## Contents

### About OpsCenter
- Key features................................................. 6

### Installing OpsCenter
- Installing OpsCenter on RHEL, CentOS, or Oracle Linux................................................. 8
- Installing OpsCenter on Debian or Ubuntu......................................................................... 9
- Installing the OpsCenter on Mac OSX or any Linux distributions................................. 10
- Installing OpsCenter agents............................................................................................... 12
  - Automatically deploying agents - Packaged installations........................................... 12
  - Manually deploying agents - Tarball installations......................................................... 13
  - Adding custom variables to agents............................................................................... 15
- OpsCenter and OpsCenter agent ports.................................................................................. 15
- Install locations.................................................................................................................... 16
  - Debian and Ubuntu Package install locations.............................................................. 16
  - Binary Tarball distribution install locations................................................................. 16
  - CentOS, OEL, and RHEL Package install locations..................................................... 17
- Starting, restarting, and stopping OpsCenter....................................................................... 17
- Starting and restarting OpsCenter Agents.......................................................................... 18

### Configuration
- Configuring user access....................................................................................................... 19
  - About user access roles................................................................................................. 19
  - Assigning or modifying user access roles................................................................. 20
  - Removing a user............................................................................................................ 20
- Configuring SSL.................................................................................................................... 21
  - Configuring SSL........................................................................................................... 21
  - Disabling SSL in Binary Tarball Installations - Linux and Mac OSX........................ 21
  - Enabling SSL in Windows installations..................................................................... 22
- Enabling HTTPS.................................................................................................................... 23
- Configuring for multiple regions....................................................................................... 24
- Configuring for multiple regions....................................................................................... 24
- Configuring events and alerts.............................................................................................. 25
  - Enabling email alerts.................................................................................................... 26
  - Enabling alerts posted to a URL................................................................................... 27
  - Verifying that events are posting correctly.................................................................. 28
- Configuring data collection and expiration....................................................................... 28
  - Estimating the amount of data generated................................................................... 28
  - Controlling data collection............................................................................................ 29
  - Storing collection data on a different cluster............................................................... 29
- Advanced configuration...................................................................................................... 30
  - OpsCenter configuration properties.......................................................................... 30
  - Cluster configuration properties................................................................................. 32

### Using OpsCenter
- Overview.............................................................................................................................. 34
- Node administration.......................................................................................................... 34
- Cluster administration....................................................................................................... 35
3.2............................................................................................................................................................ 67
3.1.1.......................................................................................................................................................... 67
3.1............................................................................................................................................................ 68
3.0.2.......................................................................................................................................................... 68
3.0............................................................................................................................................................ 68
2.1.3.......................................................................................................................................................... 69
2.1.2.......................................................................................................................................................... 69
2.1............................................................................................................................................................ 69
2.0............................................................................................................................................................ 69

Send us feedback....................................................................................................................................... 70
About OpsCenter

OpsCenter is a browser-based user interface for monitoring, administering, and configuring multiple clusters in a single centralized management console.

The OpsCenter agents are installed on the Real-time (Cassandra), Analytics (Hadoop), and Search (Solr) nodes. They use Java Management Extensions (JMX) to monitor and manage each node. Cassandra exposes a number of statistics and management operations through JMX. Using JMX, OpsCenter obtains metrics from a cluster and issues various node administration commands, such as flushing SSTables or doing a repair.

Key features

OpsCenter Enterprise Edition is bundled with DataStax support offerings. OpsCenter Community Edition is licensed for any type of use, but does not contain all the features available in the enterprise edition.

The key features of OpsCenter include:

- **Dashboard**
  - A Dashboard that displays an overview of commonly watched performance metrics
  - Adding your favorite graphs to the dashboard
  - An Overview that condenses the dashboards of multiple clusters (not visible when monitoring a single cluster)

- **Configuration and administration**
  - Basic [cluster configuration](#)
  - Administration tasks, such as adding a cluster, using simple point-and-click actions
  - Visual creation of clusters
  - Multiple cluster management from a single OpsCenter instance using [agents](#)
  - Rebalancing data across a cluster when new nodes are added
  - [Downloadable PDF cluster report](#)

- **Alerts and performance metrics**
  - Built-in external [notification](#) capabilities
  - [Alert](#) warnings of impending issues
  - Metrics are [collected](#) every minute from Cassandra, Analytics, and Search nodes and stored in a keyspace created by OpsCenter

- **Backup operations and restoring from backups**
• Automatic **backup operations**, including scheduling and removing of old backups
• **Restoring** from backups

**Database management and browsing**
• **Creating** and **managing** keyspaces and column families
• **Deleting a column family or truncating column families** to remove the data but not the column family itself
• Handling thousands of column families and **browsing data** in the database

**Other features**
• Optional **HTTPS support**
Installing OpsCenter

Installing OpsCenter on RHEL, CentOS, or Oracle Linux

Install the DataStax OpsCenter using Yum repositories on RedHat Enterprise Linux (RHEL), CentOS, and Oracle Linux (OL) distributions.

For a complete list of supported platforms, see DataStax OpsCenter — Supported Platforms.

- Yum package management utility.
- OpsCenter hardware requirements: computer with 2 cores and 2 GB RAM.
- Your Cassandra or DataStax Enterprise cluster is properly configured and running.
- IP address or hostname of the machine where you are installing OpsCenter.
- IP address or hostnames of two or three of your nodes.
- Port number used by JMX (default is 7199).
- The latest version of Oracle Java SE Runtime Environment (JRE) 6 or 7.
- OpenSSH (sshd)
- Python 2.6+

Note: If OpenSSL is version 1.0.0 is installed on RHEL 5.x, CentOS 5.x, OEL 5.5, Debian, or Ubuntu systems, you must install OpenSSL 0.9.8. See Error exceptions. ImportError:libssl.so.0.9.8.

After installing OpenSSL 0.9.8, Ubuntu 11.10 will still show 1.0.0.

To check for the software versions:

```
$ java -version
$ python -V
$ openssl version
```

Note: There are different package repositories for the Community and Enterprise versions of OpsCenter.

The CentOS, RHEL, and OL OpsCenter packaged releases create an opscenter user. OpsCenter runs as a service and runs as this user. The service initialization script is located in /etc/init.d. If the OpsCenter machine reboots, OpsCenter restarts automatically.

1. Ensure that a DSE or Cassandra cluster is installed, configured, and running. You can determine this by using the nodetool status command.

2. (CentOS 5.x/RHEL 5.x only) On the OpsCenter machine, make sure you have EPEL (Extra Packages for Enterprise Linux) installed. EPEL contains dependent packages, such as Python 2.6+. To install for both 32- and 64-bit systems:

   ```
   # rpm -Uvh http://dl.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm
   ```

   You do not have to install EPEL other machines.

3. Open the Yum repository specification /etc/yum.repos.d for editing. For example:

   ```
   # vi /etc/yum.repos.d/datastax.repo
   ```

4. In this file, add the repository for the edition you are installing. OpsCenter Enterprise Edition installations require the DataStax <username> and <password> you received in your registration confirmation email.

   - OpsCenter Community (free) Edition:
     ```
     [opscenter ]
     name = DataStax Repository
     ```
Installing OpsCenter

5. Install the OpsCenter package:
   • OpsCenter Community (free) Edition:
     # yum install opscenter-free
   • OpsCenter Enterprise (paid) Edition:
     # yum install opscenter

6. Install the OpsCenter package:
   • OpsCenter Community (free) Edition:
     sudo yum install opscenter-free
   • OpsCenter Enterprise (paid) Edition:
     sudo yum install opscenter

7. Set the [webserver] interface to the hostname or IP address of the OpsCenter machine in the /etc/opscenter/opscenterd.conf file. For example:

   [webserver]
   port = 8888
   interface = 10.183.170.161

8. Start the OpsCenter:
   sudo service opscenterd start

9. Connect to OpsCenter as described in Automatically deploying agents - Packaged installations on page 12.

   Note: You can also manually deploy agents. This is useful if an SSH connection does not exist between the agents and the OpsCenter machine, or as part of your node deployment process.

Installing OpsCenter on Debian or Ubuntu

Install the DataStax OpsCenter using APT repositories on Debian or Ubuntu distributions.

For a complete list of supported platforms, see DataStax OpsCenter – Supported Platforms.

Note: Use the exact version listed in Prerequisites. For example, the default OpenSSL on Ubuntu. If running openssl version shows that 1.0.0 is installed, you must install OpenSSL 0.9.8:

   • APT Package Manager is installed.
   • OpsCenter hardware requirements: computer with 2 cores and 2 GB RAM.
   • Your Cassandra or DataStax Enterprise cluster is properly configured and running.
   • IP address or hostname of the machine where you are installing OpsCenter.
   • IP address or hostnames of two or three of your nodes.
   • Port number used by JMX (default is 7199).
   • The latest version of Oracle Java SE Runtime Environment (JRE) 6 or 7.
   • OpenSSH (sshd)
   • Python 2.6+
   • OpenSSL: 0.9.8. (SSL is enabled by default.)
Installing OpsCenter

Note: If OpenSSL is version 1.0.0, you must install OpenSSL 0.9.8:

$ sudo apt-get install libssl0.9.8

After installing OpenSSL 0.9.8, Ubuntu 11.10 will still show 1.0.0.

To check for the software versions:

$ java -version
$ python -V
$ openssl version

Note: There are different package repositories for the Community and Enterprise versions of OpsCenter.

The OpsCenter Debian and Ubuntu packaged releases runs as a service from root. The service initialization script is located in /etc/init.d. If the machine reboots, OpsCenter restarts automatically.

1. Ensure that a DSE or Cassandra cluster is installed, configured, and running. You can determine this by using the nodetool status command.

2. Open the aptitude repository source list file (/etc/apt/sources.list) for editing. For example:

# vi/etc/apt/sources.list

3. In this file, add the repository for the edition you want to install. OpsCenter Enterprise Edition installations require the DataStax <username> and <password> you received in your registration confirmation email.

   • OpsCenter Community (free) Edition:
     deb http://debian.datastax.com/community stable main
   • OpsCenter Enterprise (paid) Edition:
     deb http://<username>:<password>@debian.datastax.com/enterprise stable main

4. Add the DataStax repository key to your aptitude trusted keys:

  # curl -L http://debian.datastax.com/debian/repo_key | apt-key add -

5. Install the OpsCenter package using the APT Package Manager:

   • OpsCenter Community (free) Edition:
     # apt-get update # apt-get install opscenter-free
   • OpsCenter Enterprise (paid) Edition:
     # apt-get update # apt-get install opscenter

6. Set the [webserver] interface to the hostname or IP address of the OpsCenter machine in the /etc/opscenter/opscenterd.conf file. For example:

   [webserver ] port = 8888
   interface = 10.183.170.161

7. Start OpsCenter:

   sudo service opscenterd start

8. Connect to OpsCenter as described in Automatically deploying agents - Packaged installations on page 12.

   Note: You can also manually deploy agents. This is useful if an SSH connection does not exist between the agents and the OpsCenter machine, or as part of your node deployment process.

Installing the OpsCenter on Mac OSX or any Linux distributions

Install the DataStax OpsCenter on Mac OSX or any Linux Distribution using the OpsCenter binary tarball.

For a complete list of supported platforms, see DataStax OpsCenter – Supported Platforms.
Installing OpsCenter

- OpsCenter hardware requirements: computer with 2 cores and 2 GB RAM.
- Your Cassandra or DataStax Enterprise cluster is properly configured and running.
- IP address or hostname of the machine where you are installing OpsCenter.
- IP address or hostnames of two or three of your nodes.
- Port number used by JMX (default is 7199).
- The latest version of Oracle Java SE Runtime Environment (JRE) 6 or 7.
- OpenSSH (sshd)
- Python 2.6+
  - (Linux only) SYSSTAT Utilities (for collection of I/O system metrics). When agents are installed from packaged installations, these utilities are installed automatically.
- The correct version of OpenSSL for your operating system.

Note: If you are installing on Ubuntu, you must use OpenSSL 0.9.8. To install:
$ sudo apt-get install libssl0.9.8

After installing OpenSSL 0.9.8, Ubuntu 11.10 will still show 1.0.0.

To check for the software versions:
$ java -version
$ python -V
$ openssl version
$ iostat -V

When you install OpsCenter using the binary tarball, an agent is deployed to the machine in which OpsCenter is installed. The installation also creates an agent tarball for deploying agents to other nodes in the cluster.

1. Ensure that a DSE or Cassandra cluster is installed, configured, and running. You can determine this by using the nodetool status command.

2. Download the tarball distribution of OpsCenter. You can either download the binary from Planet Cassandra or use the curl command:
   - OpsCenter Community (free) Edition:
     $ curl -OL http://downloads.datastax.com/community/opscenter.tar.gz
   - OpsCenter Enterprise (paid) Edition:
     $ curl -OL http://<username>:<password>@downloads.datastax.com/enterprise/opscenter.tar.gz

3. Unpack the distribution.
   $ tar -xzvf opscenter.tar.gz
   $ rm *.tar.gz

   Files for OpsCenter and a single OpsCenter agent are now in place.

4. Set the [webserver] interface (hostname or IP address of the OpsCenter machine) and any other required configuration properties for your environment in the opscenterd.conf file as described in OpsCenter configuration properties.

5. Start OpsCenter from the install location:
   $ bin/opscenter (Use -f to start in the foreground.)

6. The next step is to connect to OpsCenter and deploy the agent on the OpsCenter machine.

   Note: You can deploy only one agent using the OpsCenter console (on the machine where OpsCenter is installed). Deploying agents to other nodes in the cluster must be done manually. You cannot deploy multiple agents to other nodes automatically (by clicking Fix in the OpsCenter console).
Installing OpsCenter agents

Various methods for installing the agents.

Automatically deploying agents - Packaged installations

Quick and easy installation when you have installed OpsCenter from a package, or for installing the agent on the OpsCenter machine from a binary tarball.

This method uses the OpsCenter Dashboard to install agents. It provides an SSH connection between the agent and OpsCenter and requires login credentials.

- Root or sudo access to the machines where the agents will be installed.
- Your Cassandra or DataStax Enterprise cluster is up and running.
- OpsCenter is installed and configured.
- JMX connectivity is enabled on each node in the cluster.
- You either configured the SSH port or you will accept the default SSH port (22) for node-agent communications.

1. Open a browser window and go to the OpsCenter URL at http://<opscenter_host>:8888 where <opscenter_host> is the IP or hostname of the OpsCenter machine. For example:
   - http://110.123.4.5:8888
2. In Add Cluster, enter the Hostnames or IP addresses of two or three nodes in the cluster and set the JMX set JMX and Thrift ports credentials, and then click Save Cluster.

   After OpsCenter connects to the cluster, a Fix link appears near the top of the Dashboard.
3. Start installing the agents by clicking the Fix.

4. In Install Node Agents, click Enter Credentials.
5. In Node SSH Credentials, enter a username that has root privileges or sudo access to all of the nodes in your cluster, plus any other required credentials, and then click Done.

6. In the Install Nodes Agent dialog, click Install on all nodes.

7. If prompted, click Accept Fingerprint to add a node to the known hosts for OpsCenter.

It takes a few minutes for OpsCenter to complete the agent installations. After all agents are installed, a success message is displayed.

**Manually deploying agents - Tarball installations**

Install agents on nodes running tarball installations of Cassandra or DataStax Enterprise clusters.

