a poorly-chosen shard key can have a very detrimental effect on performance. For example, a purely time-series based shard key may result in all writes going to a single node for extended periods of time. However, a hashed shard key, while evenly distributing writes across shards, may impact read performance as a result set is retrieved from many nodes.

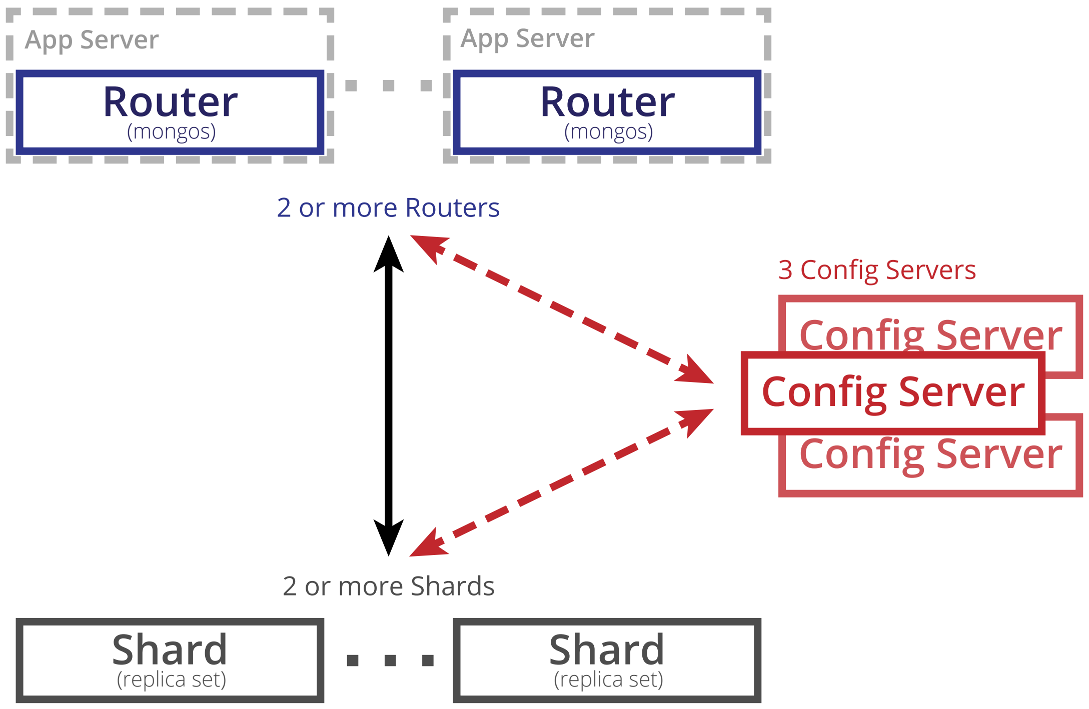


Figure 3. MongoDB Sharded Cluster

Arbiters

A MongoDB arbiter is a mongod process that has been configured not to act as a data node, but to provide only the function of voting when a replica set Primary is to be elected, to break ties and guard against a split vote. An arbiter may not become Primary, as it does not hold a copy of the data or accept writes. While it is possible to have more than one arbiter in a replica set, it is generally not recommended.

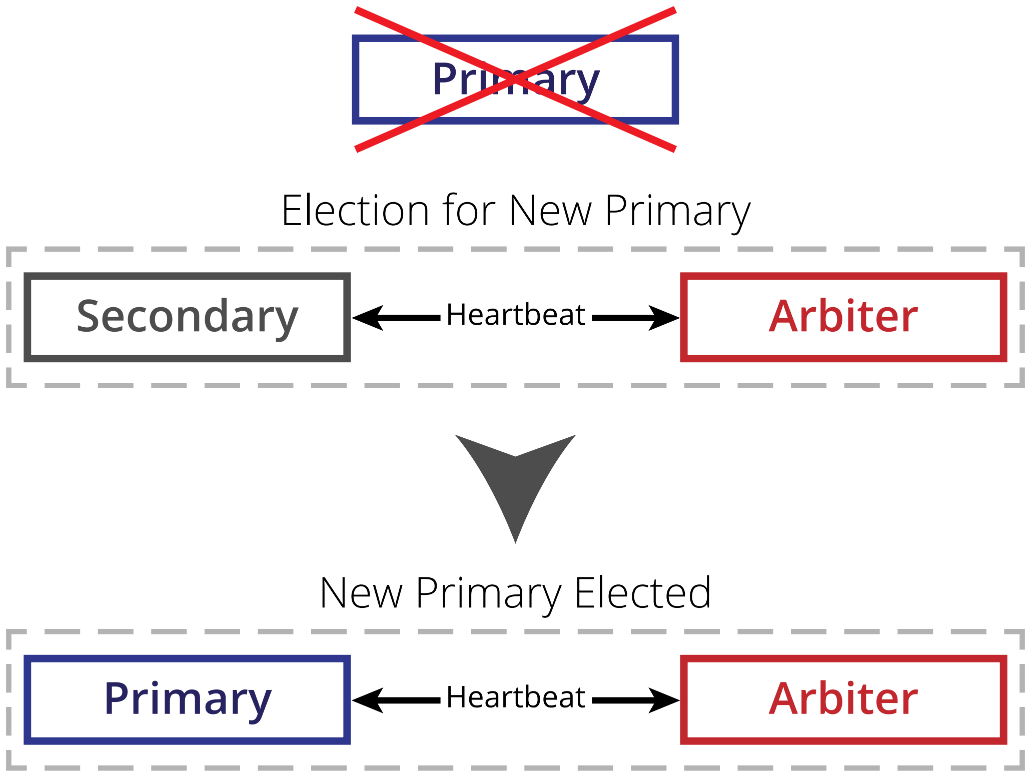


Figure 4. MongoDB elections and the arbiter process

Delayed Replica Set Members

Delayed replica set members add an additional level of redundancy, maintaining a state that is a fixed number of seconds behind the Primary. As delayed members are a "rolling backup" or a running "historical" snapshot of the data set, they can help to recover from various types of human error.

Delayed members are "hidden" replica set members, invisible to client applications, and so cannot be queried directly. They also may not become Primary during normal operations, and must be reconfigured manually in the case that they are to be used to recover from error.

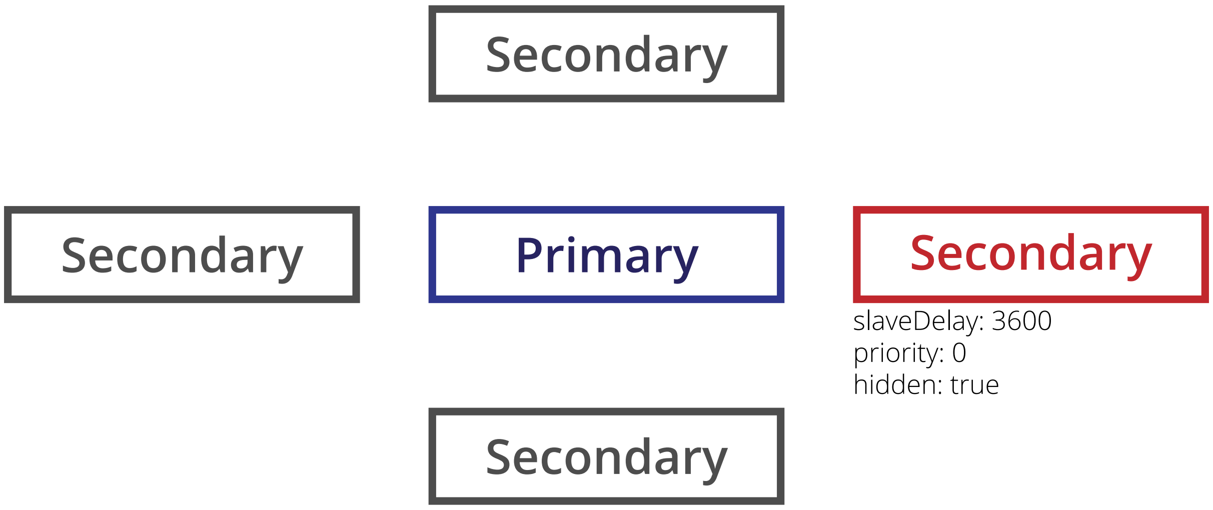


Figure 5. MongoDB Delayed Secondary node

Backups

Backing up a replica set or sharded cluster is carried out via the "mongodump" command line utility. When used with the --oplog parameter, this creates a dump of the database that includes an oplog, to create a point-in-time snapshot of the state of a mongod instance. Using mongorestore with the --replayOplog parameter, you can then fully restore the data state at the time the backup completed, avoiding inconsistency.

For more advanced backup requirements, a third party tool called "mongodb-consistent-backup" - also command line based - is available that provides fully consistent backups of sharded clusters, a complex procedure, given that sharded databases are distributed across multiple hosts.