The agent where OpsCenter is installed is automatically installed; it is not installed on the other nodes in the cluster. Only after starting OpsCenter, is the agent.tar.gz file installed in the OpsCenter installation folder. Use this file to manually deploy the agents to other nodes in a cluster.

- Your Cassandra or DataStax Enterprise cluster is up and running.
- OpsCenter is installed and configured.
- JMX connectivity is enabled on each node in the cluster.
- SYSSTAT Utilities (needed for the collection of I/O metrics).

1. Copy the agent tar file from your existing OpsCenter installation directory to your cluster node. For example, to copy to the install location:

   
   ```
   $ cd <install_location>
   $ scp agent.tar.gz <user>@<node_IP>:/<install_location>
   ```

   **Note:** The agent.tar.gz file is not created until you run OpsCenter for the first time.

2. Log in to the node, copy the agent.tar.gz file to the desired location, and unpack it. For example:

   ```
   $ ssh user@<node_IP>
   $ cd <install_location>
   $ tar -xzf agent.tar.gz
   ```
Installing OpsCenter

The binary package creates and agent directory containing the installation files.

3. Configure and start the agent:
   a) If needed, get the IP address of the OpsCenter host. On the node containing the OpsCenter:
      
      ```
      $ hostname -i
      ```

   b) On the node where you are installing the agent:
      
      ```
      $ cd agent
      $ bin/setup <opscenter_host>
      ```

      **Note:** Generally the agent can detect the listener IP address for the node, which is the IP
      address displayed for node by running `nodetool ring -h localhost`. If needed, add
      `<node_listen_address>` to the above command.

4. Start the OpsCenter agent:
   
   ```
   $ <install_location> bin/opscenter-agent (Use `-f` to start in the foreground.)
   ```

**Manually deploying agents - Packaged installations**

Deploy agents without an SSH connection between the agents and the OpsCenter machine, or to install
the agents as part of your node deployment process.

If you installed OpsCenter on a cluster node using a package, you can deploy agents on any supported
CentOS, Debian, OEL, RHEL, or Ubuntu nodes. After installation, the agent runs as a service that starts
when the machine boots up and restarts automatically.

- Root or sudo access to the machines where the agents will be installed.
- Your Cassandra or DataStax Enterprise cluster is up and running.
- OpsCenter is installed and configured.
- JMX connectivity is enabled on each node in the cluster.
- If you install from the binary tarball described below, you must install the SYSSTAT utility (needed for
  the collection of I/O metrics).

1. On the OpsCenter machine, go to the `opscenter` directory:
   
   ```
   $ cd /usr/share/opscenter
   ```

2. Copy the agent software to the home directory in your cluster node. For example:
   
   ```
   $ scp agent.tar.gz <user>@<node_IP>:~/
   ```

3. Log in to the node, go to the home directory, and unpack it. For example:
   
   ```
   $ ssh <user>@<node_IP>
   $ cd ~/
   $ tar -xzf agent.tar.gz
   ```

   The binary package creates and agent directory containing the installation files.

4. If you **have** sudo access, installing the agent using the package is recommended:
   a) If needed, get the IP address of the OpsCenter host. On the node containing the OpsCenter:
      
      ```
      $ hostname -i
      ```

   b) On the node where you are installing the agent, go to the agent directory:
      
      ```
      $ cd agent
      ```

   c) Install the agent:
      
      **RHEL installs:**
      ```
      $ sudo bin/install_agent.sh -s opscenter-agent.rpm <opscenter_host>
      ```

      **Debian installs:**
      ```
      $ sudo bin/install_agent.sh -s opscenter-agent.deb <opscenter_host>
      ```
Installing OpsCenter

Note: Generally the agent can detect the listener IP address for the node, which is the IP address displayed for node by running `nodetool ring -h localhost`. If needed, add `<node_listen_address>` to the above command.

5. If you do not have sudo access:
   a) If needed, get the IP address of the OpsCenter host. On the node containing the OpsCenter:
      
      ```
      $ hostname -i
      ```
   b) On the node where you are installing the agent, go to the agent directory:
      
      ```
      $ cd agent
      ```
   c) Run the setup command and install the agent:
      
      ```
      $ bin/setup <opscenter_host>
      ```
      
      Note: Generally the agent can detect the listener IP address for the node, which is the IP address displayed for node by running `nodetool ring -h localhost`. If needed, add `<node_listen_address>` to the above command.

6. Start the OpsCenter agent:
   
   ```
   $ <install_location> bin/opscenter-agent
   ```
   (Use `-f` to start in the foreground.)

Adding custom variables to agents

OpsCenter agents do not pick up the environment variables of the currently logged-in user by default. For example, if Java is not in the machine's PATH, you may notice errors in the agent log on start-up:

```bash
nohup: cannot run command 'java': No such file or directory
```

- On the Cassandra nodes where the agents are installed, create the file `/etc/default/opscenter-agent` and set the environment variables for `JAVA_HOME` and any other custom environment variables that the agent may need. For example:

```bash
JAVA_HOME =/usr/bin/java
```

OpsCenter and OpsCenter agent ports

A list of the default port numbers used by OpsCenter and the OpsCenter Agents:

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>OpsCenter ports</strong></td>
</tr>
<tr>
<td>8888</td>
<td>OpsCenter website. The opscenterd daemon listens on this port for HTTP requests coming directly from the browser. Configurable in <code>opscenterd.conf</code>.</td>
</tr>
<tr>
<td>50031</td>
<td>OpsCenter HTTP proxy for Job Tracker. The opscenterd daemon listens on this port for incoming HTTP requests from the browser when viewing the Hadoop Job Tracker page directly. (DataStax Enterprise only)</td>
</tr>
<tr>
<td>61620</td>
<td>OpsCenter monitoring port. The opscenterd daemon listens on this port for TCP traffic coming from the agent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th><strong>OpsCenter agents ports (on the monitored nodes)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>7199</td>
<td>JMX monitoring port. Each agent opens a JMX connection to its local node (the Cassandra or DataStax Enterprise process listening on this port). The JMX protocol requires that the client then reconnect on a randomly chosen port (1024+) after the initial handshake.</td>
</tr>
</tbody>
</table>
## Installing OpsCenter

### Port Description

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8012</td>
<td>Hadoop Job Tracker client port. The Job Tracker listens on this port for job submissions and communications from task trackers; allows traffic from each Analytics node in a DataStax Enterprise cluster.</td>
</tr>
<tr>
<td>8012</td>
<td>Hadoop Job Tracker Thrift port. The Job Tracker listens on this port for Thrift requests coming from the opscenterd daemon. (DataStax Enterprise only)</td>
</tr>
<tr>
<td>8012</td>
<td>Hadoop Job Tracker website port. The Job Tracker listens on this port for HTTP requests. If initiated from the OpsCenter UI, these requests are proxied through the opscenterd daemon; otherwise, they come directly from the browser. (DataStax Enterprise only)</td>
</tr>
<tr>
<td>8012</td>
<td>Hadoop Task Tracker website port. Each Task Tracker listens on this port for HTTP requests coming directly from the browser and not proxied by the opscenterd daemon. (DataStax Enterprise only)</td>
</tr>
<tr>
<td>61621</td>
<td>OpsCenter agent port. The agents listen on this port for SSL traffic initiated by OpsCenter.</td>
</tr>
<tr>
<td>22</td>
<td>SSH port. Configurable in <code>opscenterd.conf</code>.</td>
</tr>
</tbody>
</table>

### Solr Port and Demo applications port

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8983</td>
<td>Solr Port and Demo applications port.</td>
</tr>
</tbody>
</table>

### Cassandra client port

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9160</td>
<td>Each agent makes Thrift requests to its local node on this port. Additionally, the port can be used by the opscenterd daemon to make Thrift requests to each node in the cluster.</td>
</tr>
</tbody>
</table>

## Install locations

File locations for each type of installation.

### Debian and Ubuntu Package install locations

File locations for Debian and Ubuntu package installs.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/var/lib/opscenter</td>
<td>SSL certificates for encrypted agent/dashboard communications</td>
</tr>
<tr>
<td>/var/log/opscenter</td>
<td>Log directory</td>
</tr>
<tr>
<td>/var/run/opscenter</td>
<td>Runtime files</td>
</tr>
<tr>
<td>/usr/share/opscenter</td>
<td>JAR, agent, web application, and binary files</td>
</tr>
<tr>
<td>/etc/opscenter</td>
<td>Configuration files</td>
</tr>
<tr>
<td>/etc/init.d</td>
<td>Service start-up script</td>
</tr>
</tbody>
</table>

### Binary Tarball distribution install locations

File locations for binary-based installs.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>/agent</td>
<td>Agent installation files</td>
</tr>
</tbody>
</table>
Installing OpsCenter

<table>
<thead>
<tr>
<th>Directory</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>/bin</td>
<td>Startup and configuration binaries</td>
</tr>
<tr>
<td>/content</td>
<td>Web application files</td>
</tr>
<tr>
<td>/conf</td>
<td>Configuration files</td>
</tr>
<tr>
<td>/doc</td>
<td>License files</td>
</tr>
<tr>
<td>/lib and /src</td>
<td>Library files</td>
</tr>
<tr>
<td>/log</td>
<td>OpsCenter log files</td>
</tr>
<tr>
<td>/ssl</td>
<td>SSL files for OpsCenter to agent communications</td>
</tr>
</tbody>
</table>

**CentOS, OEL, and RHEL Package install locations**

File locations for RHEL-based package installs.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>/var/lib/opscenter</td>
<td>SSL certificates for encrypted agent/dashboard communications</td>
</tr>
<tr>
<td>/var/log/opscenter</td>
<td>Log directory</td>
</tr>
<tr>
<td>/var/run/opscenter</td>
<td>Runtime files</td>
</tr>
<tr>
<td>/usr/share/opscenter</td>
<td>JAR, agent, web application, and binary files</td>
</tr>
<tr>
<td>/etc/opscenter</td>
<td>Configuration files</td>
</tr>
<tr>
<td>/etc/init.d</td>
<td>Service startup script</td>
</tr>
</tbody>
</table>

**Starting, restarting, and stopping OpsCenter**

Commands for each type of installation.

Packaged installations include startup scripts for running OpsCenter as a service. The available `service opscenterd` options are:

```
service opscenterd start|stop|status|restart|force-reload
```

The following list shows start, stop, and restart instructions for the supported platforms:

- To start DataStax OpsCenter:
  - **Packaged installs**: `sudo service opscenterd start`
  - **Binary installs**: `bin/opscenter` (Use `-f` to start in the foreground.)
  - **Windows installs**: Start the OpsCenter Service from the Control Panel.

  **Note**: By default, DataStax Enterprise services on Windows start automatically.

- To stop DataStax OpsCenter:
  - **Packaged installs**: `sudo service opscenterd stop`
  - **Binary installs**: Find the OpsCenter Java process ID (PID) and kill the process using its PID number:
    ```
    ps -ef | grep opscenter
    sudo kill <pid>
    ```
  - **Windows installs**: Stop the OpsCenter Service from the Control Panel.

- To restart DataStax OpsCenter:
  - **Packaged installs**: `sudo service opscenterd restart`
Installing OpsCenter

- Binary installs: Find the OpsCenter Java process ID (PID), kill the process using its PID number, and then start the OpsCenter:
  
  ```
  ps -ef | grep opscenter
  sudo kill <pid>
  bin/opscenter (Use -f to start in the foreground.)
  ```

- Windows installs: Restart the OpsCenter Service from the Control Panel.

Starting and restarting OpsCenter Agents

Commands for each type of installation.

- To start the OpsCenter agent:
  
  - Packaged installs: The OpsCenter Agent starts automatically.
  - Binary installs: `$ <install_location> bin/opscenter-agent (Use -f to start in the foreground.)`
  - Windows installs: Start the OpsCenter Agent Service from the Control Panel.

- To restart OpsCenter agent:
  
  - Packaged installs: `$ sudo service opscenter-agent restart`
  - Binary installs: Find the OpsCenter agent Java process ID (PID), kill the process using its PID number, and then start the OpsCenter:
    
    ```
    ps -ef | grep opscenter-agent
    sudo kill <pid>
    $ bin/opscenter-agent (Use -f to start in the foreground.)
    ```
  
  - Windows installs: Restart the OpsCenter Agent Service from the Control Panel.
Configuration

Configuring user access

By default, access control is disabled. Any user that knows the OpsCenter URL can view all objects and perform all tasks. To control access, you configure authentication for OpsCenter users by performing these tasks:

- Assign passwords.
- Add users.
- Set access roles using the `set_passwd.py` utility.

About user access roles

OpsCenter provides two access roles: admin and user.

Admin role privileges

- Alerts
  - add
  - delete
  - modify
- Cluster operations
  - add nodes to a cluster
  - configure the cluster (all at once rather than a single node at a time)
  - rebalance
  - restart the cluster
- Column families
  - add column metadata
  - create
  - delete column metadata
  - delete index
  - drop
  - truncate
  - modify
- Keyspaces
  - create
  - drop
  - modify
- Node
  - cleanup
  - compact
  - configure
  - decommission
  - drain
  - flush
  - move
  - perform garbage collection
Configuration

- repair
- restart
- start
- stop
- Install the OpsCenter agent on Cassandra nodes
- OpsCenter configuration
  - add an existing cluster to OpsCenter
  - delete a cluster from OpsCenter
  - edit the config for a cluster
- OpsCenter is monitoring
- Provisioning
  - add nodes to an existing cluster
  - provision a new cluster (local or EC2)
- Run a one-off backup
- Run a restore of a backup
- Scheduled backups
  - add
  - delete
  - modify

User role privileges

Users assigned the user role can perform all other OpsCenter tasks.

Assigning or modifying user access roles

The first time you assign an access role to an administrator or user, OpsCenter generates a password file and enables access control. Authentication is required to access OpsCenter for viewing objects and performing tasks.

To create or modify access roles:

1. Run the set_passwd.py utility.
   For example, to create user johndoe with admin role privileges:

   ```
   $ python /usr/share/opscenter/bin/set_passwd.py johndoe admin
   Please enter a password for 'johndoe'.
   Password:
   ```

2. After configuring authentication, restart OpsCenter.
   $ service opscenterd restart

   Restarting is required only when you create the first user (because it enables access control). No restart is required for adding, modifying, or removing users.

Removing a user

To remove a user:

1. Edit the OpsCenter password file:
   - Packaged installs: /etc/opscenter/.passwds
   - Binary installs: <install_dir>/passwds

2. Delete the line of the user that you want to remove (<username>:<password_hash>:<role>).
   johndoe:5e8848...42d8:admin
Restarting is not required to remove a user. Restarting is required to delete the password file. Deleting the password file disables access control. If you delete all users, you will not be able to access OpsCenter.

## Configuring SSL

### Configuring SSL

OpsCenter uses Secure Socket Layer (SSL) to encrypt the communication protocol and authenticate traffic between OpsCenter agents and the main OpsCenter daemon (Linux and Mac OS X) or the DataStax OpsCenter Service (Windows).

The default SSL state depends on the operating system:

- Linux and Mac OS X: enabled. To disable, see Disabling SSL in Binary Tarball Installations - Linux and Mac OS X on page 21
- Windows: disabled. To enable, see Enabling SSL in Windows installations on page 22

Consider disabling SSL if you are running OpsCenter and DataStax Enterprise or DataStax Community under the following conditions:

- On a secure internal network.
- In a development environment where agents and OpsCenter run on the same computer free from network threats.
- In a situation where you are not concerned about someone listening to OpsCenter traffic.
- In automatic deployments of OpsCenter to avoid re-installation of agents. (Unless you disable SSL, installing OpsCenter generates SSL files for encryption and requires re-installation of agents.)
- On a computer that does not have the required version of OpenSSL.

If you have no need for SSL, you can simply disable the SSL option to avoid installing OpenSSL.

### SSL requirements

When the SSL option is enabled, OpsCenter requires a specific version of OpenSSL for each supported operating system.

<table>
<thead>
<tr>
<th>Version</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9.8</td>
<td>CentOS 5.x, Debian, Mac OS X, Oracle Linux 5.5, RHEL 5.x, SUSE Enterprise 11.x, Ubuntu, and Windows</td>
</tr>
<tr>
<td>1.0.0</td>
<td>CentOS 6.x, Oracle Linux 6.1, and RHEL 6.x</td>
</tr>
</tbody>
</table>

To determine which version of OpenSSL is installed on a Linux or Mac OS X system, use the following command:

```
openssl version
```

### Disabling SSL in Binary Tarball Installations - Linux and Mac OS X

SSL is enabled on Linux and Mac OS X installations by default.

To disable SSL on Linux and Mac OS X, you need to modify the configuration files for OpsCenter and its agents.

**On the OpsCenter machine:**

1. Go to the directory containing the OpsCenter configuration file (`opscenterd.conf`):
   - Packaged installs: `cd /etc/opscenter`
   - Binary installs: `cd /<install_location`
Configuration

2. Open \texttt{opscenterd.conf}, for editing. For example:
   \texttt{sudo vi opscenterd.conf}

3. Add the following to \texttt{opscenterd.conf}:
   \begin{verbatim}
   [agents]
   use_ssl = false
   \end{verbatim}

4. Restart OpsCenter:
   \begin{itemize}
   \item Packaged installs: \texttt{sudo service opscenterd restart}
   \item Binary installs: Find the OpsCenter Java process ID (PID), kill the process using its PID number, and then start the OpsCenter:
     \begin{verbatim}
     ps -ef | grep opscenter
     sudo kill <pid>
     bin/opscenter \texttt{(Use -f to start in the foreground.)}
     \end{verbatim}
   \end{itemize}

On the agent machine:

5. Go to the directory containing the OpsCenter agent configuration file (\texttt{address.yaml}):
   \texttt{cd /<install_location>/conf}

6. Open \texttt{address.yaml} for editing. For example:
   \texttt{sudo vi address.yaml}

7. Add the following command and set its value to 0.
   \begin{verbatim}
   use_ssl: 0
   \end{verbatim}

8. Restart OpsCenter:
   \begin{itemize}
   \item Packaged installs: \texttt{sudo service opscenterd restart}
   \item Binary installs: Find the OpsCenter Java process ID (PID), kill the process using its PID number, and then start the OpsCenter:
     \begin{verbatim}
     ps -ef | grep opscenter
     sudo kill <pid>
     bin/opscenter \texttt{(Use -f to start in the foreground.)}
     \end{verbatim}
   \end{itemize}

Enabling SSL in Windows installations

By default, SSL is disabled on Windows installations.

To enable SSL, you run \texttt{setup.py} (which generates the required SSL keys and certificates), modify the configuration files for OpsCenter and its agent, and then restart the DataStax OpsCenter Agent Service.

1. Go to the opscenter\bin directory:
   \begin{verbatim}
   Program Files (x86) > DataStax Community > opscenter > bin
   \end{verbatim}

2. Click or double-click \texttt{setup.py} to run it.
   The agentKeyStore key pairs are generated and appear in \texttt{opscenter\ssl} directory.

3. Go to the opscenter\conf directory:
   \begin{verbatim}
   DataStax Community > opscenter > conf
   \end{verbatim}

4. Open the configuration file for OpsCenter, opscenterd.conf, in a text editor such as Notepad.

5. In the agents section, change \texttt{use_ssl} from 0 to 1 (or \texttt{true}), and then save the file.
6. Go to the `opscenter\agent\conf` directory:
   DataStax Community > opscenter > agent > conf
7. Open the configuration file for OpsCenter agent, `address.yaml`, in a text editor.
8. In the `address.yaml` file, change the value for `use_ssl` from 0 to 1, and then save the file.
   ```yaml
   use_ssl: 1
   ```
9. From the Control Panel, restart the DataStax OpsCenter Agent Service.

---

**Enabling HTTPS**

You can enable or disable Hypertext Transfer Protocol Secure (HTTPS) support in OpsCenter.

1. Open the OpsCenter configuration file, `opscenterd.conf`, located in one of these directories:
   - **Package installations:** `/etc/opscenter/opscenterd.conf`
   - **Binary tarball installations (Linux and Mac OSX):** `<install_location>/conf/opscenterd.conf`
   - **Windows installations:** `Program Files (x86)\DataStax Community\opscenter\conf\opscenterd.conf`
2. Scroll to the `[webserver]` section.

   This snippet from `opscenterd.conf` shows the `[webserver]` section that you change:

   ```conf
   [webserver]
   port = 8888
   interface = 127.0.0.1
   # The following settings can be used to enable ssl support for the opscenter
   # web application. Change these values to point to the ssl certificate and key
   ```
# that you wish to use for your OpsCenter install, as well as the port you
would like
# to serve ssl traffic from.
#ssl_keyfile = /var/lib/opscenter/ssl/opscenter.key
#ssl_certfile = /var/lib/opscenter/ssl/opscenter.pem
#ssl_port = 8443

3. Remove the comment markers (#) in front of ssl_keyfile, ssl_certfile, and ssl_port.
   You can use the default values for the ssl_keyfile and ssl_certfile or replace them with the path to your
   own private and public certificates.
4. Save opscenterd.conf and restart OpsCenter.

Configuring for multiple regions

OpsCenter 2.1 and later can operate in multiple regions or IP forwarding deployments. Use the following
approach for deployments where a public IP forwards to a private IP on the agent, but that machine is not
aware of (that is, can’t bind to) the public IP.

To configure OpsCenter agents for multiple regions or IP forwarding:
1. Open the address.yaml file for editing.
   • Package installations: /var/lib/opscenter-agent/conf directory
   • Binary tarball installations (Linux and Mac OSX): <install_location>/conf directory
2. Add the following option to the address.yaml:
   • local_interface: (Optional) The IP used to identify the node. If broadcast_address is set
     in cassandra.yaml, this should be the same as that; otherwise, it is typically the same as
     listen_address in cassandra.yaml. A good check is to confirm that this address is the same as the
     address that nodetool ring outputs.
   • agent_rpc_interface: The IP that the agent HTTP server listens on. In a multiple region deployment,
     this is typically a private IP.
   • agent_rpc_broadcast_address: The IP that the central OpsCenter process uses to connect to the
     agent.
3. Repeat the above steps for each node.

here is the configuration for a three node cluster that spans two regions:

| Region: us-west | Availability Zone: us-west-2 |
| Node1 | public IP: 198.51.100.1 | private IP: 10.11.12.1 | Cassandra (cassandra.yaml ) | broadcast_address: 198.51.100.1 | listen_address: 10.11.12.1 | Agent (address.yaml ) | local address: 198.51.100.1 | agent_rpc_interface: 10.11.12.1 | agent_rpc_broadcast_address: 198.51.100.1 | OpsCenter (opscenterd.conf ) | interface: 198.51.100.1 |
| Node2 | public IP: 198.51.100.23 |
private IP: 10.11.12.15
Cassandra (cassandra.yaml)
  broadcast_address: 198.51.100.23
  listen_address: 10.11.12.15
Agent (address.yaml)
  local_address: 198.51.100.23
  agent_rpc_interface: 10.11.12.15
  agent_rpc_broadcast_address: 198.51.100.23

Region: us-east
Availability Zone: us-east-1

Node1
  public IP: 203.0.113.20
  private IP: 10.11.13.28
Cassandra (cassandra.yaml)
  broadcast_address: 203.0.113.20
  listen_address: 10.11.13.28
Agent (address.yaml)
  local_address: 203.0.113.20
  agent_rpc_interface: 10.11.13.28
  agent_rpc_broadcast_address: 203.0.113.20

Configuring events and alerts

The OpsCenter Event Log page displays a continuously updated list of events and alerts.

The following list reflects the most detailed logging level available for Cassandra, DataStax Enterprise,
and OpsCenter events:

- DEBUG(0)
- INFO (1)
- WARN (2)
- ERROR (3)
- CRITICAL (4)
- ALERT (5)

Events

Data for these events is stored in the events and events_timeline column families in the OpsCenter
keyspace:

<table>
<thead>
<tr>
<th>Event</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPACTION</td>
<td>0</td>
<td>Major compaction has occurred.</td>
</tr>
<tr>
<td>CLEANUP</td>
<td>1</td>
<td>Unused keys have been removed or cleaned up.</td>
</tr>
<tr>
<td>REPAIR</td>
<td>2</td>
<td>A repair operation has been initiated.</td>
</tr>
<tr>
<td>FLUSH</td>
<td>3</td>
<td>Memtables have been flushed to disk.</td>
</tr>
<tr>
<td>DRAIN</td>
<td>4</td>
<td>The commit log has been emptied, or drained.</td>
</tr>
<tr>
<td>DECOMMISSION</td>
<td>5</td>
<td>A leaving node has streamed its data to another node.</td>
</tr>
<tr>
<td>MOVE</td>
<td>6</td>
<td>Like NODE_MOVE; a new token range has been assigned.</td>
</tr>
<tr>
<td>NODE_DOWN</td>
<td>13</td>
<td>A node has stopped responding.</td>
</tr>
<tr>
<td>NODE_UP</td>
<td>14</td>
<td>An unresponsive node has recovered.</td>
</tr>
</tbody>
</table>
## Configuration

<table>
<thead>
<tr>
<th>Event</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODE_LEFT</td>
<td>15</td>
<td>A node has left, or been removed from, the ring.</td>
</tr>
<tr>
<td>NODE_JOIN</td>
<td>16</td>
<td>A node has joined the ring.</td>
</tr>
<tr>
<td>NODE_MOVE</td>
<td>17</td>
<td>A node has been assigned a new token range (the token has moved).</td>
</tr>
<tr>
<td>OPSC_UP</td>
<td>18</td>
<td>OpsCenter has been started and is operating.</td>
</tr>
<tr>
<td>OPSC_DOWN</td>
<td>19</td>
<td>OpsCenter was stopped or stopped running.</td>
</tr>
<tr>
<td>GC</td>
<td>20</td>
<td>Java garbage collection has been initiated.</td>
</tr>
</tbody>
</table>

### Alerts

Optionally, you can configure OpsCenter to send alerts for selected levels of events. These alerts can be provided remotely by email, or through HTTP to a selected URL. Alerts are disabled by default.

Alerts are triggered only by events from the OpsCenter API/UI. For example, a nodetool move operation submitted from the command line does not trigger an alert. However, a move operation launched using Dashboard > List View > Actions > Move controls in the OpsCenter does trigger an alert.

All alerts contain the following information about each event captured:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>api_source_ip</td>
<td>IP that originally sent the request.</td>
<td>67.169.50.240</td>
</tr>
<tr>
<td>target_node</td>
<td>Destination of a STREAMING action.</td>
<td>10.1.1.11</td>
</tr>
<tr>
<td>event_source</td>
<td>Component that caused the event.</td>
<td>OpsCenter (i.e., restart, start)</td>
</tr>
<tr>
<td>user</td>
<td>OpsCenter user that caused the event.</td>
<td>opscener_user</td>
</tr>
<tr>
<td>time</td>
<td>Normal timestamp for the event.</td>
<td>1311025650414527</td>
</tr>
<tr>
<td>action</td>
<td>Type of event (see above table)</td>
<td>20</td>
</tr>
<tr>
<td>message</td>
<td>Description of the event.</td>
<td>Garbage Collecting node 10.1.1.13</td>
</tr>
<tr>
<td>level</td>
<td>Numerical code for the log level.</td>
<td>1</td>
</tr>
<tr>
<td>source_node</td>
<td>Node where the event originated.</td>
<td>10.1.1.13</td>
</tr>
<tr>
<td>level_str</td>
<td>Logging level of the event.</td>
<td>INFO</td>
</tr>
</tbody>
</table>

### Enabling email alerts

OpsCenter can post alerts to selected email addresses.

To enable email alerts, you must edit the `<install_locations>/email.conf` file and provide valid SMTP server host and port information. This file is located in the following directories:

1. Make sure that you have valid SMTP mail accounts to send and receive alerts.
2. On the OpsCenter daemon host, open the email.conf file for editing.
3. Set enabled to 1.
4. Provide valid values for your SMTP host, port, user, and password.
5. Enable Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol on your system if you want secure communications. Typically, SSL is required.
6. Provide valid values for the to_addr and from_addr email addresses. The to_addr value is the account that will receive alerts.
7. Optionally, set the level of alerts to send and the desired subject line.
8. Save `<install_locations>/email.conf` and restart the OpsCenter daemon.

To send alerts to multiple email addresses, create a different email conf file with settings for each email address. All conf files are loaded so you can name them `email1.conf`, `email2.conf`, and so on.

In a system with email alerts enabled for critical and alert-level events, `email.conf` looks like:

```plaintext
[email ] enabled =1
# levels can be comma delimited list of any of the following: #
DEBUG,INFO,WARN,ERROR,CRITICAL,ALERT # If left empty, will
listen for all levels levels =CRITICAL,ALERT
smtp_host =smtp.gmail.com
smtp_port =465
smtp_user =mercury@gmail.com
smtp_pass =*********
smtp_use_ssl =1
smtp_use_tls =0
to_addr =cassandra_admin@acme.com
from_addr =mercury@gmail.com
subject =OpsCenter Event
```

**Enabling alerts posted to a URL**

OpsCenter can post alerts to a URL if you provide a correctly formatted POST script. For example, a simple PHP script containing `print_r($_POST);` should be sufficient for getting started.

To enable URL posting on the OpsCenter side:

1. Edit the `posturl.conf` file and provide a path to your script.
   - **Package installations:** `/etc/opscenter/event-plugins`
   - **Binary tarball installations (Linux and Mac OSX):** `<install_location>/opscenter/conf/event-plugins`
   - **Windows installations:** `Program Files (x86)\DataStax Community\opscenter\conf\event-plugins`
2. Make sure your web server and posting script are configured to receive alerts.
3. On the OpsCenter daemon host, open `posturl.conf` for editing.
4. Set enabled to 1. For url, provide a valid path to your posting script.
   ```plaintext
   url=http://50.1.1.11/postOPSCevents.php
   ```
5. Optionally, select the desired logging level. The default is to listen for all levels of events.
6. Save posturl.conf and restart the OpsCenter daemon.

In a system with posting enabled for critical and alert-level events, `posturl.conf` looks like:

```plaintext
[posturl]
enabled = 1
url =http://10.1.1.11/postOPSCevents.php
# levels can be comma delimited list of any of the following:
# DEBUG,INFO,WARN,ERROR,CRITICAL,ALERT
# If left empty, will listen for all levels
levels =CRITICAL,ALERT
```
Verifying that events are posting correctly

You can set preferences to specify how the posting is handled on the receiving side.