Monitoring

There are a number of commercial tools, both official and unofficial, available on the market for monitoring MongoDB. These tools, in general, are single product management utilities, focusing on MongoDB exclusively. Many focus only on certain specific aspects, such as collection management in an existing MongoDB architecture, or on backups, or on deployment. Without proper planning, this can lead to a situation where a proliferation of additional tools must be deployed and managed in your environment.

The command line tools provided with MongoDB, “mongotop” and “mongostat” can provided a detailed view of your environments performance, and can be used to diagnose issues. In addition, MongoDB’s “mongo” command line client can also run ”rs.status()” - or in a sharded cluster ”sh.status() - to view the status of replica sets or clusters and their member hosts. The ”db.stats()” command returns a document that addresses storage use and data volumes, and their are equivalents for collections, as well as other calls to access many internal metrics.

Synopsis

This has been a brief synopsis of considerations for administering MongoDB. Even at such a high level though, it should immediately be obvious that while it is possible to administer a replica set or sharded cluster from the command line using available tools, this does not scale in an environment with many replica sets or with a large production sharded cluster. In medium to large environments comprising many hosts and databases, it quickly becomes unfeasible to manage everything with command line tools and scripts. While internal tools and scripts can be developed to deploy and maintain the environment, this adds the burden of managing new development, revision control systems, and processes. A simple upgrade of a database server may become a complex process if tooling changes are required to support new database server versions.

But without internal tools and scripts, how do we automate and manage MongoDB clusters?

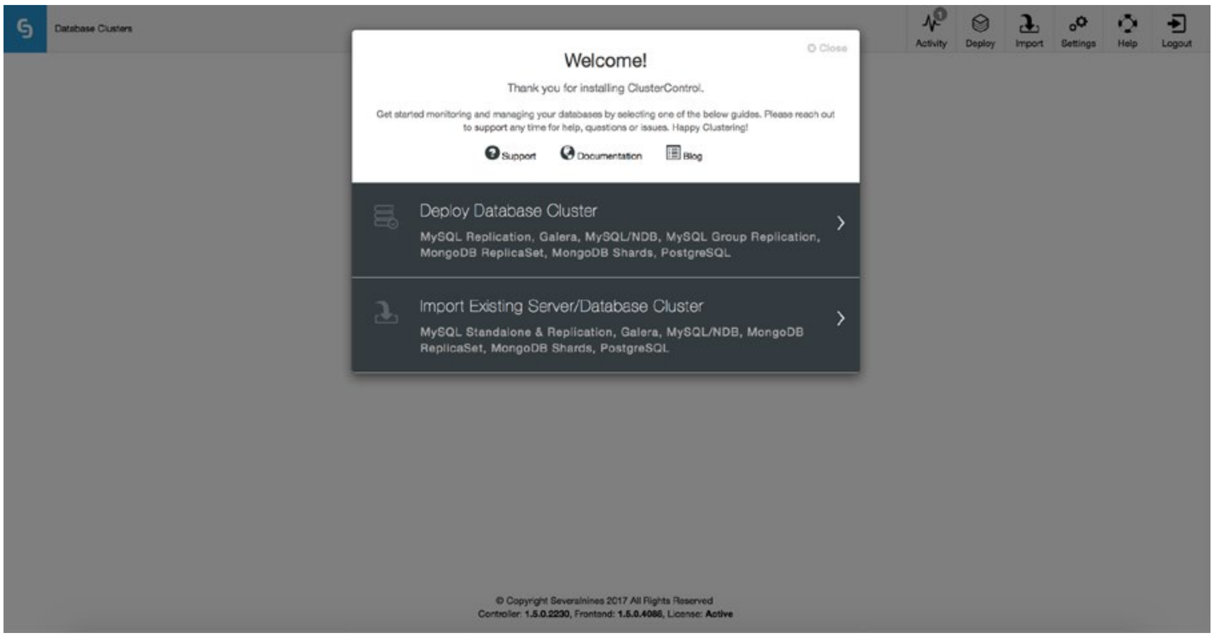
Automation with ClusterControl

ClusterControl from Severalnines is designed to answer these questions and more. ClusterControl allows you to deploy, backup, and monitor not only your MongoDB replica sets and sharded clusters, but also your other Open Source database assets. MySQL with native replication, NDB or Galera, and PostgreSQL, as well as MongoDB are managed in a single web-based unified interface. Here we take a look at ClusterControl's features in more detail.

Deployment

A core feature of ClusterControl is the deployment of MongoDB replica sets and sharded clusters, in addition to its other supported database servers and clusters. If your environment has already been deployed, ClusterControl has that covered: you can simply import the existing environment and begin managing it right away.

Once you have logged in, simply select the operation, Deploy or Import, and follow the wizard-like interface.



Welcome!

Thank you for installing ClusterControl.

Get started monitoring and managing your databases by selecting one of the below guides. Please reach out to support any time for help, questions or issues. Happy Clustering!

Deploy Database Cluster

MySQL Replication, Galera, MySQL/NDB, MySQL Group Replication, MongoDB ReplicaSet, MongoDB Shards, PostgreSQL

Import Existing Server/Database Cluster

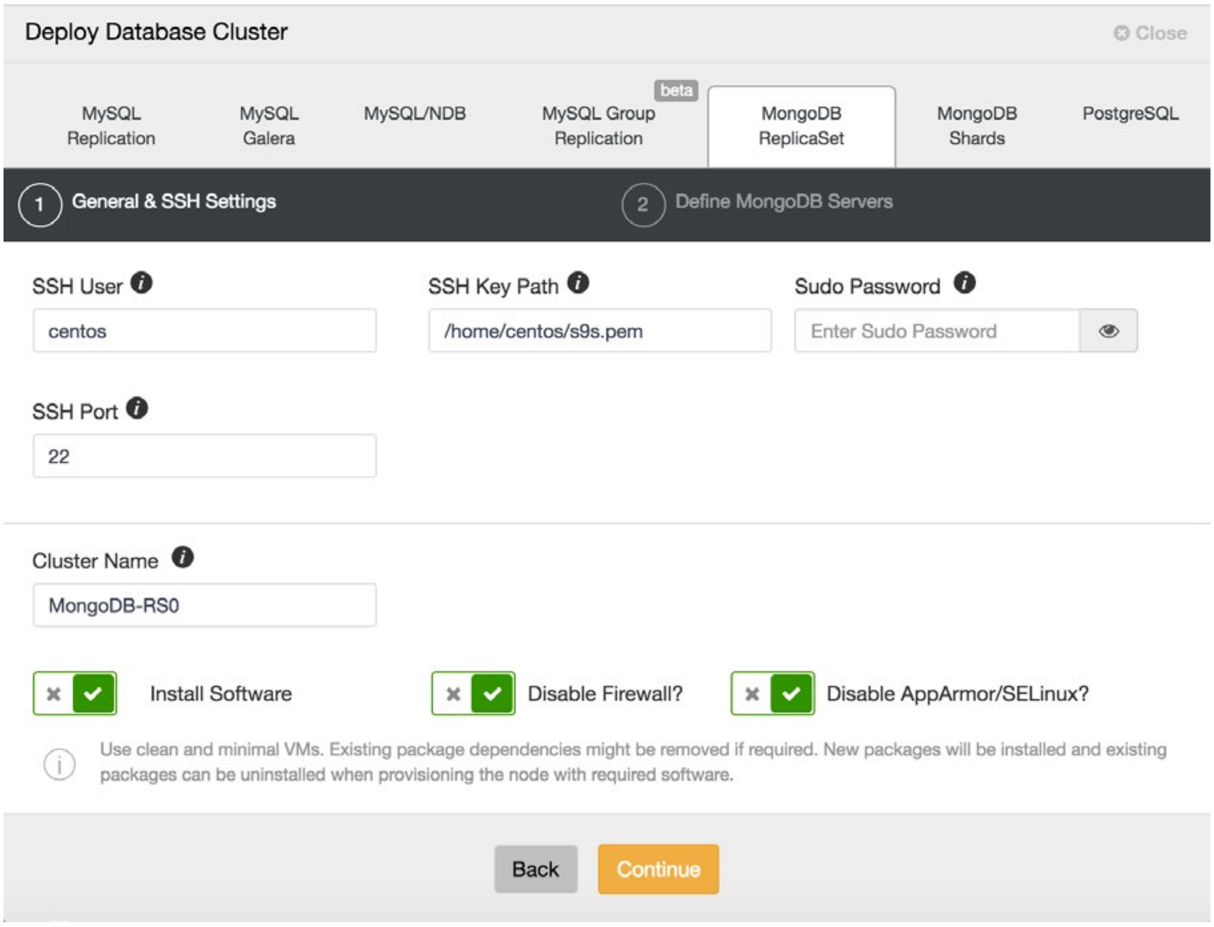
MySQL Standalone & Replication, Galera, MySQL/NDB, MongoDB ReplicaSet, MongoDB Shards, PostgreSQL

© Copyright Severalnines 2017 All Rights Reserved
Controller: **1.5.0.2230**, Frontend: **1.5.0.4086**, License: **Active**

Figure 6. ClusterControl initial screen

On the initial deployment screen you will provide ssh credentials appropriate to the hosts on which you are deploying your replica set or cluster. As ClusterControl uses password-less ssh to connect to and configure your hosts, an ssh key is required. For security reasons, it is advisable to use an unprivileged user account to log into the hosts, so a sudo password can be provided to facilitate administrative tasks. If the user account does not prompt for a sudo password, this is not needed.