1. Post events to a file such as /tmp/events on the web server host.
2. Create a script.
   URL: http://10.1.1.11/postOPSCevents.php

<?php
   file_put_contents( '/tmp/events', print_r ( $_POST,true ),
   FILE_APPEND );
?>

3. Deploy the script. You might need to restart the web server.
4. Launch a logged event, such as an OpsCenter restart or garbage compaction from Dashboard > Cluster > List View.
Output to /tmp looks something like this:

Array
( [api_source_ip ] => 67.169.50.240
 [target_node ] => None
 [event_source ] => OpsCenter
 [user ] => None
 [ time ] => 1311025598851602
 [action ] => 20
 [message ] => Garbage Collecting node 50.1.1.24
 [level ] => 1
 [source_node ] => 50.1.1.24
 [level_str ] => INFO )

Configuring data collection and expiration

OpsCenter collects system and column family metrics data for each node in your cluster.

OpsCenter creates its own keyspace within a cluster for storing collected metrics. This data can also be stored on a cluster other than the one currently being managed by OpsCenter. Metrics data is collected at regular intervals and stored within your cluster in a keyspace called OpsCenter. The column families containing metric data continue to grow. You can configure how long you want to keep historical metrics. Data expires after configurable time periods.

Estimating the amount of data generated

The following table provides guidance for estimating the amount of metrics data generated:

<table>
<thead>
<tr>
<th>Number of days</th>
<th>Number of column families monitored</th>
<th>MB per node</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>5</td>
<td>200</td>
</tr>
<tr>
<td>31</td>
<td>10</td>
<td>300</td>
</tr>
<tr>
<td>31</td>
<td>20</td>
<td>500</td>
</tr>
<tr>
<td>365</td>
<td>5</td>
<td>250</td>
</tr>
<tr>
<td>365</td>
<td>10</td>
<td>380</td>
</tr>
<tr>
<td>365</td>
<td>20</td>
<td>630</td>
</tr>
</tbody>
</table>
The default upper limit of data collected is 365 days.

**Controlling data collection**

To help control consumption of disk space, OpsCenter provides two ways to limit the growth of OpsCenter performance data by:

- Excluding specified keyspaces and column families from performance data collection
- Shortening the time period after which performance data automatically expires

**Excluding keyspaces and column families**

By default, OpsCenter does not collect performance data for its own keyspace or the Cassandra system keyspace. You can manually add any other keyspaces or column families that you do not want to monitor in the [cassandra_metrics] section of the configuration file.

For example, to prevent data collection for the keyspace test as well as the column family Keyspace1.Standard1, uncomment and edit the following values in the OpsCenter cluster configuration file (`<cluster_specific>.conf`):

```plaintext
[cassandra_metrics]
ignored_keyspaces = system, OpsCenter, test
ignored_column_families = Keyspace1.Standard1
```

Column families are specified in the format:

```
<keyspace_name>.<column_family_name>.
```

**Changing performance data expiration times**

Performance data stored in OpsCenter expires after configurable time periods. The default values are designed to provide efficient compaction and eventual deletion of the data, with faster expiration times for the more granular, larger-volume data rollups.

- One-minute rollups (1min_ttl) expire after one week, or 604800 seconds.
- Five-minute rollups (5min_ttl) expire after four weeks, or 2419200 seconds.
- Two-hour rollups (2hr_ttl) expire after one year, or 31536000 seconds.

To change expiration time period:

In this example, the one-minute and five-minute rollups are set to expire twice as fast as the defaults, and two-hour rollups are set to be kept indefinitely (expiration is disabled).

1. Edit the `conf/clusters/<cluster>.conf` file.
2. Add the following time-to-live (ttl) values under a [cassandra_metrics] section:

   ```plaintext
   [cassandra_metrics]
   1min_ttl = 302400
   5min_ttl = 1209600
   2hr_ttl = -1
   ```

3. Restart OpsCenter.
   
   Data collected after restarting OpsCenter expires according to the new setting. The data collected before restarting OpsCenter expires according to the setting in effect when it was collected.

**Storing collection data on a different cluster**

If you do not want OpsCenter to store data in an OpsCenter keyspace on the cluster being managed, you can store the data on a separate cluster.
Add a section to the cluster configuration file.

- `/etc/opscenter/clusters/<MyCluster>.conf`
- `<install_location>/conf/clusters/<MyCluster>.conf`

```
[storage_cassandra]
seed_hosts = host1, host2
api_port = 9160
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seed_hosts</td>
<td>A comma-delimited list of at least one of the nodes in the cluster where you want to store collection data.</td>
</tr>
<tr>
<td>api_port</td>
<td>The Thrift port used by that cluster.</td>
</tr>
<tr>
<td>username</td>
<td>The appropriate username (if using Thrift security).</td>
</tr>
<tr>
<td>password</td>
<td>The appropriate password (if using Thrift security).</td>
</tr>
<tr>
<td>keyspace</td>
<td>The name of the keyspace to store the collection data in. (OpsCenter by default.)</td>
</tr>
</tbody>
</table>

**Advanced configuration**

To configure advanced capabilities, you can manually modify the configuration files.

**Note:** The OpsCenter console is the most convenient way to configure basic OpsCenter settings.

- `opscenterd.conf`: configures the properties for the OpsCenter daemon.
- `<cluster_specific>.conf`: configures properties for each cluster monitored by OpsCenter. This file is created when you add a cluster to the Opscenter.

**OpsCenter configuration properties**

These properties are configured in the `opscenterd.conf` file. The location of the `opscenterd.conf` file depends on the type of installation:

- Packaged installs: `/etc/opscenter/opscenterd.conf`
- Binary installs: `<install_location>/conf/opscenterd.conf`
- Windows installs: `Program Files (x86)\DataStax Community\opscenter\conf\opscenterd.conf`

**Note:** After changing properties in this file, you must restart OpsCenter for the changes to take effect.

**[agents] ssh_port**

The Secure Shell (SSH) port that listens for agent-OpsCenter communications. Add an [agents] section, if one doesn’t already exist, to the `opscenterd.conf`. In this section, add the `ssh_port` option and a value for the port number:

```
[agents]
ssh_port = 2222
```

**[webserver] port**

The HTTP port used for client connections to the OpsCenter web server. Default is 8888.
Optional HTTPS support. To enable, remove the comment markers (#) in front of properties prefixed with ssl in the opscenterd.conf file, as described in Configuring HTTPS.

[webserver] interface
The interface that the web server uses to listen for client connections. The interface must be an externally accessible IP address or host name.

[logging] level
The logging level for OpsCenter. Available levels are (from most to least verbose): TRACE, DEBUG, INFO, WARN, or ERROR.

The OpsCenter log file is located in /var/log/opscenter/opscenterd.log.

[stat_reporter] interval
Reporting to DataStax Support. By default, OpsCenter periodically sends usage metrics about the cluster to DataStax Support.

To disable the phone-home functionality, add the following lines to your opscenterd.conf file:

```
[stat_reporter ]
interval = 0
```

Additional configuration metric collection properties are available in Metrics Collection Properties on page 33.

[authentication] passwd_file
Full path to the file for configuring password authentication for OpsCenter. If this file does not exist, OpsCenter does not verify passwords.

To enable password authentication, use the set_passwd.py utility to create users and set their password and role. OpsCenter currently has two available roles: admin or user.

Statistics reporter properties
A complete breakdown of the data OpsCenter communicates back to DataStax. The data is sent in a key-value JSON format.

The following information is recorded about the OpsCenter install:

- install_id—This is a random uuid generated when OpsCenter starts for the first time. This is used for associating reports from the same install.
- is_paid—This is a flag indicating whether or not this is the free or enterprise version of OpsCenter.
- opscenter_version—The version of OpsCenter in use.
- opscenter_ram—The amount of RAM on the OpsCenter machine.
- opscenter_cores—The number of cores on the OpsCenter machine.
- opscenter_os—The generic name of the operating system of the OpsCenter machine. For example, linux/windows/mac.
- opscenter_os_sub—The specific name of the operating system of the OpsCenter machine. For example CentOS/Ubuntu/Debian.
- opscenter_os_version—The operating system version of the OpsCenter machine.
- opscenter_arch—The architecture of the OpsCenter machine.
- python_version—The version of python running on the OpsCenter machine.
- opscenter_instance_type—The instance type the OpsCenter machine, if OpsCenter is running in EC2.
- separate_storage—A flag indicating if OpsCenter is storing metrics in the cluster it is monitoring.
- config_diff—A list of the OpsCenter config options that were modified to be different than the defaults. This includes the names of the options that were changed but not the values of those options.

These statistics are collected about each cluster OpsCenter is monitoring:

- cluster_id—An md5 hash of the cluster name. Used for identifying unique clusters while maintaining anonymity.
Configuration

- **conf_id**—An md5 hash of the file name the config for the cluster is stored in. Used for the same purposes as cluster_id.
- **partitioner**—The partitioner the cluster is using.
- **snitch**—The snitch the cluster is using.
- **keyspace_count**—The number of keyspaces in the cluster.
- **columnfamily_count**—The number of column families in the cluster.
- **strategy_options**—A list of the replication options used for each keyspace in the cluster.
- **cql3_cf_count**—The number of column families created with CQL3 in the cluster.
- **node_count**—The number of nodes in the cluster.
- **avg_token_count**—The average number of tokens per node.
- **cassandra_versions**—A list of the different Cassandra versions in the cluster.
- **bdp_version**—A list of the different DataStax Enterprise versions in the cluster.
- **rack_map**—A map of each rack in the cluster and how many nodes are in that rack.
- **dc_count**—The number of datacenters in the cluster.
- **free_space**—The amount of free disk space across the cluster.
- **used_space**—The amount of used disk space across the cluster.
- **cluster_os**—A list of the different operating systems used across the cluster.
- **cluster_ram**—The average amount of ram per node in the cluster.
- **cluster_cores**—The average number of cores per node in the cluster.
- **cluster_instance_types**—A list of the EC2 instance types in the cluster, if EC2 is being used.

Cluster configuration properties

These properties inform OpsCenter about the Real-time (Cassandra), Analytics (Hadoop), and Search (Solr) nodes that it is monitoring.

Cassandra connection properties

These properties are configured in the cluster-specific `opscenterd.conf` file.

The location of the `opscenterd.conf` file depends on the type of installation:

- **Packaged installs:** `/etc/opscenter/clusters/<cluster_specific>.conf`
- **Binary installs:** `<install_location>/conf/clusters/<cluster_specific>.conf`
- **Windows installs:** `Program Files (x86)\DataStax Community\opscenter\conf\clusters\<cluster_specific>.conf`

**Note:** After changing properties in this file, restart OpsCenter for the changes to take effect.

[jmx] port

The JMX (Java Management Extensions) port of your cluster. In Cassandra versions 0.8 and higher, the JMX port is 7199.

[cassandra] seed_hosts

A Cassandra seed node is used to determine the ring topology and obtain gossip information about the nodes in the cluster. This should be the same comma-delimited list of seed nodes as the one configured for your Cassandra or DataStax Enterprise cluster by the seeds property in the `cassandra.yaml` configuration file.

[cassandra] api_port

The Thrift remote procedure call port configured for your cluster. Same as the rpc_port property in the `cassandra.yaml` configuration file. Default is 9160.

[cassandra] install_location

The directory in which Cassandra is installed. If install_location is not specified, OpsCenter looks in the package-specific installation locations. For a tarball installation of DataStax Enterprise, the install_location is `<dse_install_location>/resources/cassandra`. 

[cassandra] conf_location
The location of the `cassandra.yaml` configuration file. If `install_location` is specified, but `conf_location` is not, then `conf_location` is assumed to be `<install_location>/conf/cassandra.yaml`. If `conf_location` is specified, it must be the absolute path to the Cassandra configuration file on all nodes. These settings are cluster-wide and require that the specified locations be correct for every node.

**Metrics Collection Properties**
These properties are used to limit the keyspaces and column families for which you collect metrics.

```
[cassandra_metrics] ignored_keyspaces
A comma-delimited list of Cassandra keyspaces for which you do not want to collect performance metrics. By default, the system and OpsCenter keyspaces are excluded.
```

```
[cassandra_metrics] ignored_column_families
A comma-delimited list of Cassandra column families for which you do not want to collect performance metrics. Entries should be in the form of keyspace_name.columnfamily_name.
```

**Performance Data Expiration Properties**
These properties set the expiration time for data stored in the OpsCenter keyspace. Each time period for rolling up data points into summary views has a separate expiration threshold, or time-to-live (ttl) value expressed in seconds. By default, shorter time periods have lower values that result in more efficient expiration and compaction of the relatively larger volumes of data. Uncomment these properties to change the default expiration periods for performance data. Properties and default values are:

```
1min_ttl = 604800
One-minute rollups expire after after one week, or 604800 seconds.
```

```
5min_ttl = 2419200
Five-minute rollups expire after four weeks, or 2419200 seconds.
```