You also have the option to disable the iptables or ufw firewall, and to disable AppArmor or SELinux, on the host to avoid issue with initial deployment.



SSH User: centos

SSH Key Path: /home/centos/s9s.pem

Sudo Password: Enter Sudo Password

SSH Port: 22

Cluster Name: MongoDB-RS0

✓ Install Software

✓ Disable Firewall?

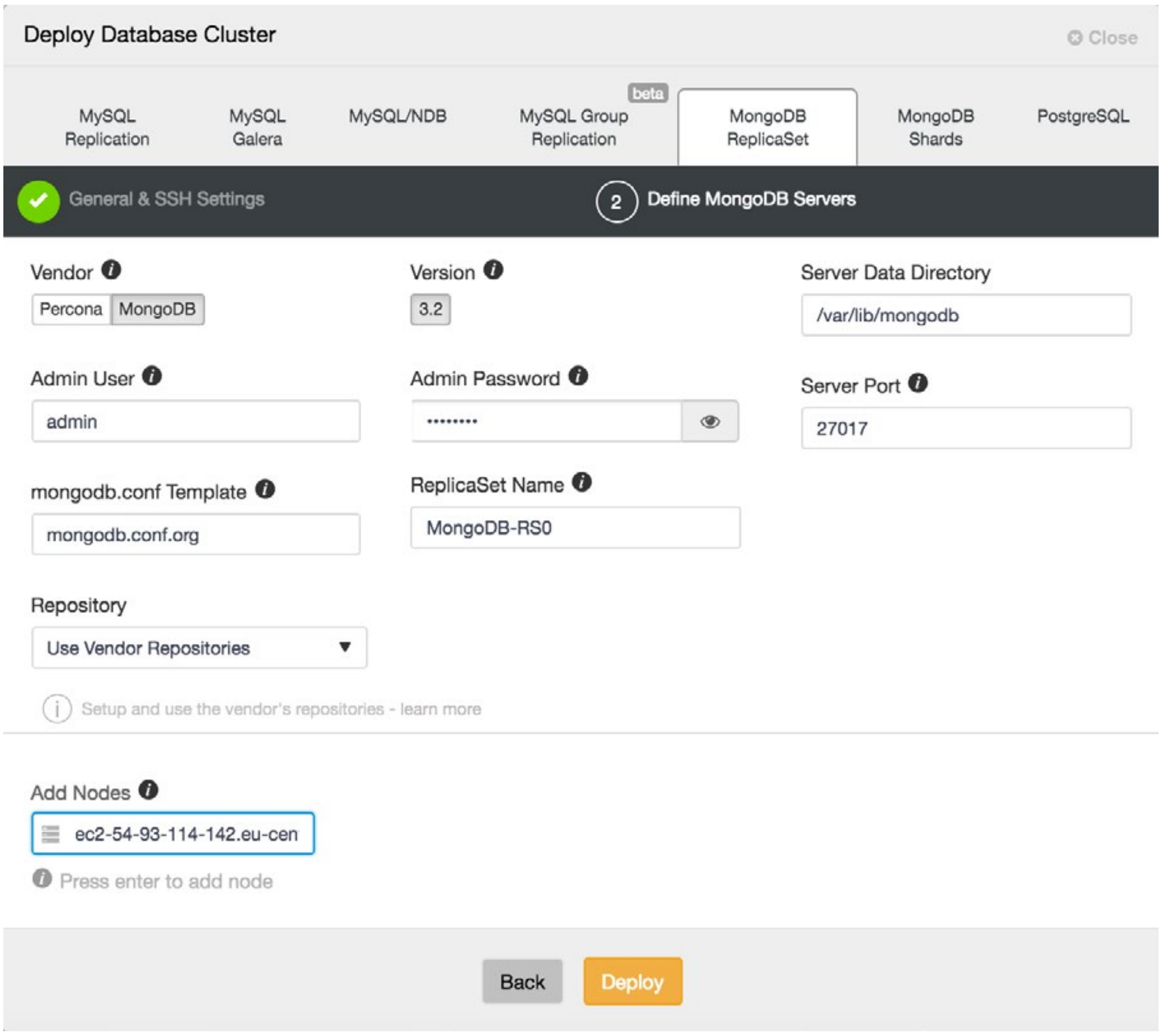
✓ Disable AppArmor/SELinux?

Use clean and minimal VMs. Existing package dependencies might be removed if required. New packages will be installed and existing packages can be uninstalled when provisioning the node with required software.

Figure 7. ClusterControl MongoDB Replica Set deployment

On the following screen, you can choose to install MongoDB binaries from either MongoDB Inc or from Percona. Here also, you must specify your MongoDB administrative user account and password as user level security is mandated. On this screen also, you can see which configuration template is being used. ClusterControl uses configuration file templates to ensure repeatable deployments. Templates are stored on the ClusterControl host and can be edited directly there, or through the ClusterControl UI.

As shown in Figure 8, you can also choose to use the vendor repositories, if you wish, or choose your own repository. In addition, you can create a new repository on the ClusterControl host automatically, to freeze the version of MongoDB that ClusterControl will deploy to the current release. Once you carried out the appropriate configuration here, click Deploy, to deploy your replica set.


Deploy Database Cluster

Close

MySQL Replication	MySQL Galera	MySQL/NDB	MySQL Group Replication (beta)	MongoDB ReplicaSet	MongoDB Shards	PostgreSQL

General & SSH Settings — 2 Define MongoDB Servers

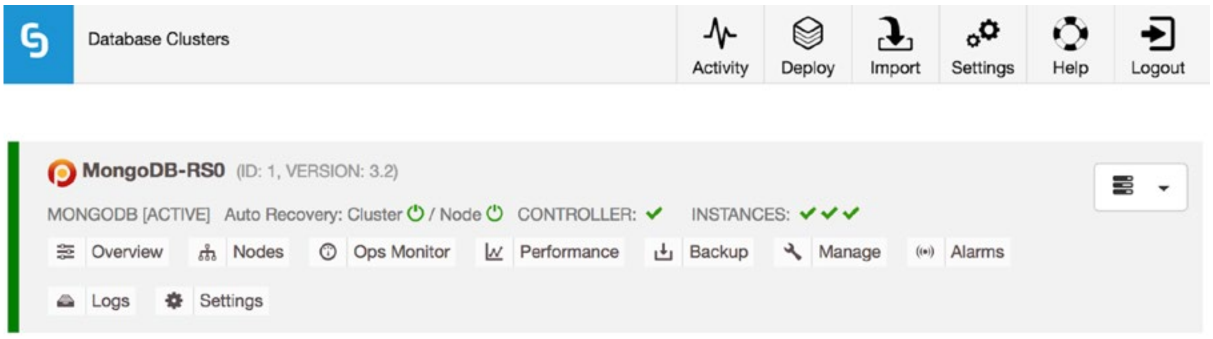
Vendor	Percona / MongoDB
Version	3.2
Server Data Directory	/var/lib/mongodb
Admin User	admin
Admin Password	••••••••
Server Port	27017
mongodb.conf Template	mongodb.conf.org
ReplicaSet Name	MongoDB-RS0
Repository	Use Vendor Repositories

Setup and use the vendor's repositories - learn more

Add Nodes: ec2-54-93-114-142.eu-cen

Press enter to add node

Figure 8. ClusterControl MongoDB Replica Set deployment


Database Clusters

MongoDB-RS0 (ID: 1, VERSION: 3.2)

MONGODB [ACTIVE] Auto Recovery: Cluster / Node CONTROLLER: ✓ INSTANCES: ✓✓✓

Figure 9. When deployment is finished, your replica set is visible in the UI

As mentioned previously, ClusterControl can deploy not only Replica Sets, but also Sharded Clusters. Two methods of doing so are supported. First, you can convert an existing MongoDB Replica Set into a Sharded Cluster, as shown in Figure 10, below.