```
2hr_ttl = 31536000
Two-hour rollups expire after one year, or 31536000 seconds.
```
Using OpsCenter

OpsCenter is a Web application for monitoring and administering all nodes in a Cassandra cluster from one centralized console. It runs on the client-side in a web browser.

Overview

The major areas of functionality.

At the top of every functional area of OpsCenter, you can access these functions:

- Create Cluster: create a cluster.
- Add a cluster: add clusters to OpsCenter to monitor. Available in OpsCenter Enterprise Edition only.
- Feedback: an online form that sends your evaluation of OpsCenter or any comments to us.
- Report: information about clusters that OpsCenter manages in PDF format.

OpsCenter is divided into these main functional areas:

- **Overview**—Survey each cluster’s Dashboard in this condensed view. Displayed when multiple clusters are present.
- **Dashboard**—View graphs of the most commonly watched Cassandra performance metrics.
- **Software update notification**—Be notified when an upgrade to OpsCenter or Cassandra or DSE on any of your clusters is available. The actual upgrade processes will be manual. When a new version is available a link at the top of the Dashboard appears. Clicking on it displays a dialog with a link to the new version of the software.
- **Cluster**—See your cluster from different perspectives and perform certain maintenance operations on cluster nodes.
- **Cluster administration**—Add, modify, or remove a cluster from OpsCenter. Available in OpsCenter Enterprise Edition only.
- **Performance**—Monitor a number of Cassandra cluster performance metrics. Real-time and historical performance metrics are available at different granularities: cluster-wide, per node, or per column family.
- **Alerts**—Configure alert thresholds for a number of Cassandra cluster-wide, column family, and operating system metrics. Available in OpsCenter Enterprise Edition only.
- **Schema**—Create and manage keyspaces and the column families within them.
- **Data backups**—Visually take, schedule, and manage backups across all registered clusters. Restore to clusters from backups. Available in OpsCenter Enterprise Edition only.
- **Data Explorer**—Browse through column family data.
- **Event Log**—View the most recent OpsCenter log events, such as OpsCenter startup and shutdown.

Node administration

In the Cluster area of OpsCenter, you select different views of the nodes comprising your Cassandra cluster, and then, perform node management.

Node management operations

OpsCenter Enterprise Edition 3.2 can monitor and administer Cassandra 1.0 and later, but with Cassandra 1.2 it will not work as described in these procedures if you enable virtual nodes. When you enable virtual nodes, OpsCenter chooses a single token for each node for operations, such as collecting metrics. Attempting to move nodes, rebalance nodes, and perform other tasks involving token ranges is not supported.
Each of the cluster views (Ring, Physical, and List) has an **Actions** button.

- In the Ring View or Physical View of a cluster, click the graphic representation of the node. The Actions button appears in the dialog.
- In the List View of a cluster, select a node. If necessary, expand the List View window to see the Actions button on the right side.

Click the Actions button to access a drop-down list of node management operations. To avoid impacting cluster performance, perform actions that move data between nodes at low-usage times. Examples of such actions are move, decommission, and repair.

This table describes actions on the drop-down list:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>View metrics</td>
<td>Redirects you to the Performance area of OpsCenter where you can select metrics graphs and configure performance views for the selected node.</td>
</tr>
<tr>
<td>View replication</td>
<td>Shows the replication relationships between the selected node and other nodes in the cluster. Visible in Ring View and Physical View.</td>
</tr>
<tr>
<td>Configure</td>
<td>Allows you to change the configuration of the node.</td>
</tr>
<tr>
<td>Start</td>
<td>Starts the DSE / Cassandra process on the node.</td>
</tr>
<tr>
<td>Stop</td>
<td>Stops the DSE / Cassandra process on the node.</td>
</tr>
<tr>
<td>Restart</td>
<td>Restarts the DSE / Cassandra process on the node.</td>
</tr>
<tr>
<td>Cleanup</td>
<td>Removes rows that the node is no longer responsible for. This is usually done after changing the partitioner tokens or the replication options for a cluster.</td>
</tr>
<tr>
<td>Compact</td>
<td>Performs a major compaction, which is not a recommended procedure in most Cassandra clusters.</td>
</tr>
<tr>
<td>Flush</td>
<td>Causes the recent writes currently stored in memory (memtables) to be flushed to disk as persistent SSTables.</td>
</tr>
<tr>
<td>Repair</td>
<td>Makes a node consistent with its replicas by doing an in-memory comparison of all the rows of a column family, and resolving any discrepancies between replicas by updating outdated rows with the current data.</td>
</tr>
<tr>
<td>Perform GC</td>
<td>Forces the Java Virtual Machine (JVM) on the selected node to perform a garbage collection (GC).</td>
</tr>
<tr>
<td>Decommission</td>
<td>Removes a node from the cluster and streams its data to neighboring replicas.</td>
</tr>
<tr>
<td>Drain</td>
<td>Causes the recent writes currently stored in memory (memtables) to be flushed to disk as persistent SSTables, and then makes the node read-only (the node will stop accepting new writes). This is usually done when upgrading a node.</td>
</tr>
<tr>
<td>Move</td>
<td>Changes the partitioner token assignment for the node, thus changing the range of data that the node is responsible for.</td>
</tr>
</tbody>
</table>

**Cluster administration**

You can administer clusters from OpsCenter.

OpsCenter Enterprise Edition 3.2 can monitor and administer Cassandra 1.2 as described in these procedures unless you enable virtual nodes. When you enable virtual nodes, OpsCenter chooses a single
Using OpsCenter

token for each node for operations, such as collecting metrics. Attempting to move nodes, rebalance nodes, and perform other tasks involving token ranges is not supported.

**OpsCenter Cassandra / DSE Support**

These represent the versions that we officially support and test.

<table>
<thead>
<tr>
<th>OpsCenter version</th>
<th>Cassandra versions</th>
<th>DataStax Enterprise versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4.x</td>
<td>0.7, 0.8, 1.0, 1.1</td>
<td>1.0, 2.0</td>
</tr>
<tr>
<td>2.0</td>
<td>0.8, 1.0, 1.1</td>
<td>1.0, 2.0</td>
</tr>
<tr>
<td>2.1</td>
<td>0.8, 1.0, 1.1</td>
<td>1.0, 2.0</td>
</tr>
<tr>
<td>3.0</td>
<td>1.0, 1.1, 1.2</td>
<td>1.0, 2.x, 3.0</td>
</tr>
<tr>
<td>3.1</td>
<td>1.0, 1.1, 1.2</td>
<td>1.0, 2.x, 3.0.x</td>
</tr>
<tr>
<td>3.2</td>
<td>1.1, 1.2</td>
<td>2.x, 3.0.x, 3.1.x</td>
</tr>
</tbody>
</table>

**Creating a cluster**

To add a cluster to Opscenter.

**Note:** When deploying to EC2, the OpsCenter daemon-to-agent communication must be open to 0.0.0.0 in the EC2 security group (that is for ports 61620 and 61621).

1. Click Create cluster.
2. Fill out the form as appropriate.
3. Click Save Cluster.

**Adding a cluster**

To add a cluster to Opscenter.

1. Click Add a cluster.
2. Enter the hostname or IP address for the nodes comprising the cluster.

   ec2-123-45-6-789.us-west-1.compute.amazonaws.com
   ec2-234-56-7-890.us-west-1.compute.amazonaws.com

3. If you are not using the default JMX or Thrift ports, enter the appropriate port numbers.
4. If required, click Add Credentials and enter the username and password for JMX or Thrift ports.
5. (Optional) You can check the DSE security (kerberos) is enabled on my cluster and enter the service name.
6. (Optional) You can check the Client node encryption is enabled on the cluster and enter your PEM encoded certificate in CA Certificate File Path.
   If you have a CER encoded certificate, use the following command to convert it.
   ```
   $ openssl x509 -inform der -in certificate.cer -out certificate.pem
   ```
7. (Optional) You can check the Validate SSL Certificates and enter the Truststore File Path and Trustore Password.
   For more information about enabling Kerberos see Security in the DSE Documentation.
8. Click Add Cluster.
Generating a report

To generate a PDF report about the cluster being monitored, click Report at the top of the OpsCenter interface. The report shows the version of OpsCenter, number of clusters and nodes being monitoring, gigabytes of storage used, name of the cluster, and information about nodes in the cluster. The node information includes:

- Node name and IP address
- Cassandra software version
- DataStax software version
- Memory usage
- Operating system running on the node

You can save or print the PDF report.

Collecting diagnostic data

Download a compressed tarball that contains information about the OpsCenter daemon and all the nodes in a specified cluster.

You can attach this diagnostic data to support tickets.

1. Click Diagnostics to download the tarball.
   You see the message: “Collecting cluster data; please wait, this may take a few minutes...”
2. Save the tarball to your local machine.
   diagnostics.tar.gz
   Depending on your browser settings, you may or may not be prompted for a file directory to save the tarball in.

Adding a node to a cluster

To add a node to a cluster.

1. Click Add Node in a cluster view.
2. Enter the following:

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>The version of DSE to install on the node.</td>
</tr>
<tr>
<td>DataStax credentials</td>
<td>The username and password you received when registering to Download DSE.</td>
</tr>
<tr>
<td>Nodes</td>
<td>The hostname or IP address, token, and software to install on the node (from Cassandra, Solr, and Hadoop). You can add more than one node by clicking Add.</td>
</tr>
<tr>
<td>Node credentials (sudo)</td>
<td>The username and password to authenticate on the host. (Optional) the private SSH key to use to use for authentication.</td>
</tr>
</tbody>
</table>

Configuring a cluster

You can manage cassandra.yaml from OpsCenter.

If the cluster exists in multiple datacenters, you can configure cassandra.yaml for a single datacenter or for all nodes in a cluster. To manage cassandra.yaml for a single node by clicking on the Actions dropdown for a node.

1. Click Configure Cluster in any of the Cluster views.
2. Edit the value for any of the options.
Using OpsCenter

For a description of the options in the file, see the documentation for the version of Cassandra or DSE which the cluster or node is running. For example, cassandra.yaml.

3. When finished editing, click Save.
4. You will be prompted to perform a rolling restart on the nodes in your cluster, or you may click Cancel if you do not wish to do a restart.

Removing a cluster

To remove a cluster.

This removes the cluster from OpsCenter; it does not delete the cluster.

1. Click Edit Cluster.
2. Click Delete Cluster.

When you delete a cluster, any EC2 nodes are not deleted.

Rebalancing a cluster

Cluster rebalancing is a process that makes sure each node in a Cassandra cluster is managing an equal amount of data.

Currently, OpsCenter only supports rebalancing on clusters using the random partitioner or murmur 3 partitioner. Ordered partitioners are not supported. When using the random partitioner or murmur 3 partitioner, a rebalance is usually required only when you have changed the cluster topology in some way, such as adding or removing nodes or changing the replica placement strategy.

A cluster is considered balanced when each node is responsible for an equal range of data. This is done by evaluating the partitioner tokens assigned to each node to make sure that the data ranges each node is responsible for are even. Even though a cluster is considered balanced, it is still possible that one or more nodes have more data than the others. This is because the size of the rows is not taken into account, only the number of rows managed by each node.

1. In the Cluster section of OpsCenter, select Ring View, Physical View or List View.

   The dashboard displays the specified view of your cluster.

2. Click the Rebalance Cluster button.

   OpsCenter checks if the token ranges are evenly distributed across the nodes in your cluster.

3. If your cluster is already balanced, then there is nothing for OpsCenter to do. If the cluster does require rebalancing, OpsCenter performs the following steps:
   • Calculates appropriate token ranges for each node and identify the nodes that need to move.
   • Makes sure that there is the appropriate free space to perform the rebalancing.
   • Moves nodes, one node at a time so as to lessen the impact on the cluster workloads. A move operation involves changing the partitioner token assignment for the node, thus changing the range of data that the node is responsible for. A move will stream data from other nodes.
   • After a move is complete on a node, runs cleanup. A cleanup operation removes rows that the node is no longer responsible for.

4. If you cancel a rebalance operation before all nodes are moved, you can resume it at a later time by clicking the Rebalance Cluster button again.

Restarting a cluster

To restart a cluster.

You can start, stop, or restart the Cassandra or DSE service on any node. This can be done via the Actions dropdown on a node.

There is also rolling restart functionality for the entire cluster. This can be done via the Restart Cluster in any of the Cluster views.

1. In any cluster view, click on a node.
2. In the contextual menu select **Restart** from the **Actions** dropdown.

**Modifying a cluster setting**

To modify a cluster setting:

1. Click **Edit Cluster**.
2. Change the IP addresses of cluster nodes.
3. Change JMX and Thrift listen port numbers.
4. Click **Add** credentials if the ports require authentication.
5. (Optional) You can check the DSE security (kerberos) is enabled on my cluster and enter the service name.
6. (Optional) You can check the Client node encryption is enabled on the cluster and enter your PEM encoded certificate in CA Certificate File Path.
   
   If you have a CER encoded certificate, use the following command to convert it.
   
   ```bash
   $ openssl x509 -inform der -in certificate.cer -out certificate.pem
   ```
7. (Optional) You can check the Validate SSL Certificates and enter the Truststore File Path and Trustore Password.
   
   For more information about enabling Kerberos see **Security** in the DSE Documentation.
8. Click **Save Cluster**.

**Performance metrics**

In the Performance area of OpsCenter, you monitor a number of performance metrics about a Cassandra cluster. Real-time and historical performance metrics are available at different granularities: cluster-wide, per node, or per column family. OpsCenter 3.0 and later has been optimized to handle thousands of column families efficiently.

**Using performance metrics**

Select Performance in the OpsCenter Console to view these types of metrics:

• Cluster Performance Metrics
• Pending Task Metrics
• Column Family Metrics

When you add a graph, you choose the Metric and the source that OpsCenter uses to collect the data for the graph:

• Cluster wide
• All nodes
• The node running Opscenter

Several commonly-used performance metrics graphs are displayed initially. Data appears in the graphs after you set alerts.

You can save, delete, and choose the default view of graphs. Click the link to save presets at the top of the Performance area. The **Save**, **Delete**, and **Make Default** menu options are available after saving more than one view.

**Cluster performance metrics**

Cluster metrics are aggregated across all nodes in the cluster. Cluster metrics are a good way to monitor cluster performance at a high level. OpsCenter tracks a number of cluster-wide metrics for read performance, write performance, memory and capacity.
Watching for variations in cluster performance can signal potential performance issues that may require further investigation. For general performance monitoring, watching for spikes in read and write latency, along with an accumulation of pending operations can signal issues that may require further investigation. Drilling down on high-demand column families can further pinpoint the source of performance issues with your application.

Write requests
The number of write requests per second. Monitoring the number of requests over a given time period can give you an idea of system write workload and usage patterns.

Write request latency
The response time (in milliseconds) for successful write requests. The time period starts when a node receives a client write request, and ends when the node responds back to the client. Optimal or acceptable levels of write latency vary widely according to your hardware, your network, and the nature of your write load. For example, the performance for a write load consisting largely of granular data at low consistency levels would be evaluated differently from a load of large strings written at high consistency levels.

Read requests
The number of read requests per second. Monitoring the number of requests over a given time period can give you an idea of system read workload and usage patterns.

Read request latency
The response time (in milliseconds) for successful read requests. The time period starts when a node receives a client read request, and ends when the node responds back to the client. Optimal or acceptable levels of read latency vary widely according to your hardware, your network, and the nature of your application read patterns. For example, the use of secondary indexes, the size of the data being requested, and the consistency level required by the client can all impact read latency. An increase in read latency can signal I/O contention. Reads can slow down when rows are fragmented across many SSTables and compaction cannot keep up with the write load.

Cassandra JVM memory usage
The average amount of Java heap memory (in megabytes) being used by Cassandra processes. Cassandra opens the JVM with a heap size that is half of available system memory by default, which still allows an optimal amount of memory remaining for the OS disk cache. You may need to increase the amount of heap memory if you have increased column family memtable or cache sizes and are getting out-of-memory errors. If you monitor Cassandra Java processes with an OS tool such as top, you may notice the total amount of memory in use exceeds the maximum amount specified for the Java heap. This is because Java allocates memory for other things besides the heap. It is not unusual for the total memory consumption of the JVM to exceed the maximum value of heap memory.

JVM CMS collection count
The number of concurrent mark-sweep (CMS) garbage collections performed by the JVM per second. These are large, resource-intensive collections. Typically, the collections occur every 5 to 30 seconds.

JVM CMS collection time
The time spent collecting CMS garbage in milliseconds per second (ms/sec).