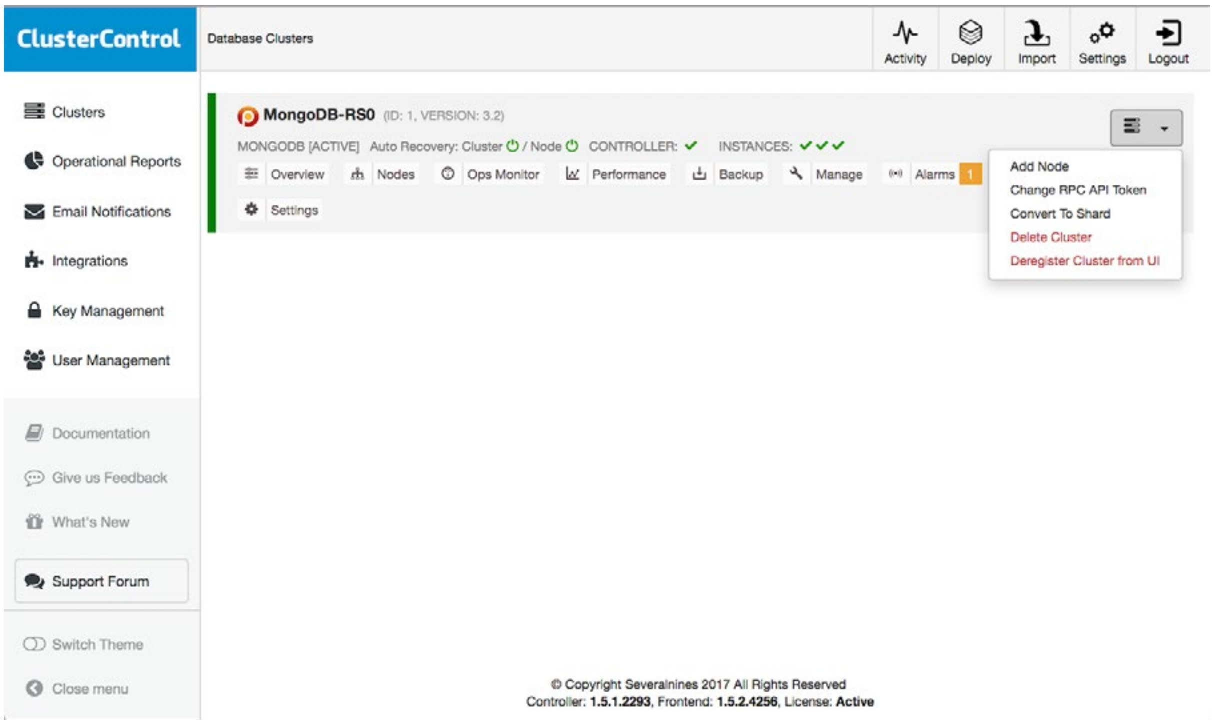


Figure 10. A Replica Set can be converted to a Sharded Cluster

When "Convert to Shard" is clicked, you are prompted to add at least one Config server (for production environments, you should add three), and a router, also known as a "mongos" process.

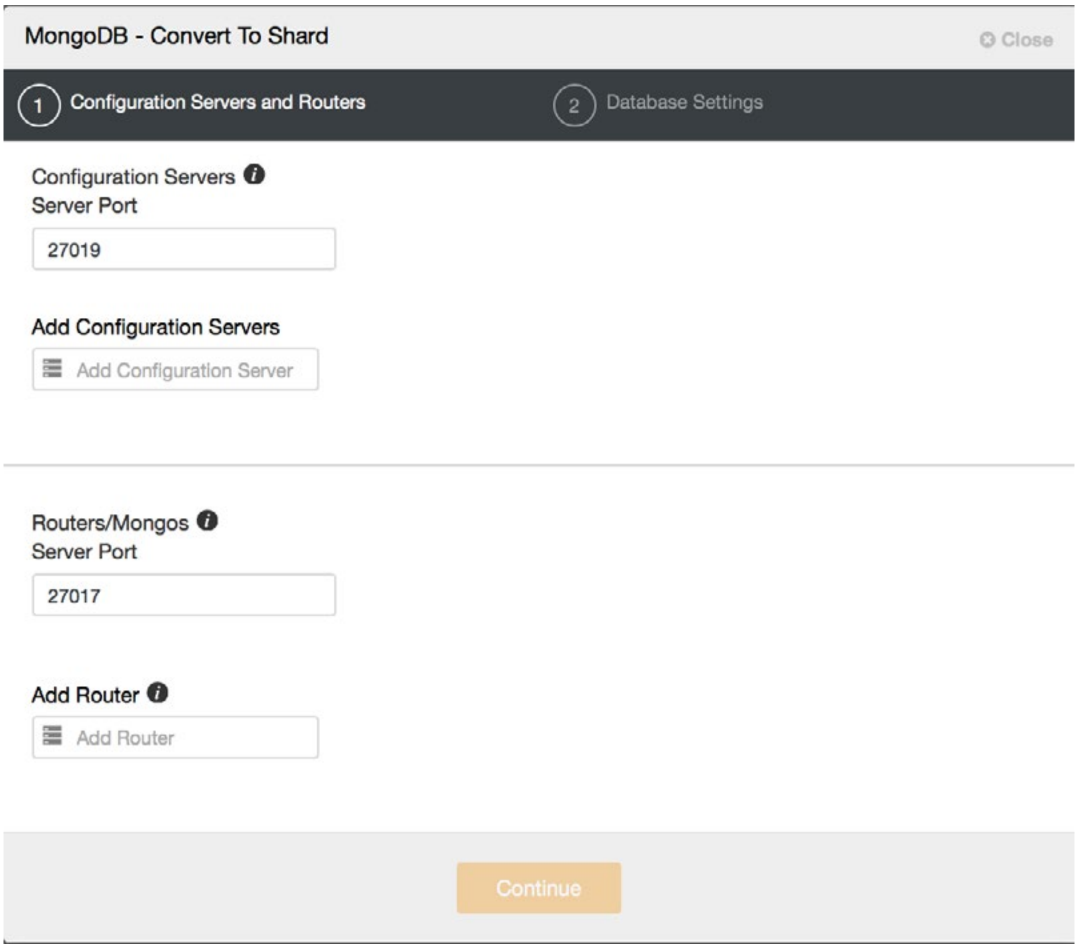


Figure 11. Adding config servers and routers (mongos)

The final stage is to choose your MongoDB configuration templates for config server and router, as well as your data directory. Finally, click deploy.

When complete, as you can see in Figure 12, below, your Database Clusters screen will show your shard health instead of individual instances. It is also possible to add additional shards as needed from this view, as shown.

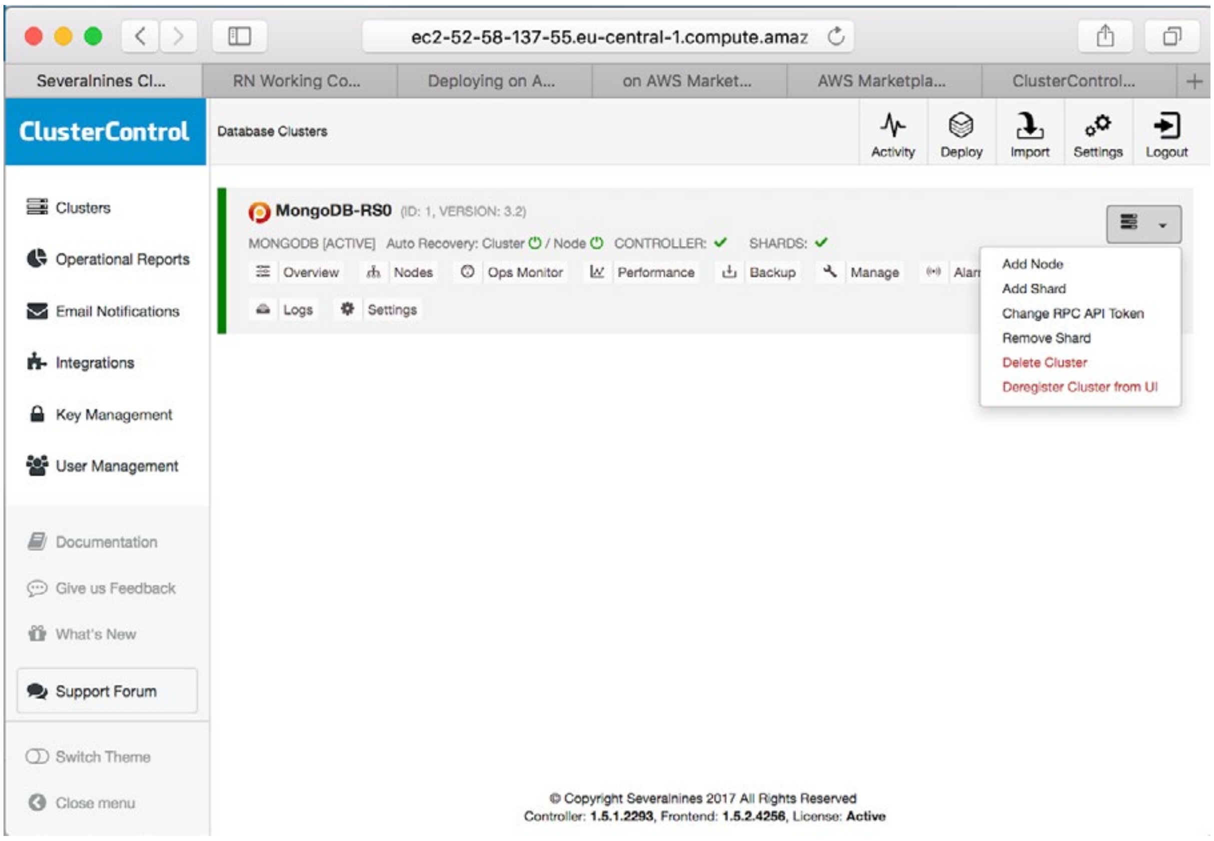


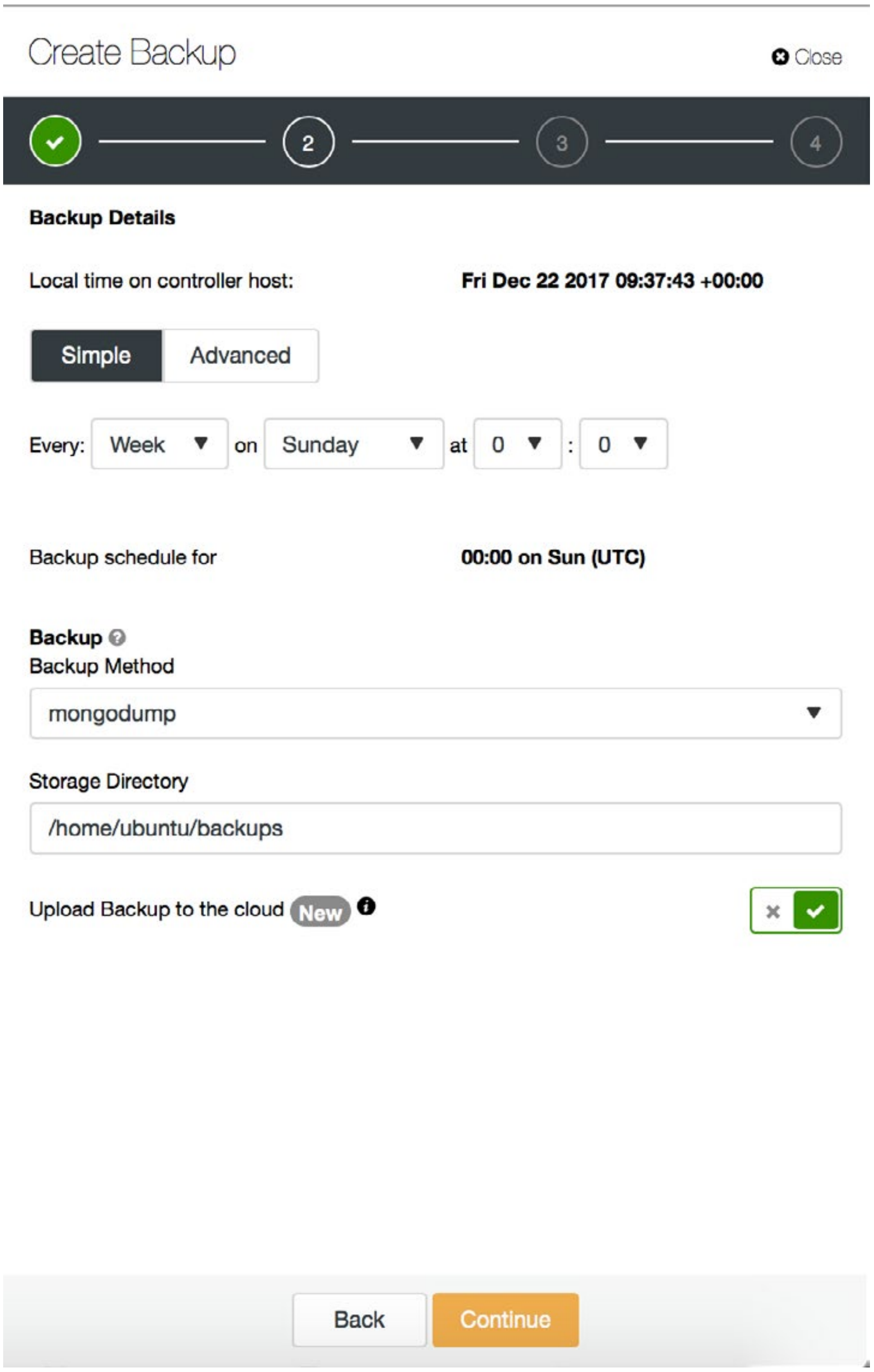
Figure 12. Add a shard to your Sharded Cluster

Backup & Restore

Backups being critical to any production environment, ClusterControl has support for fully consistent backup and restore of your MongoDB replica set or sharded cluster.

Backups can be taken manually, or can be scheduled regularly or once off. Centralisation of backups is supported, with backups stored either on the Controller filesystem, including network-mounted directories, or uploaded to a preconfigured Cloud provider - currently supported providers are Google Cloud Platform and Amazon Web Services. This allows you to take full advantage of advanced lifecycle management functionality provided by Amazon and Google for such features as custom retention schedules, long-term archival, and encryption at rest, among others. Backup retention is configurable; you can choose to retain your backup for any time period, or to never delete backups. AES256 encryption is employed to secure your backups against rogue elements.

For rapid recovery, backups can be restored directly into the backed up cluster - ClusterControl handles the full restore process from launch to cluster recovery, removing error prone manual steps from the process.


Create Backup

Backup Details

Local time on controller host: **Fri Dec 22 2017 09:37:43 +00:00**

[✓ Simple] Advanced

Every: Week on Sunday at 0 : 0

Backup schedule for **00:00 on Sun (UTC)**

Backup

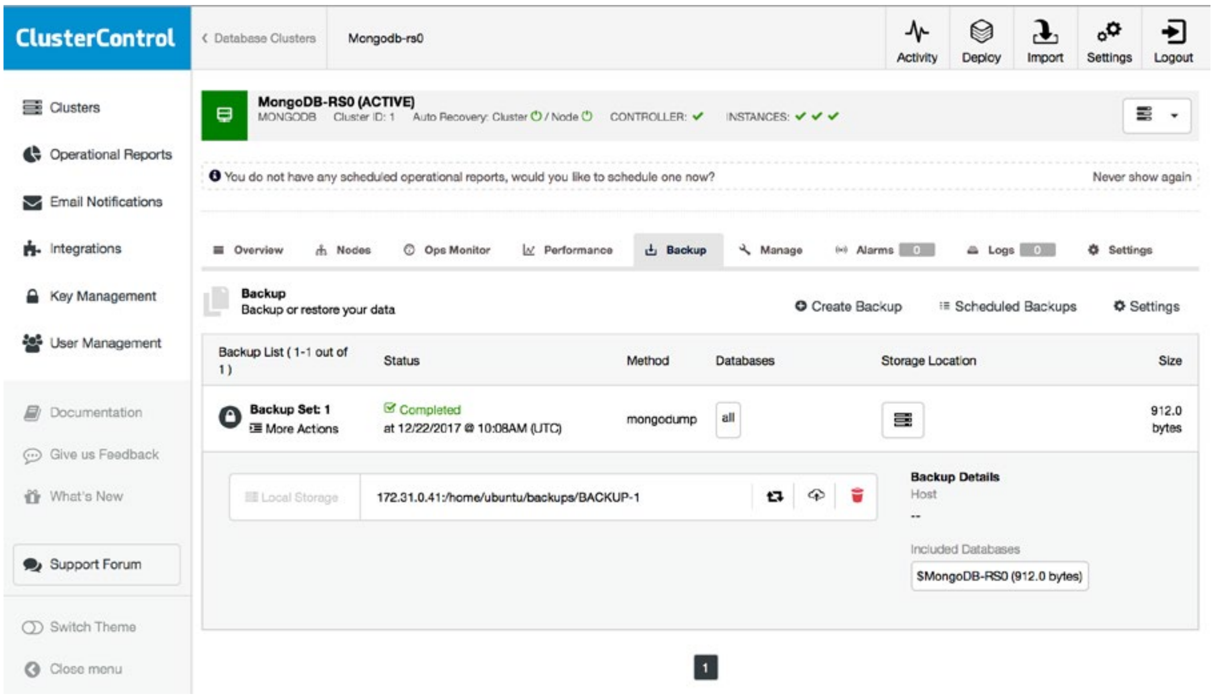
Backup Method: mongodump

Storage Directory: /home/ubuntu/backups

Upload Backup to the cloud New: ✓

Figure 13. Scheduling your backups

Existing backups can also be administered directly from the ClusterControl UI, as shown in Figure 14, below.