Note: A ms/sec unit defines the number of milliseconds for garbage collection for each second that passes. For example, the percentage of time spent on garbage collection in one millisecond (.001 sec) is 0.1%.
**JVM ParNew collection count**

The number of parallel new-generation garbage collections performed by the JVM per second. These are small and not resource intensive. Normally, these collections occur several times per second under load.

**JVM ParNew Collection Time**

The time spent performing ParNew garbage collections in ms/sec. The rest of the JVM is paused during ParNew garbage collection. A serious performance hit can result from spending a significant fraction of time on ParNew collections.

**Data size**

The size of column family data (in gigabytes) that has been loaded/inserted into Cassandra, including any storage overhead and system metadata. DataStax recommends that data size not exceed 70 percent of total disk capacity to allow free space for maintenance operations such as compaction and repair.

**Total bytes compacted**

The number of sstable data compacted in bytes per second.

**Total compactions**

The number of compactions (minor or major) performed per second.

**Pending task metrics**

Pending task metrics track requests that have been received by a node, but are waiting to be processed. An accumulation of pending tasks on a node can indicate a potential bottleneck in performance and should be investigated.

Cassandra maintains distinct thread pools for different stages of execution. Each of these thread pools provide granular statistics on the number of pending tasks for that particular process. If you see pending tasks accumulating, it is indicative of a cluster that is not keeping up with the workload. Essentially, pending tasks mean that things are backing up, which is usually caused by a lack of (or failure of) cluster resources such as disk bandwidth, network bandwidth or memory.

**Pending task metrics for writes**

Pending tasks for the following metrics indicate that write requests are arriving faster than they can be handled.

**Flushes pending**

The flush process flushes memtables to disk as SSTables. This metric shows the number of memtables queued for the flush process. The optimal number of pending flushes is 0 (or at most a very small number). A value greater than 0 indicates either I/O contention or degrading disk performance (see disk metrics such as disk latency, disk throughput, and disk utilization for indications of disk health).

**Flush sorter tasks pending**

The flush sorter process performs the first step in the overall process of flushing memtables to disk as SSTables.

**memtable post flushers pending**

The memtable post flush process performs the final step in the overall process of flushing memtables to disk as SSTables.
Using OpsCenter

Write requests pending
The memtable post flush process performs the final step in the overall process of flushing memtables to disk as SSTables.

Replicate on write tasks pending
When an insert or update to a row is written, the affected row is replicated to all other nodes that manage a replica for that row. This is called the ReplicateOnWriteStage. This metric tracks the pending tasks related to this stage of the write process. During low or moderate write load, you should see 0 pending replicate on write tasks (or at most a very low number). A continuous high number signals a need to investigate disk I/O or network contention problems.

Pending task metrics for reads
Pending tasks for the following metrics indicate I/O contention, and can manifest in degrading read performance.

Read requests pending
The number of read requests that have arrived into the cluster but are waiting to be handled. During low or moderate read load, you should see 0 pending read operations (or at most a very low number). A continuous high number of pending reads signals a need for more capacity in your cluster or to investigate disk I/O contention. Pending reads can also indicate an application design that is not accessing data in the most efficient way possible.

Read repair tasks pending
The number of read repair operations that are queued and waiting for system resources in order to run. The optimal number of pending read repairs is 0 (or at most a very small number). A value greater than 0 indicates that read repair operations are in I/O contention with other operations. If this graph shows high values for pending tasks, this may suggest the need to run a node repair to make nodes consistent. Or, for column families where your requirements can tolerate a certain degree of stale data, you can lower the value of the column family parameter read_repair_chance.

Compactions pending
An upper bound of the number of compactions that are queued and waiting for system resources in order to run. This is a worst-case estimate. The compactions pending metric is often misleading. An unrealistic, high reading often occurs. The optimal number of pending compactions is 0 (or at most a very small number). A value greater than 0 indicates that read operations are in I/O contention with compaction operations, which usually manifests itself as declining read performance. This is usually caused by applications that perform frequent small writes in combination with a steady stream of reads. If a node or cluster frequently displays pending compactions, that is an indicator that you may need to increase I/O capacity by adding nodes to the cluster. You can also try to reduce I/O contention by reducing the number of insert/update requests (have your application batch writes for example), or reduce the number of SSTables created by increasing the memtable size and flush frequency on your column families.

Pending task metrics for cluster operations
Pending tasks for the following metrics indicate a backup of cluster operational processes such as those maintaining node consistency, system schemas, fault detection, and inter-node communications. Pending tasks for resource-intensive operations (such as repair, bootstrap or decommission) are normal and expected while that operation is in progress, but should continue decreasing at a steady rate in a healthy cluster.
**Manual repair tasks pending**

The number of operations still to be completed when you run anti-entropy repair on a node. It will only show values greater than 0 when a repair is in progress. Repair is a resource-intensive operation that is executed in stages: comparing data between replicas, sending changed rows to the replicas that need to be made consistent, deleting expired tombstones, and rebuilding row indexes and bloom filters. Tracking the state of this metric can help you determine the progress of a repair operation. It is not unusual to see a large number of pending tasks when a repair is running, but you should see the number of tasks progressively decreasing.

**Gossip tasks pending**

Cassandra uses a protocol called gossip to discover location and state information about the other nodes participating in a Cassandra cluster. In Cassandra, the gossip process runs once per second on each node and exchanges state messages with up to three other nodes in the cluster. Gossip tasks pending shows the number of gossip messages and acknowledgments queued and waiting to be sent or received. The optimal number of pending gossip tasks is 0 (or at most a very small number). A value greater than 0 indicates possible network problems (see network traffic for indications of network health).

**Hinted handoff pending**

While a node is offline, other nodes in the cluster will save hints about rows that were updated during the time the node was unavailable. When a node comes back online, its corresponding replicas will begin streaming the missed writes to the node to catch it up. The hinted handoff pending metric tracks the number of hints that are queued and waiting to be delivered once a failed node is back online again. High numbers of pending hints are commonly seen when a node is brought back online after some down time. Viewing this metric can help you determine when the recovering node has been made consistent again. Hinted handoff is an optional feature of Cassandra. Hints are saved for a configurable period of time (an hour by default) before they are dropped. This prevents a large accumulation of hints caused by extended node outages.

**Internal responses pending**

The number of pending tasks from various internal tasks such as nodes joining and leaving the cluster.

**Migrations pending**

The number of pending tasks from system methods that have modified the schema. Schema updates have to be propagated to all nodes, so pending tasks for this metric can manifest in schema disagreement errors.

**Miscellaneous tasks pending**

The number of pending tasks from other miscellaneous operations that are not ran frequently.

**Request response pending**

The progress of rows of data being streamed from the receiving node. Streaming of data between nodes happens during operations such as bootstrap and decommission when one node sends large numbers of rows to another node.

**Streams pending**

The progress of rows of data being streamed from the sending node. Streaming of data between nodes happens during operations such as bootstrap and decommission when one node sends large numbers of rows to another node.
Column family performance metrics
Column family metrics allow you to drill down and locate specific areas of your application workloads that
are the source of performance issues. If you notice a performance trend at the OS or cluster level, viewing
column family metrics can provide a more granular level of detail.

The metrics for KeyCache Hits, RowCache Hits and SSTable Size can only be viewed on a single column
family at a time. Otherwise, all column family metrics are available for specific column families as well as
for all column families on a node.

In addition to monitoring read latency, write latency and load on a column family, you should also monitor
the hit rates on the key and row caches for column families that rely on caching for performance. The
more requests that are served from the cache, the better response times will be.

OpsCenter 3.2 and later has been optimized to handle thousands of column families efficiently. If a
column family experiences a dramatic dip in performance, check the Pending Tasks metric for a back-up
in queued operations.

Viewing SSTable Size and SSTable Count for a specific column family (or counts for all families) can help
with compaction tuning.

Column family local writes
The write load on a column family measured in requests per second. This metric includes all writes to a
given column family, including write requests forwarded from other nodes. This metric can be useful for
tracking usage patterns of your application.

Column family local write latency
The response time in milliseconds for successful write requests on a column family. The time period
starts when nodes receive a write request, and ends when nodes respond. Optimal or acceptable levels
of write latency vary widely according to your hardware, your network, and the nature of your write load.
For example, the performance for a write load consisting largely of granular data at low consistency levels
would be evaluated differently from a load of large strings written at high consistency levels.

Column family local reads
The read load on a column family measured in requests per second. This metric includes all reads to a
given column family, including read requests forwarded from other nodes. This metric can be useful for
tracking usage patterns of your application.

Column family local read latency
The response time in milliseconds for successful reads on a column family. The time period starts
when a node receives a read request, and ends when the node responds. Optimal or acceptable levels
of read latency vary widely according to your hardware, your network, and the nature of your application
read patterns. For example, the use of secondary indexes, the size of the data being requested, and the
consistency level required by the client can all impact read latency. An increase in read latency can signal
I/O contention. Reads can slow down when rows are fragmented across many SSTables and compaction
cannot keep up with the write load.

Column family key cache requests
The total number of read requests on the row key cache.

Column family key cache hits
The number of read requests that resulted in the requested row key being found in the key cache.
Column family key cache hit rate
The percentage of cache requests that resulted in a cache hit that indicates the effectiveness of the key cache for a given column family. The key cache is used to find the exact location of a row on disk. If a row is not in the key cache, a read operation will populate the key cache after accessing the row on disk so subsequent reads of the row can benefit. Each hit on a key cache can save one disk seek per SSTable. If the hits line tracks close to the requests line, the column family is benefiting from caching. If the hits fall far below the request rate, this suggests that you could take actions to improve the performance benefit provided by the key cache, such as adjusting the number of keys cached.

Column family row cache requests
The total number of read requests on the row cache. This metric is only meaningful for column families with row caching configured (it is not enabled by default).

Column family row cache hits
The number of read requests that resulted in the read being satisfied from the row cache. This metric is only meaningful for column families with row caching configured (it is not enabled by default).

Column family row cache hit rate
The percentage of cache requests that resulted in a cache hit that indicates the effectiveness of the row cache for a given column family. This metric is only meaningful for column families with row caching configured (it is not enabled by default). The graph tracks the number of read requests in relationship to the number of row cache hits. If the hits line tracks close to the requests line, the column family is benefiting from caching. If the hits fall far below the request rate, this suggests that you could take actions to improve the performance benefit provided by the row cache, such as adjusting the number of rows cached or modifying your data model to isolate high-demand rows.

Column family SSTable size
The current size of the SSTables for a column family. It is expected that SSTable size will grow over time with your write load, as compaction processes continue doubling the size of SSTables. Using this metric together with SSTable count, you can monitor the current state of compaction for a given column family. Viewing these patterns can be helpful if you are considering reconfiguring compaction settings to mitigate I/O contention.

Column family SSTable count
The current number of SSTables for a column family. When column family memtables are persisted to disk as SSTables, this metric increases to the configured maximum before the compaction cycle is repeated. Using this metric together with SSTable size, you can monitor the current state of compaction for a given column family. Viewing these patterns can be helpful if you are considering reconfiguring compaction settings to mitigate I/O contention.

Column family pending reads and writes
The number of pending reads and writes on a column family. Pending operations are an indication that Cassandra is not keeping up with the workload. A value of zero indicates healthy throughput. If out-of-memory events become an issue in your Cassandra cluster, it may help to check cluster-wide pending tasks for operations that may be clogging throughput.

Bloom filters are used to avoid going to disk to try to read rows that don't actual exist.

Column family bloom filter space used
The size of the bloom filter files on disk. This grows based on the number of rows in a column family and is tunable through the per-CF attribute, bloom_filter_fp_chance; increasing the value of this attribute
shrinks the bloom filters at the expense of a higher number of false positives. Cassandra reads the bloom filter files and stores them on the heap, so large bloom filters can be expensive in terms of memory consumption.

**Note:** Bloom filters are used to avoid going to disk to try to read rows that don’t actually exist.

**Column family bloom filter false positives**

The number of false positives, which occur when the bloom filter said the row existed, but it actually did not exist in absolute numbers.

**Column family bloom filter false positive ratio**

The fraction of all bloom filter checks resulting in a false positive. This should normally be at or below .01. A higher reading indicates that the bloom filter is likely too small.

**System performance metrics**

As with any database system, Cassandra performance greatly depends on underlying systems on which it is running. Tracking operating system metrics on your Cassandra nodes to watch for disk I/O, network, memory and CPU utilization trends can help you identify and troubleshoot hardware-related performance problems.

Monitoring Cassandra nodes for increasing disk and CPU utilization can help identify and remedy issues before performance degrades to unacceptable levels. The graphs in OpsCenter provide a quick way to view variations in OS metrics at a glance, and drill-down for specific data points. Especially in systems with heavy write loads, monitoring disk space is also important. It allows for advanced expansion planning while there is still adequate capacity to handle expansion and rebalancing operations.

**Memory**

Shows memory usage metrics in megabytes.

- Linux—Shows how much total system memory is currently used, cached, buffered or free.
- Windows—Shows the available physical memory, the cached operating system code, and the allocated pool-paged-resident and pool-nonpaged memory.
- Mac OS X—Shows free and used system memory.

**CPU**

Shows average percentages for CPU utilization metrics, which is the percentage of time the CPU was idle subtracted from 100 percent. CPU metrics can be useful for determining the origin of CPU performance reduction.

- Linux—Shows how much time the CPU devotes to system and user processes, to tasks stolen by virtual operating systems, to waiting for I/O to complete, and to processing nice tasks. High percentages of nice may indicate that other processes are crowding out Cassandra processes, while high percentages of iowait may indicate I/O contention. On fully virtualized environments like Amazon EC2, a Cassandra cluster under load may show high steal values while other virtual processors use the available system resources.
- Windows and Mac OS X—Shows how much time the CPU spends on user processes and system processes.

**Load**

The amount of work that a computer system performs. An idle computer has a load number of 0 and each process using or waiting for CPU time increments the load number by 1. Any value above one indicates that the machine was temporarily overloaded and some processes were required to wait. Shows minimum, average, and maximum OS load expressed as an integer.