MongoDB-RS0 (ACTIVE)
MONGODB Cluster ID: 1 Auto Recovery: Cluster / Node CONTROLLER: ✓ INSTANCES: ✓ ✓ ✓

You do not have any scheduled operational reports, would you like to schedule one now? Never show again

Backup
Backup or restore your data

Backup List (1-1 out of 1)	Status	Method	Databases	Storage Location	Size
Backup Set: 1 More Actions	Completed at 12/22/2017 @ 10:08AM (UTC)	mongodump	all		912.0 bytes

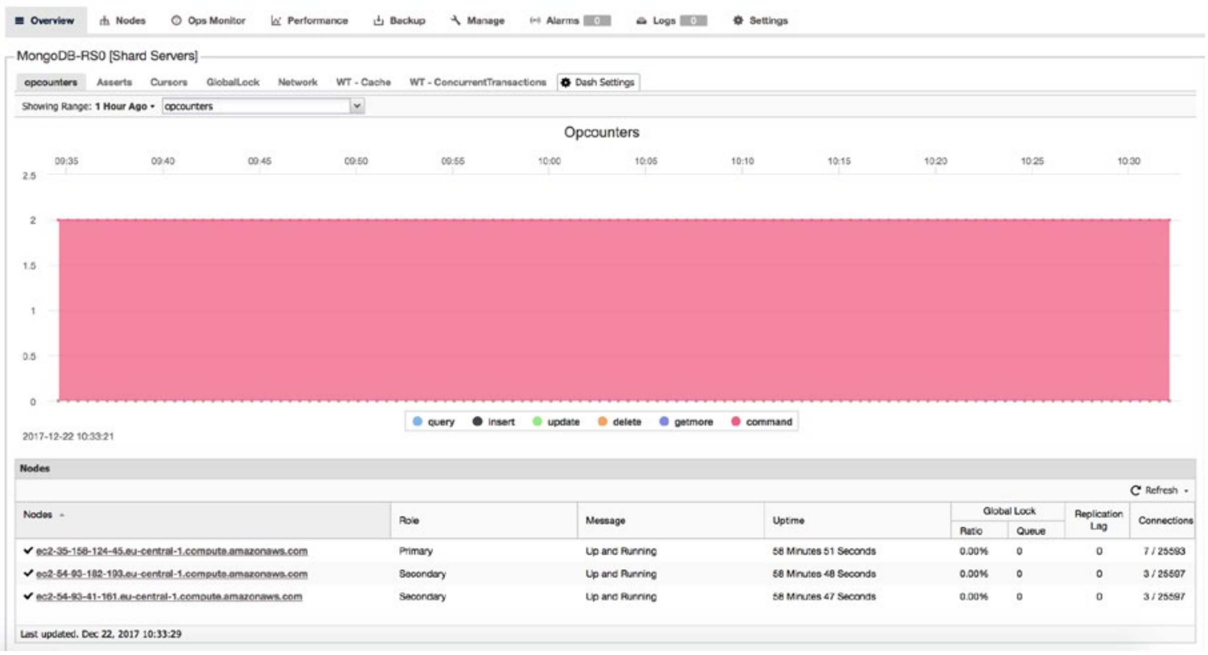
Local Storage 172.31.0.41:/home/ubuntu/backups/BACKUP-1

Backup Details
Host
--
Included Databases
$MongoDB-RS0 (912.0 bytes)

Figure 14. Manage existing backups

Monitoring

ClusterControl's enterprise monitoring functionality includes configurable custom dashboards capable of displaying any or all of the standard host metrics, as well as all key MongoDB performance metrics.



MongoDB-RS0 [Shard Servers]

Showing Range: 1 Hour Ago opcounters

2017-12-22 10:33:21

Nodes

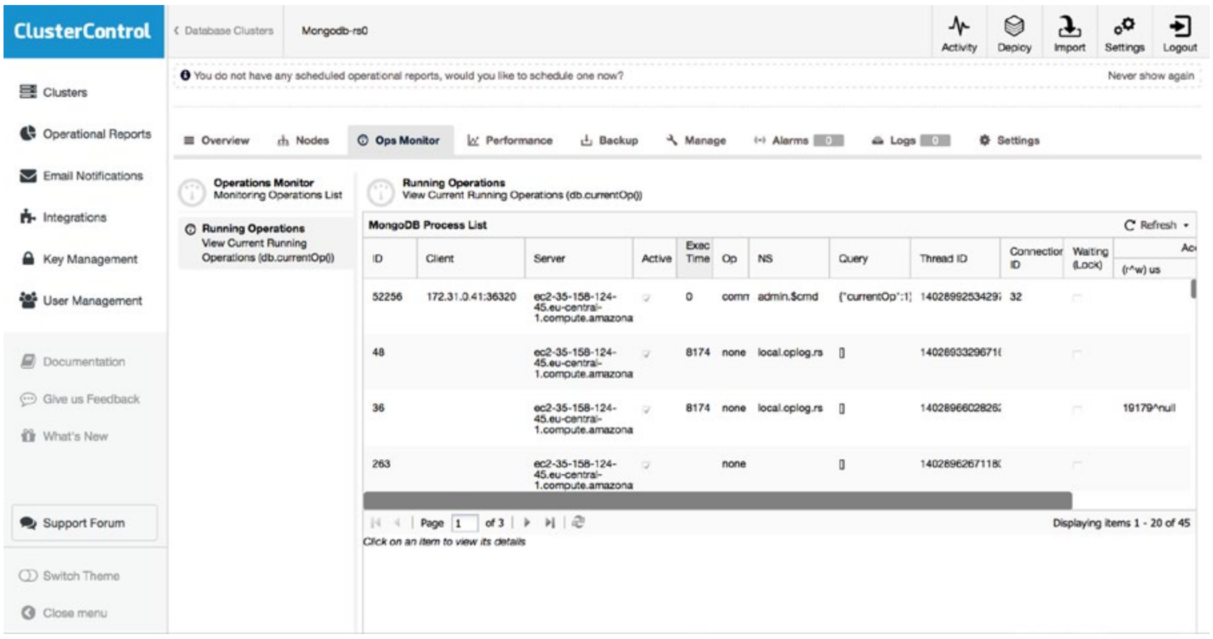
Nodes	Role	Message	Uptime	Global Lock		Replication Lag	Connections
				Ratio	Queue		
✓ ec2-35-158-124-45.eu-central-1.compute.amazonaws.com	Primary	Up and Running	58 Minutes 51 Seconds	0.00%	0	0	7 / 25593
✓ ec2-54-93-182-193.eu-central-1.compute.amazonaws.com	Secondary	Up and Running	58 Minutes 48 Seconds	0.00%	0	0	3 / 25597
✓ ec2-54-93-41-161.eu-central-1.compute.amazonaws.com	Secondary	Up and Running	58 Minutes 47 Seconds	0.00%	0	0	3 / 25597

Last updated. Dec 22, 2017 10:33:29

Figure 15. Overview

The default Overview includes a view of all MongoDB OpCounters, as well as tabs for Asserts, Cursors, GlobalLock, Network, WT-Cache, and WT-ConcurrentTransactions, givign a comprehensive view of your cluster's performance. This dashboard is fully editable, allowing the addition of any key metric from a list of dozens, as well as the facility to combine metrics for a complete holistic view of your deployment.

In addition, the Ops Monitor (Figure 16) gives a view of all running operations, helping track down any performance issues in your cluster.



You do not have any scheduled operational reports, would you like to schedule one now? Never show again

MongoDB Process List

ID	Client	Server	Active	Exec Time	Op	NS	Query	Thread ID	Connection ID	Waiting (Lock)	(r^w) us
52256	172.31.0.41:36320	ec2-35-158-124-45.eu-central-1.compute.amazona		0	comm	admin.$cmd	{"currentOp":1}	14028992534297	32		
48		ec2-35-158-124-45.eu-central-1.compute.amazona		8174	none	local.oplog.rs	[]	1402893329671(			
36		ec2-35-158-124-45.eu-central-1.compute.amazona		8174	none	local.oplog.rs	[]	1402896602826;			19179^null
263		ec2-35-158-124-45.eu-central-1.compute.amazona			none		[]	1402896267118(			

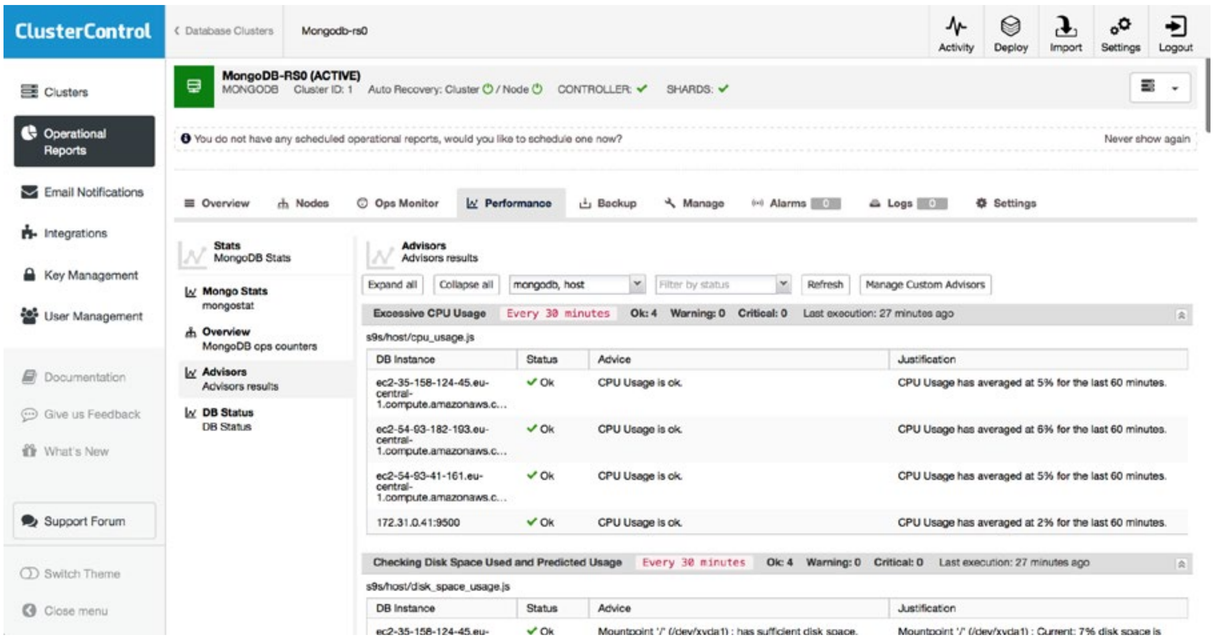
Page 1 of 3 Displaying items 1 - 20 of 45

Click on an item to view its details

Figure 16. Ops Monitor

MongoDB Advisors

Advanced monitoring functionality is included through ClusterControl "Advisors", found under the "Performance" tab, as shown in Figure 17. Advisors provide specific advice on how to address issues in areas such as performance, security, log management, configuration, storage space, and others.



MongoDB-RS0 (ACTIVE)
MONGODB Cluster ID: 1 Auto Recovery: Cluster / Node CONTROLLER: ✓ SHARDS: ✓

You do not have any scheduled operational reports, would you like to schedule one now? Never show again

Advisors
Advisors results

Excessive CPU Usage Every 30 minutes Ok: 4 Warning: 0 Critical: 0 Last execution: 27 minutes ago

s9s/host/cpu_usage.js

DB Instance	Status	Advice	Justification
ec2-35-158-124-45.eu-central-1.compute.amazonaws.c...	✓ Ok	CPU Usage is ok.	CPU Usage has averaged at 5% for the last 60 minutes.
ec2-54-93-182-193.eu-central-1.compute.amazonaws.c...	✓ Ok	CPU Usage is ok.	CPU Usage has averaged at 6% for the last 60 minutes.
ec2-54-93-41-161.eu-central-1.compute.amazonaws.c...	✓ Ok	CPU Usage is ok.	CPU Usage has averaged at 5% for the last 60 minutes.
172.31.0.41:9500	✓ Ok	CPU Usage is ok.	CPU Usage has averaged at 2% for the last 60 minutes.

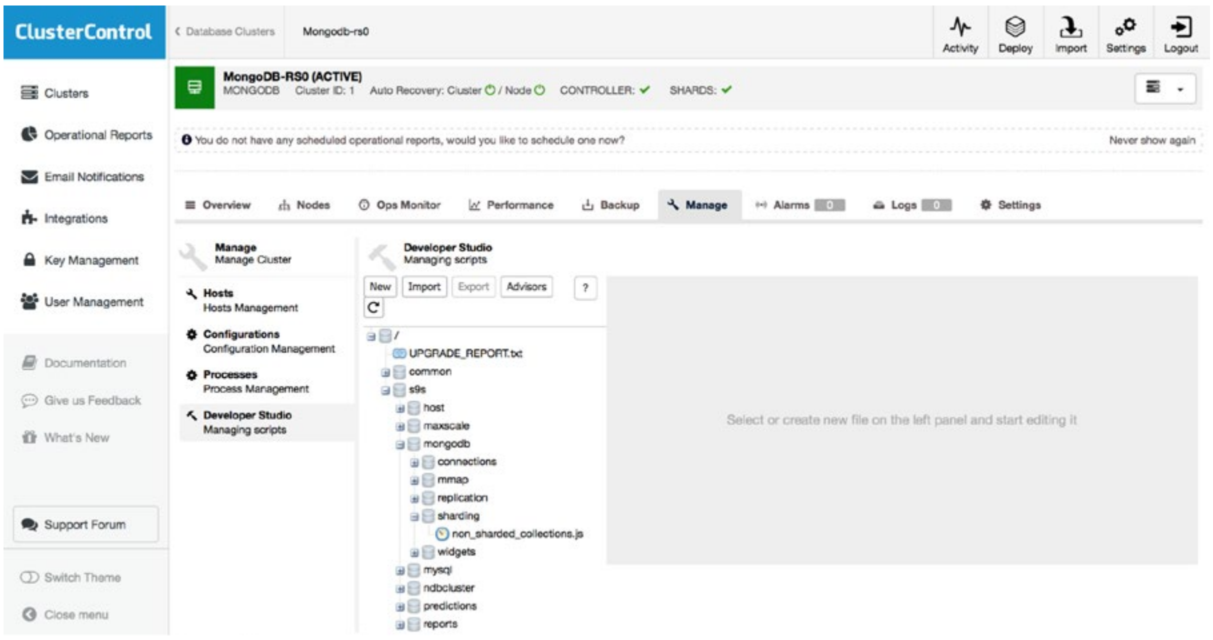
Checking Disk Space Used and Predicted Usage Every 30 minutes Ok: 4 Warning: 0 Critical: 0 Last execution: 27 minutes ago

s9s/host/disk_space_usage.js

DB Instance	Status	Advice	Justification
ec2-35-158-124-45.eu-	✓ Ok	Mountpoint '/' (/dev/xvda1) : has sufficient disk space.	Mountpoint '/' (/dev/xvda1) : Current: 7% disk space is

Figure 17. Advisors

Advisors are fully customisable through the built in Developer Studio (Figure 18), using the ClusterControl Domain-Specific Language (DSL), a Javascript-like language. With the ClusterControl DSL, new Advisors can be created, or existing Advisors can be extended for an experience fully tailored for your environment.



You do not have any scheduled operational reports, would you like to schedule one now?

Select or create new file on the left panel and start editing it

Figure 18. Developer Studio

While, as mentioned, you can create your own custom advisors, several are provided for key performance and management criteria out of the box. These include *Replication Lag* and *Replication Window*, critically important metrics for cluster resiliency, as well as *Error Detection*, an *Authentication/Authorisation Sanity Check* to aid in security management and the *Unsharded Databases and Collections Check* for sharded clusters, to ensure that everything that should be sharded, is sharded.

There are also more general purpose advisors such as the *Disk Mount Options* advisor, to ensure maximum performance of your cluster.

Integrations

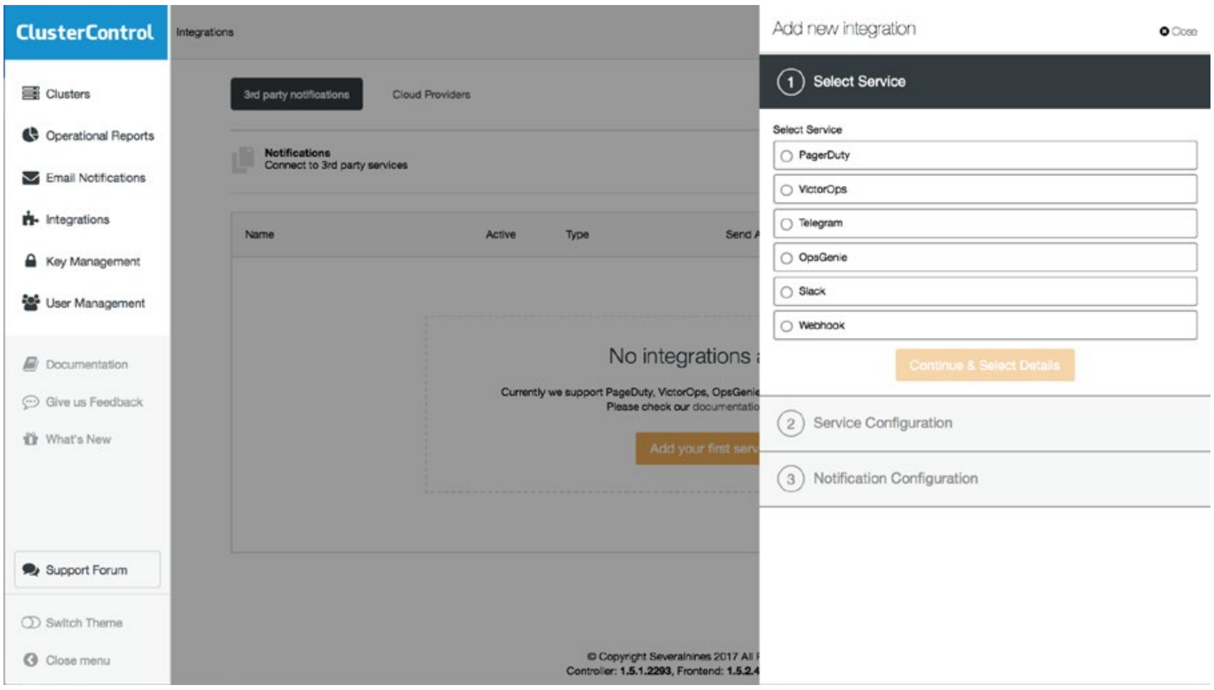


Figure 19. Adding integrations

For ease of integration into your existing corporate environment, ClusterControl supports integrations with commonly used products and technologies. Currently supported integrations are PagerDuty, VictorOps, OpsGenie, Slack, and Telegram. Integration of other applications is supported through Webhooks.