**Disk usage (GB)**
Tracks growth or reduction in the amount of available disk space used. If this metric indicates a growth trend leading to high or total disk space usage, consider strategies to relieve it, such as adding capacity to the cluster. DataStax recommends leaving 30-50% free disk space for optimal repair and compaction operations.

**Disk usage (percentage)**
The percentage of disk space that is being used by Cassandra at a given time. When Cassandra is reading and writing heavily from disk, or building SSTables as the final product of compaction processes, disk usage values may be temporarily higher than expected.

**Disk throughput**
The average disk throughput for read and write operations, measured in megabytes per second. Exceptionally high disk throughput values may indicate I/O contention. This is typically caused by numerous compaction processes competing with read operations. Reducing the frequency of memtable flushing can relieve I/O contention.

**Disk rates**
- Linux and Windows—Averaged disk speed for read and write operations.
- Mac OS X—Not supported.

**Disk latency**
- Linux and Windows—Measures the average time consumed by disk seeks in milliseconds. Disk latency is among the higher-level metrics that may be useful to monitor on an ongoing basis by keeping this graph posted on your OpsCenter performance console. Consistently high disk latency may be a signal to investigate causes, such as I/O contention from compactions or read/write loads that call for expanded capacity.
- Mac OS X—Not supported.

**Disk request size**
- Linux and Windows—The average size in sectors of requests issued to the disk.
- Mac OS X—Not supported.

**Disk queue size**
- Linux and Windows—The average number of requests queued due to disk latency issues.
- Mac OS X—Not supported.

**Disk utilization**
- Linux and Windows—The percentage of CPU time consumed by disk I/O.
- Mac OS X—Not supported.

**Network traffic**
The speed at which data is received and sent across the network, measured in kilobytes per second.
Alert metrics

From the Alerts area of OpsCenter Enterprise Edition, you can configure alert thresholds for a number of Cassandra cluster-wide, column family, and operating system metrics. This proactive monitoring feature is available only in OpsCenter Enterprise Edition.

Commonly watched alert metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node down</td>
<td>When a node is not responding to requests, it is marked as down.</td>
</tr>
<tr>
<td>Write requests</td>
<td>The number of write requests per second. Monitoring the number of writes over a given time period can give you an idea of system write workload and usage patterns.</td>
</tr>
<tr>
<td>Write request latency</td>
<td>The response time (in milliseconds) for successful write operations. The time period starts when a node receives a client write request, and ends when the node responds back to the client.</td>
</tr>
<tr>
<td>Read requests</td>
<td>The number of read requests per second. Monitoring the number of reads over a given time period can give you an idea of system read workload and usage patterns.</td>
</tr>
<tr>
<td>Read request latency</td>
<td>The response time (in milliseconds) for successful read operations. The time period starts when a node receives a client read request, and ends when the node responds back to the client.</td>
</tr>
<tr>
<td>CPU usage</td>
<td>The percentage of time that the CPU was busy, which is calculated by subtracting the percentage of time the CPU was idle from 100 percent.</td>
</tr>
<tr>
<td>Load</td>
<td>Load is a measure of the amount of work that a computer system performs. An idle computer has a load number of 0 and each process using or waiting for CPU time increments the load number by 1.</td>
</tr>
</tbody>
</table>

Advanced Cassandra alert metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heap max</td>
<td>The maximum amount of shared memory allocated to the JVM heap for Cassandra processes.</td>
</tr>
<tr>
<td>Heap used</td>
<td>The amount of shared memory in use by the JVM heap for Cassandra processes.</td>
</tr>
<tr>
<td>JVM CMS collection count</td>
<td>The number of concurrent mark-sweep (CMS) garbage collections performed by the JVM per second.</td>
</tr>
<tr>
<td>JVM ParNew collection count</td>
<td>The number of parallel new-generation garbage collections performed by the JVM per second.</td>
</tr>
<tr>
<td>Metric</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JVM CMS collection time</td>
<td>The time spent collecting CMS garbage in milliseconds per second (ms/sec).</td>
</tr>
<tr>
<td>JVM ParNew collection time</td>
<td>The time spent performing ParNew garbage collections in ms/sec.</td>
</tr>
<tr>
<td>Data size</td>
<td>The size of column family data (in gigabytes) that has been loaded/inserted into Cassandra, including any storage overhead and system metadata.</td>
</tr>
<tr>
<td>Compactions pending</td>
<td>The number of compaction operations that are queued and waiting for system resources in order to run. The optimal number of pending compactions is 0 (or at most a very small number). A value greater than 0 indicates that read operations are in I/O contention with compaction operations, which usually manifests itself as declining read performance.</td>
</tr>
<tr>
<td>Total bytes compacted</td>
<td>The number of sstable data compacted in bytes per second.</td>
</tr>
<tr>
<td>Total compactions</td>
<td>The number of compactions (minor or major) performed per second.</td>
</tr>
<tr>
<td>Flush sorter tasks pending</td>
<td>The flush sorter process performs the first step in the overall process of flushing memtables to disk as SSTables. The optimal number of pending flushes is 0 (or at most a very small number).</td>
</tr>
<tr>
<td>Flushes pending</td>
<td>The flush process flushes memtables to disk as SSTables. This metric shows the number of memtables queued for the flush process. The optimal number of pending flushes is 0 (or at most a very small number).</td>
</tr>
<tr>
<td>Gossip tasks pending</td>
<td>Cassandra uses a protocol called gossip to discover location and state information about the other nodes participating in a Cassandra cluster. In Cassandra, the gossip process runs once per second on each node and exchanges state messages with up to three other nodes in the cluster. Gossip tasks pending shows the number of gossip messages and acknowledgments queued and waiting to be sent or received. The optimal number of pending gossip tasks is 0 (or at most a very small number).</td>
</tr>
<tr>
<td>Hinted handoff pending</td>
<td>While a node is offline, other nodes in the cluster will save hints about rows that were updated during the time the node was unavailable. When a node comes back online, its corresponding replicas will begin streaming the missed writes to the node to catch it up. The hinted handoff pending metric tracks the number of hints that are queued and waiting to be delivered once a failed node is back online again. High numbers of pending hints are</td>
</tr>
<tr>
<td>Metric</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>commonly seen when a node is brought back online after some down time. Viewing this metric can help you determine when the recovering node has been made consistent again.</td>
</tr>
<tr>
<td>Internal response pending</td>
<td>The number of pending tasks from various internal tasks such as nodes joining and leaving the cluster.</td>
</tr>
<tr>
<td>Manual repair tasks pending</td>
<td>The number of operations still to be completed when you run anti-entropy repair on a node. It will only show values greater than 0 when a repair is in progress. It is not unusual to see a large number of pending tasks when a repair is running, but you should see the number of tasks progressively decreasing.</td>
</tr>
<tr>
<td>Memtable postflushers pending</td>
<td>The memtable post flush process performs the final step in the overall process of flushing memtables to disk as SSTables. The optimal number of pending flushes is 0 (or at most a very small number).</td>
</tr>
<tr>
<td>Migrations pending</td>
<td>The number of pending tasks from system methods that have modified the schema. Schema updates have to be propagated to all nodes, so pending tasks for this metric can manifest in schema disagreement errors.</td>
</tr>
<tr>
<td>Miscellaneous tasks pending</td>
<td>The number of pending tasks from other miscellaneous operations that are not ran frequently.</td>
</tr>
<tr>
<td>Read requests pending</td>
<td>The number of read requests that have arrived into the cluster but are waiting to be handled. During low or moderate read load, you should see 0 pending read operations (or at most a very low number).</td>
</tr>
<tr>
<td>Read repair tasks pending</td>
<td>The number of read repair operations that are queued and waiting for system resources in order to run. The optimal number of pending read repairs is 0 (or at most a very small number). A value greater than 0 indicates that read repair operations are in I/O contention with other operations.</td>
</tr>
<tr>
<td>Replicate on write tasks pending</td>
<td>When an insert or update to a row is written, the affected row is replicated to all other nodes that manage a replica for that row. This is called the ReplicateOnWriteStage. This metric tracks the pending tasks related to this stage of the write process. During low or moderate write load, you should see 0 pending replicate on write tasks (or at most a very low number).</td>
</tr>
<tr>
<td>Request response pending</td>
<td>Streaming of data between nodes happens during operations such as bootstrap and decommission when one node sends large numbers of rows to</td>
</tr>
<tr>
<td>Metric</td>
<td>Definition</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Streams pending</td>
<td>Streaming of data between nodes happens during operations such as bootstrap and decommission when one node sends large numbers of rows to another node. The metric tracks the progress of the streamed rows from the sending node.</td>
</tr>
<tr>
<td>Write requests pending</td>
<td>The number of write requests that have arrived into the cluster but are waiting to be handled. During low or moderate write load, you should see 0 pending write operations (or at most a very low number).</td>
</tr>
</tbody>
</table>

**Advanced column family alert metrics**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local writes</td>
<td>The write load on a column family measured in operations per second. This metric includes all writes to a given column family, including write requests forwarded from other nodes.</td>
</tr>
<tr>
<td>Local write latency</td>
<td>The response time in milliseconds for successful write operations on a column family. The time period starts when nodes receive a write request, and ends when nodes respond.</td>
</tr>
<tr>
<td>Local reads</td>
<td>The read load on a column family measured in operations per second. This metric includes all reads to a given column family, including read requests forwarded from other nodes.</td>
</tr>
<tr>
<td>Local read latency</td>
<td>The response time in microseconds for successful read operations on a column family. The time period starts when a node receives a read request, and ends when the node responds.</td>
</tr>
<tr>
<td>Column family key cache hits</td>
<td>The number of read requests that resulted in the requested row key being found in the key cache.</td>
</tr>
<tr>
<td>Column family key cache requests</td>
<td>The total number of read requests on the row key cache.</td>
</tr>
<tr>
<td>Column family key cache hit rate</td>
<td>The key cache hit rate indicates the effectiveness of the key cache for a given column family by giving the percentage of cache requests that resulted in a cache hit.</td>
</tr>
<tr>
<td>Column family row cache hits</td>
<td>The number of read requests that resulted in the read being satisfied from the row cache.</td>
</tr>
<tr>
<td>Column family row cache requests</td>
<td>The total number of read requests on the row cache.</td>
</tr>
<tr>
<td>Column family row cache hit rate</td>
<td>The key cache hit rate indicates the effectiveness of the row cache for a given column family by</td>
</tr>
</tbody>
</table>
Using OpsCenter

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>Definition</td>
</tr>
<tr>
<td>giving the percentage of cache requests that</td>
<td>The size of the bloom filter files on disk.</td>
</tr>
<tr>
<td>resulted in a cache hit.</td>
<td>The number of false positives, which occur when the bloom filter said the row existed, but it actually did not exist in absolute numbers.</td>
</tr>
<tr>
<td>The fraction of all bloom filter checks</td>
<td>The current size of live SSTables for a column family. It is expected that SSTable size will grow over time with your write load, as</td>
</tr>
<tr>
<td>resulting in a false positive.</td>
<td>compaction processes continue doubling the size of SSTables. Using this metric together with SSTable count, you can monitor the current</td>
</tr>
<tr>
<td>The current size of the data directories</td>
<td>The current number of SSTables for a column family. When column family memtables are persisted to disk as SSTables, this metric</td>
</tr>
<tr>
<td>for the column family including space not</td>
<td>increases to the configured maximum before the compaction cycle is repeated. Using this metric together with live disk used, you</td>
</tr>
<tr>
<td>reclaimed by obsolete objects.</td>
<td>can monitor the current state of compaction for a given column family.</td>
</tr>
<tr>
<td>The number of pending reads and writes on</td>
<td>The number of pending reads and writes on a column family. Pending operations are an indication that Cassandra is not keeping up with</td>
</tr>
<tr>
<td>a column family.</td>
<td>the workload. A value of zero indicates healthy throughput.</td>
</tr>
</tbody>
</table>

**Advanced system alert metrics**

As with any database system, Cassandra performance greatly depends on underlying systems on which it is running. To configure advanced system metric alerts, you should first have an understanding of the baseline performance of your hardware and the averages of these system metrics when the system is handling a typical workload.

**Linux memory metrics**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory free</td>
<td>System memory that is not being used.</td>
</tr>
<tr>
<td>Memory used</td>
<td>System memory used by application processes.</td>
</tr>
<tr>
<td>Memory buffered</td>
<td>System memory used for caching file system metadata and tracking in-flight pages.</td>
</tr>
<tr>
<td>Memory shared</td>
<td>System memory that is accessible to CPUs.</td>
</tr>
<tr>
<td>Memory cached</td>
<td>System memory used by the OS disk cache.</td>
</tr>
</tbody>
</table>
### Linux CPU metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Percentage of time the CPU is idle.</td>
</tr>
<tr>
<td>Iowait</td>
<td>Percentage of time the CPU is idle and there is a pending disk I/O request.</td>
</tr>
<tr>
<td>Nice</td>
<td>Percentage of time spent processing prioritized tasks. Niced tasks are also counted in system and user time.</td>
</tr>
<tr>
<td>Steal</td>
<td>Percentage of time a virtual CPU waits for a real CPU while the hypervisor services another virtual processor.</td>
</tr>
<tr>
<td>System</td>
<td>Percentage of time allocated to system processes.</td>
</tr>
<tr>
<td>User</td>
<td>Percentage of time allocated to user processes.</td>
</tr>
</tbody>
</table>

### Linux Disk metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk usage</td>
<td>Percentage of disk space Cassandra uses at a given time.</td>
</tr>
<tr>
<td>Free disk space</td>
<td>Available disk space in GB.</td>
</tr>
<tr>
<td>Used disk space</td>
<td>Used disk space in GB.</td>
</tr>
<tr>
<td>Disk read throughput</td>
<td>Average disk throughput for read operations in megabytes per second. Exceptionally high disk throughput values may indicate I/O contention.</td>
</tr>
<tr>
<td>Disk write throughput</td>
<td>Average disk throughput for write operations in megabytes per second.</td>
</tr>
<tr>
<td>Disk read rate</td>
<td>Averaged disk speed for read operations.</td>
</tr>
<tr>
<td>Disk write rate</td>
<td>Averaged disk speed for write operations.</td>
</tr>
<tr>
<td>Disk latency</td>
<td>Average time consumed by disk seeks in milliseconds.</td>
</tr>
<tr>
<td>Disk request size</td>
<td>Average size in sectors of requests issued to the disk.</td>
</tr>
<tr>
<td>Disk queue size</td>
<td>Average number of requests queued due to disk latency.</td>
</tr>
<tr>
<td>Disk utilization</td>
<td>Percentage of CPU time consumed by disk I/O.</td>
</tr>
</tbody>
</table>

### Windows memory metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available memory</td>
<td>Physical memory that is not being used.</td>
</tr>
<tr>
<td>Pool nonpaged</td>
<td>Physical memory that stores the kernel and other system data structures.</td>
</tr>
</tbody>
</table>
### Using OpsCenter

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pool paged resident</td>
<td>Physical memory allocated to unused objects that can be written to disk to free memory for reuse.</td>
</tr>
<tr>
<td>System cache resident</td>
<td>Physical pages of operating system code in the file system cache.</td>
</tr>
</tbody>
</table>

### Windows CPU metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Percentage of time the CPU is idle.</td>
</tr>
<tr>
<td>Privileged</td>
<td>Percentage of time the CPU spends executing kernel commands.</td>
</tr>
<tr>
<td>User</td>
<td>Percentage of time allocated to user processes.</td>
</tr>
</tbody>
</table>

### Windows Disk metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk usage</td>
<td>Percentage of disk space Cassandra uses at a given time.</td>
</tr>
<tr>
<td>Free disk space</td>
<td>Available disk space in GB.</td>
</tr>
<tr>
<td>Used disk space</td>
<td>Used disk space in GB.</td>
</tr>
<tr>
<td>Disk read throughput</td>
<td>Average disk throughput for read operations in megabytes per second. Exceptionously high disk throughput values may indicate I/O contention.</td>
</tr>
<tr>
<td>Disk write throughput</td>
<td>Average disk throughput for write operations in megabytes per second.</td>
</tr>
<tr>
<td>Disk read rate</td>
<td>Averaged disk speed for read operations.</td>
</tr>
<tr>
<td>Disk write rate</td>
<td>Averaged disk speed for write operations.</td>
</tr>
<tr>
<td>Disk latency</td>
<td>Average time consumed by disk seeks in milliseconds.</td>
</tr>
<tr>
<td>Disk request size</td>
<td>Average size in sectors of requests issued to the disk.</td>
</tr>
<tr>
<td>Disk queue size</td>
<td>Average number of requests queued due to disk latency.</td>
</tr>
<tr>
<td>Disk utilization</td>
<td>Percentage of CPU time consumed by disk I/O.</td>
</tr>
</tbody>
</table>

### Mac OS X memory metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free memory</td>
<td>System memory that is not being used.</td>
</tr>
<tr>
<td>Used memory</td>
<td>System memory that is being used by application processes.</td>