Command-Line Access

For ease of automation, ClusterControl also includes a commandline utility, ‘s9s’, which can also be installed from the website. This can be used to deploy, monitor, and backup ClusterControl resources, such as nodes, and clusters. In addition, s9s also allows you to manage users, and to view the status of running processes on ClusterControl-managed nodes.

Below are a few examples to get you started, and the documentation can be found on the website and also through context sensitive help in the tool itself.

Help

```
$ s9s --help
```

Usage

```
s9s COMMAND [OPTION...]
```

Where *COMMAND* is:
`account` - to manage accounts on clusters;
`backup` - to view, create and restore database backups;
`cluster` - to list and manipulate clusters;
`job` - to view jobs;
`maint` - to view and manipulate maintenance periods;
`metatype` - to print metatype information;
`node` - to handle nodes;
`process` - to view processes running on nodes;
`script` - to manage and execute scripts;
`server` - to manage hardware resources;
`user` - to manage users.

MongoDB deploy cluster example

The following command deploys a three-node MongoDB ReplicaSet by MongoDB Inc, with “replica_set_0”:

```
$ s9s cluster --create --cluster-type=mongodb --nod
es=”10.0.0.148;10.0.0.189;10.0.0.219” --vendor=10gen --pro-
vider-version=’3.2’ --os-user=root --db-admin=’admin’
--db-admin-passwd=’MyS3cr3tPass’ --cluster-name=’MongoDB
ReplicaSet 3.2’ --wait
```

MongoDB create backup example

The following command creates a backup using mongodump on 10.0.0.148 and stored it inside ClusterControl host under /storage/backups directory:

```
$ s9s backup --create --backup-method=mongodump --clus-
ter-name=’MongoDB ReplicaSet 3.2’ --nodes=10.0.0.148 --back-
up-directory=/storage/backups
```

MongoDB cluster status example:

The following command shows the status of the created MongoDB ReplicaSet:

```
$ s9s cluster --stat
MongoDB ReplicaSet 3.2
                                        Name: MongoDB Repli-
caSet 3.2              Owner: dba/users
```

```
      ID: 6                               State: STARTED
    Type: MONGODB                        Vendor: 10gen
3.2
  Status: All nodes are operational.
  Alarms:  0 crit   0 warn
    Jobs:  0 abort  0 defnd  0 dequd  0 faild  0 finsd  0
runng
  Config: ‘/etc/cmon.d/cmon_6.cnf’
 LogFile: ‘/var/log/cmon_6.log’

HOSTNAME    CPU   MEMORY   SWAP    DISK       NICs
10.0.0.148 1  6% 992M 139M 0B 0B 19G 15G  7.3K/s 33K/s
10.0.0.189 1  6% 992M 144M 0B 0B 19G 15G  8.8K/s 54K/s
10.0.0.219 1  5% 992M 143M 0B 0B 19G 15G  8.0K/s 51K/s
10.0.0.156 2 10% 3.6G 1.4G 0B 0B 19G 4.5G 301K/s 75K/s
```

MongoDB node status example

The following command returns configuration options for MongoDB node, 10.0.0.148:

```
$ s9s node --list-config --nodes=10.0.0.148
GROUP             OPTION NAME               VALUE
storage           dbPath                    /var/lib/mongodb
storage           journal.enabled           true
storage           engine                    wiredTiger
mmapv1            smallFiles                false
systemLog         destination               file
systemLog         path                      /var/log/mon-
godb/mongod.log
systemLog         logAppend                 true
net               port                      27017
processManagement fork                      true
processManagement pidFilePath               /var/run/mon-
godb/mongod.pid
setParameter      enableLocalhostAuthBypass true
replication       replSetName               replica_set_0
sharding          clusterRole               shardsvr
sharding          security.keyFile          /etc/mongo-clus-
ter.key
Total: 14
```

This is just a brief outline of s9s functionality, and many more operations are possible, as you can see above. The s9s command line tool facilitates scripting of many ClusterControl operations, allowing integration with your new and existing management utilities and scripts.

In Conclusion

In this Whitepaper, we have reviewed the challenges involved in managing MongoDB at scale and have introduced mitigating features of ClusterControl from Severalnines. As a best of breed database management solution, ClusterControl brings consistency and reliability to your database environment, and simplifies your database operations at scale.

About ClusterControl

ClusterControl is the all-inclusive open source database management system for users with mixed environments that removes the need for multiple management tools. ClusterControl provides advanced deployment, management, monitoring, and scaling functionality to get your MySQL, MongoDB, and PostgreSQL databases up-and- running using proven methodologies that you can depend on to work. At the core of ClusterControl is it's automation functionality that let's you automate many of the database tasks you have to perform regularly like deploying new databases, adding and scaling new nodes, running backups and upgrades, and more. Severalnines provides automation and management software for database clusters. We help companies deploy their databases in any environment, and manage all operational aspects to achieve high-scale availability.

About Severalnines

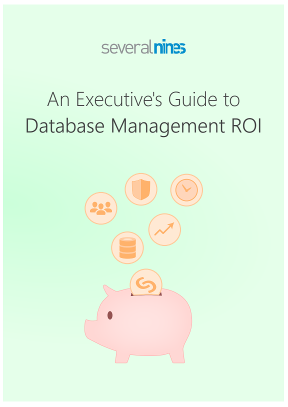
Severalnines provides automation and management software for database clusters. We help companies deploy their databases in any environment, and manage all operational aspects to achieve high-scale availability.

Severalnines' products are used by developers and administrators of all skills levels to provide the full 'deploy, manage, monitor, scale' database cycle, thus freeing them from the complexity and learning curves that are typically associated with highly available database clusters. Severalnines is often called the "anti-startup" as it is entirely self-funded by its founders. The company has enabled over 12,000 deployments to date via its popular product ClusterControl. Currently counting BT, Orange, Cisco, CNRS, Technicolor, AVG, Ping Identity and Paytrail as customers. Severalnines is a private company headquartered in Stockholm, Sweden with o ces in Singapore, Japan and the United States. To see who is using Severalnines today visit:

https://www.severalnines.com/company

Deploy Manage Monitor Scale

Related Whitepapers



An Executive's Guide to Database Management ROI

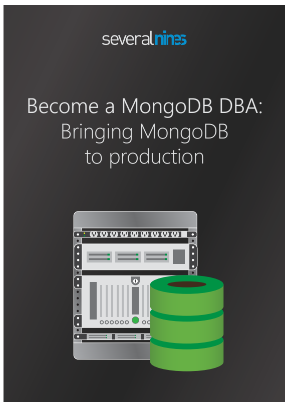
This guide discusses the options available to IT leaders when bringing in open source databases into their environments as well as general information on the open source database market. Also included in this whitepaper are an analysis of the costs of both doing and not doing select actions which are essential to managing open source databases.

Download here

DIY Cloud Database on Amazon Web Services: Best Practices

Over the course of this paper, we cover the details of AWS infrastructure deployment, considerations for deploying your database server(s) in the cloud, and finish with an example overview of how to automate the deployment and management of a MongoDB cluster using ClusterControl.

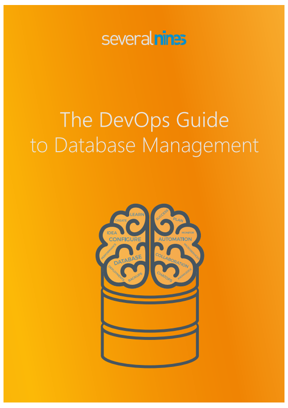
Download here



Become a MongoDB DBA: Bringing MongoDB to Production

Learn from our MongoDB experts what it takes to ensure your MongoDB stacks are production-ready. This whitepaper includes tips and tricks that we have collected from our best resources to help you deploy, monitor, manage and scale MongoDB in your environment.

Download here



The DevOps Guide to Database Management

Relational databases are not very flexible by nature, while DevOps is all about flexibility. This creates many challenges that need to be overcome. This white paper discusses three core challenges faced by DevOps when it comes to managing databases. It also discusses how Severalnines ClusterControl can be used to address these challenges.

Download here

severalnines

This white paper reviews the challenges involved in managing MongoDB at scale and introduces mitigating features of ClusterControl from Severalnines. As a best of breed database management solution, ClusterControl brings consistency and reliability to your database environment, and simplifies your database operations at scale.

Deploy

Manage

Monitor

Scale