</tr>
</tbody>
</table>
### Mac OS X CPU metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Percentage of time the CPU is idle.</td>
</tr>
<tr>
<td>System</td>
<td>Percentage of time allocated to system processes.</td>
</tr>
<tr>
<td>User</td>
<td>Percentage of time allocated to user processes.</td>
</tr>
</tbody>
</table>

### Mac OS X disk metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk usage</td>
<td>Percentage of disk space Cassandra uses at a given time.</td>
</tr>
<tr>
<td>Free space</td>
<td>Available disk space in GB.</td>
</tr>
<tr>
<td>Used disk space</td>
<td>Used disk space in GB.</td>
</tr>
<tr>
<td>Disk throughput</td>
<td>Average disk throughput for read/write operations in megabytes per second.</td>
</tr>
<tr>
<td></td>
<td>Exceptionally high disk throughput values may indicate I/O contention.</td>
</tr>
</tbody>
</table>

### Managing backups and restoring from backups

Using OpsCenter Enterprise Edition, you can take, schedule, and manage backups across all registered clusters.

A backup is a snapshot of all on-disk data files (SSTable files) stored in the data directory. Backups are taken per keyspace and while the system is online. A backup first flushes all in-memory writes to disk, then makes a hard link of the SSTable files for each keyspace. Backups are stored in the snapshots directory of the column family that's being snapshotted. For example, /var/lib/cassandra/data/cfs/snapshots.

You must have enough free disk space on the node to accommodate making snapshots of your data files. A single snapshot requires little disk space. However, snapshots will cause your disk usage to grow more quickly over time because a snapshot prevents old obsolete data files from being deleted. OpsCenter Data Backups allows you to specify a schedule to remove old backups and prevent backups from being taken when disk space falls below a specified level.

**Note:** OpsCenter Data Backups does not show or manage manual snapshots taken using the nodetool snapshot command.

### Scheduling a backup

To schedule a backup:

1. In the OpsCenter Dashboard, click **Data Backups**.
2. Click **Schedule Backup**.
3. In Add Backup, select the backup parameters:
   a) **Select a Keyspace to backup**—Select the keyspace that you want to back up.
   b) **Schedule**—Select a frequency and timezone for your backup. GMT is the default timezone.
   c) **Cleanup**—Choose a frequency to remove old backups. (If not specified, you should manually cleanup snapshots.)
4. Click Save.
5. To set the percentage of free disk space at which backups are prevented, click **Configure** and then enter the appropriate information.
Using OpsCenter

The percentage of free disk space that you set applies to all nodes in the cluster. Detailed information about the backup is recorded in the Event Log.

Restoring from a backup
You can restore from any local backups that have been run by OpsCenter, but not from snapshots run from nodetool. You can pick any subset of column families that exist in the snapshot to restore.

To restore a backup:
1. Click Data Backups.
2. Find the backup you wish to restore in the list of backups.
3. Click Restore.
4. Choose which keyspaces to restore.
5. (Optional) Choose individual column families within a chosen keyspace to restore.
6. (Optional) Click the Truncate/delete existing data before restore checkbox.
7. (Optional) Click the Throttle stream throughput at ____ MB checkbox and enter the desired throttling value.
8. Click Restore.

Data modeling

Keyspaces
You can create a new keyspace or manage keyspaces.

Selecting the Data Modeling area in the OpsCenter console lists the keyspaces in the cluster that you are monitoring.

When you create a keyspace, you give it a name, choose a replica placement strategy, the total number of replicas you want, and how those replicas are divided across your data centers (if you have a multiple data center cluster).

Creating a keyspace

To create a new keyspace.
1. Select Schema in the OpsCenter console.
2. Select Add in the Schema section of OpsCenter.
3. Give the keyspace a name. Keyspace names should not contain spaces or special characters or exceed the filename size limit of your operating system (for example, 255 bytes on most Linux file systems). Keyspace names are case sensitive.
4. Set the replica placement strategy. The replica placement strategy (along with the cluster-configured snitch) determines how replicas are placed on nodes throughout the cluster. Use one of three built-in replica placement strategies:
   a) SimpleStrategy—Single data center, rack unaware replica placement. This is the default strategy.
   b) NetworkTopologyStrategy—Single or multiple data center, rack aware replica placement. This is the recommended strategy.
   c) OldNetworkTopologyStrategy—Two data centers only, rack aware replica placement. This strategy is deprecated.
5. Choose how many total copies that you want of your keyspace data (replication factor). The NetworkTopologyStrategy requires you to configure how many replicas you want per data center. The data center name you enter should match the data center name used by your cluster-configured snitch. Make sure to name the data center(s) correctly according to your snitch configuration.
6. If you do not want to start defining column families within your new keyspace right away, uncheck the I would like to create a **Column Family** checkbox.

7. Click **Save Keyspace**.

**Managing a keyspace**

To manage a keyspace:

1. Select **Schema** in the OpsCenter console.
2. From the list of keyspaces, select one of the keyspaces.
   
   In **Keyspace Settings**, the replica placement strategy options for the keyspace appear.
3. From the list of column families (below Settings), select a column family to view its properties and to view or change performance tuning metrics.
4. Click **Add** in the Column Family to add a column family to the keyspace.
5. Click **Delete** to delete the keyspace.

**Managing column families**

When you create a column family in Cassandra using an application, the CLI, or CQL 2 or earlier, the column family appears in OpsCenter. You can use Schema to manage the column family.

You can also create one type of column family: the dynamic column family. Dynamic column families are those that do not specify column names or values when the column family is created. An application typically supplies this metadata. CQL 3, the default query language in Cassandra, does not support dynamic column families. Earlier versions of CQL and the CLI support dynamic column families.

This version of OpsCenter does not support defining static column families (per-column meta data), row key data types, or schema information for super column sub-columns described in Cassandra 1.0, or earlier.

**Creating a dynamic column family**

To create a new dynamic column family:

1. Select **Schema** in the OpsCenter console.
2. From the list of keyspaces, select the keyspace to contain the column family.
3. Give the column family a name.
   
   Column family names should not contain spaces or special characters and cannot exceed the filename size limit of your operating system (255 bytes on most Linux file systems).
   
   By default, column families are created with standard columns (**column_type**: **Standard**). If you want a column family containing super columns choose **column_type**: **Super**.
4. Use **compare_with** to set the default data type for column names (or super column names). Setting the default data type also sets the column sort order for the column family. For example, choosing **LongType** would sort the columns within a row in numerical order. The sort order cannot be changed after a column family is created, so choose wisely.
5. Use **default_validation_class** to set the default data type for column values (or super column sub-column values). Always set this for dynamic column families.
6. Click **Save Column Family**.

**Managing a column family**

To manage a column family:

1. Select **Schema** in the OpsCenter console.
2. From the list of keyspaces, select a keyspace.
   
   The #CFs columns shows how many column families each keyspace contains.
3. From the list of the column families, select a column family.
Using OpsCenter

Click one of the following buttons:

- **Add**—See above.
- **Delete**—Completely removes the column family from the keyspace. You may select more than one column family in a keyspace to delete.
- **View Metrics**—Presents metrics for a column family. In the Metric Options dialog, select a column family (CF) metric to view. To aggregate measurements across the entire cluster, all nodes in the data center, or in a particular node, select Cluster Wide, All Nodes, or the IP address of a node. At this point, you can add a graph of the measurements to the Performance Metrics area, or choose a different column family to measure.
- **Truncate**—Deletes all data from the column family but does not delete the column family itself. Removal of the data is irreversible.

4. When you select a column family, you see a list of manageable attributes: Properties (fully editable), Metadata (Add or Delete), and Secondary Indexes (Add or Delete).

Browsing data

**Browsing a Cassandra database**

To browse a Cassandra database

1. Select Data Explorer in the OpsCenter console. A list of keyspaces in the cluster appears. By default, the list includes the OpsCenter keyspace, which contains column families of data in the cluster.
2. Click one of the keyspaces. For example, click the OpsCenter keyspace. The list of column families appears.
3. Click a column family. For example, click `events_timeline` and expand the OpsCenter console window, so you can see more values. A row keys, columns, and data values of the `events_timeline` column family appear in tabular format. You may notice that some data values are hex-encoded, not human-readable values, and other values are perfectly readable. Internally, Cassandra stores row key names, column names and column values as hex byte arrays. If possible, OpsCenter converts data to text you can read.
4. Click the heading row of the table to see the sort control (the arrow on the right). Toggle the sort control to sort the rows in ascending or descending order.
5. To browse through the data, use the scroll bar or click Next to page through the data.

**Examining row data**

To zoom in on a particular row:

1. Assuming the `events_timeline` column family is still selected, type the key for a row in the Key text entry box. For example, type the row key `timeline_start`. If you set up a secondary index on the column family, the Key drop-down lists key choices.
2. Click the search control (the hourglass icon). The columns and values of the row appear.

**Removing all traces of Cassandra or DSE packages**

After a failed attempt to provision a node or a cluster, you must remove all traces of Cassandra or DSE packages from the affected nodes.

1. Log into each node.
2. Run the following BASH script on each node using `sudo`.

---

---
# Stop services
/etc/init.d/cassandra stop
/etc/init.d/dse stop
/etc/init.d/opscenter-agent stop

# Remove packages
PACKAGES=(dsc dsc1.1 dsc12 cassandra apache-cassandra dsc-demos \
dse dse-libhadoop-native dse-libhadoop dse-libcassandra dse-hive dse-\libhive dse-pig \
dse-libpig dse-demos dse-libsqoop dse-libtomcat dse-liblog4j dse-libso\lr
dse-libmahout dse-full)
DEB_PACKAGES=(python-cql python-thrift-basic)
RPM_PACKAGES=(python26-cql python26-thrift)
if [ `which dpkg` ]; then
    PLIST=(${PACKAGES[@]} ${DEB_PACKAGES[@]})
dpkg -P ${PLIST[*]}
    rm -rf /etc/apt/sources.list.d/datastax.list
else
    PLIST=(${PACKAGES[@]} ${RPM_PACKAGES[@]})
yum -y remove ${PLIST[*]}
    rm -rf /etc/yum.repos.d/datastax.repo
fi

# Cleanup log and configuration files
rm -rf /var/lib/cassandra/* /var/log/{cassandra,hadoop,hive,pig}/* /etc/{cass\ndra,dse}/* /
usr/share/{dse,dse-demos} /etc/default/{dse,cassandra}

• copy, paste, and run the script from the command line
• save the script to a local file and copy to each node to run from the command line
Upgrading

Before upgrading, check that the Cassandra or DataStax Enterprise version is compatible with the upgraded OpsCenter version.

<table>
<thead>
<tr>
<th>OpsCenter version</th>
<th>Cassandra version</th>
<th>DataStax Enterprise version</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>1.1, 1.2</td>
<td>2.x, 3.0.x, 3.1.x</td>
</tr>
<tr>
<td>3.1</td>
<td>1.0, 1.1, 1.2</td>
<td>1.0, 2.x, 3.0.x</td>
</tr>
<tr>
<td>3.0</td>
<td>1.0, 1.1, 1.2</td>
<td>1.0, 2.x, 3.0</td>
</tr>
<tr>
<td>2.1</td>
<td>0.8, 1.0, 1.1</td>
<td>1.0, 2.0</td>
</tr>
<tr>
<td>2.0</td>
<td>0.8, 1.0, 1.1</td>
<td>1.0, 2.0</td>
</tr>
<tr>
<td>1.4.x</td>
<td>0.7, 0.8, 1.0, 1.1</td>
<td>1.0, 2.0</td>
</tr>
</tbody>
</table>

Use the information in this section to upgrade to OpsCenter 3.2 from 2.1.x, 2.0, 1.4.x, or 1.3.x.

Upgrading package installations

Describes installing the new OpsCenter 3.2 package and restarting the opscenterd daemon.

To upgrade to OpsCenter 3.2:

1. On the OpsCenter daemon host, run the appropriate command to update the packages:
   - Debian or Ubuntu
     
     # apt-get update
   - RHEL or CentOS
     
     # yum clean all

2. Install the upgraded OpsCenter package:
   - Debian or Ubuntu:
     
     # apt-get install opscenter
   - RHEL or CentOS:
     
     # yum install opscenter

3. If the package manager prompts you for options regarding opscenterd.conf, choose to keep your currently installed version.

4. Restart the OpsCenter daemon.

   # service opscenterd restart

Upgrading tarball installations

1. Download and extract the new tarball.
2. Copy the following files and directories from the old tarball installation directory to the new one.

   - `conf/clusters/*`
   - `conf/event-plugins/*`
   - `conf/install_id`
   - `conf/log4j.properties`
   - `conf/opscenterd.conf`
   - `conf/ssl.conf`
   - `ssl/*`

3. Stop the opscenterd instance (if it is running) and start it from the new tarball installation directory.

4. Upgrade the agents either through the GUI or by manually installing from the new tarballs.

### Uprading agents

If OpsCenter agents require upgrading for the new release, you are prompted to do by a Fix link located near the top of the OpsCenter console.

For information about installing the upgraded agents, see Installing OpsCenter agents.

#### From tarballs

If the agents will be upgraded manually with tarballs, copy the new agent.tar.gz to all nodes, extract it, and copy the following files from the old agent tarball directories to the new ones:

   - `conf/*`
   - `ssl/*`
Troubleshooting

This section lists some common problems experienced with OpsCenter and solutions or workarounds.

Internet Explorer web browser not supported

If you try to load the OpsCenter client in Microsoft Internet Explorer, a dialog displays informing you that it is not supported.

OpsCenter is only supported on the latest versions of:
• Apple Safari
• Google Chrome
• Mozilla Firefox

The SSTables in this snapshot '<tag>' are not compatible

If you receive an error message that includes “The SSTables in this snapshot '<tag>' are not compatible with the current version of Cassandra”, it means you must upgrade your snapshot to the current major version of Cassandra or DSE.

How to
1. Log in to each node.
2. Run the sstableupgrade script for every keyspace and column family you wish to restore, passing it the keyspace, column family, and OpsCenter snapshot tag that you got from the error message.
   How you run the script depends on how you installed Cassandra or DSE.
3. Retry the restore from OpsCenter.

OpsCenter data growing too large

A bug, which has been fixed in 3.2.1, was not setting a TTL on metrics data being collected for a managed cluster. Depending on your environment, this could cause some column families in the OpsCenter keyspace to grow too large. The most common offenders will be the pdps (raw data points) and rollups60 (1m data points) column families.

If any of the column families have grown too large, you can truncate them to reclaim the space. If you are not comfortable losing historical data for that granularity (for example, 1m), please contact DataStax support.

Cannot create a keyspace

Due to a Python 2.6 or earlier bug, some users experience a problem adding a keyspace using Data Modeling OpsCenter features. OpsCenter cannot save a newly created keyspace.

Upgrading Python generally fixes this problem.

Error exceptions.ImportError:libssl.so.0.9.8

Occurs when OpenSSL 1.0.0 is installed on RHEL 5.x, CentOS 5.x, OEL 5.5, Debian, or Ubuntu systems:
message.exceptions.ImportError: libssl.so.0.9.8
To fix, you can do either of the following:

- Install OpenSSL 0.9.8:
  1. RHEL-based systems: `sudo yum install openssl098`
  2. Debian-based systems: `sudo apt-get install libssl0.9.8`

- Install Python libssl 1.x:
  1. Remove all OpenSSL modules from the OpsCenter installation `lib` directory. You do not have to remove them globally.
     - Packaged installs: `/usr/share/opscenter/content/lib`
     - Binary installs: `<install_location>/lib`
     - You can easily find these directories using:
       `find . -name OpenSSL -type d | xargs rm -rf`
  2. Install the latest pyOpenSSL module:
     - Debian-based systems: `apt-get install python-openssl`
     - RHEL-based systems: `yum install python-openssl`
  3. Start or restart opscenterd.

**Python used to run OpsCenter not built with SSL**

In order to protect your AWS credentials when launching EC2 instances, OpsCenter needs to use HTTPS, but if the version of Python that is running opscenterd was not compiled with SSL support OpsCenter will not run even if SSL has been disabled in the configuration file.

To resolve the issue, first ensure that OpenSSL 0.9.8 is installed on your system. If you have compiled Python manually, it is recommended that you install Python 2.6+ through your package manager. On CentOS and RedHat Enterprise Linux, this is most easily done through EPEL packages.

If you must compile Python manually, make sure that SSL support is enabled. This blog post explains the process for Python 2.5, but the same steps will work for Python 2.6 or 2.7.

**OpsCenter agent port setting conflict**

If you have a problem with OpsCenter, check for conflicts in port settings. The OpsCenter Agent uses port 7199 by default. If you have not changed the default port, check that Cassandra or another process on the node, is not set up to use this port.

If you set the OpsCenter Agent port to a host name instead of an IP address, the DNS provider must be online to resolve the host name. If the DNS provider is not online, intermittent problems should be expected.

**Limiting the metrics collected by OpsCenter**

If you have many column families, the number of metrics OpsCenter collects can become quite large. For information about how to reduce the number of keyspaces or column families that are monitored, see Controlling data collection.
Troubleshooting

Java not installed or JAVA_HOME environment variable not set

If Java is not installed or if OpsCenter cannot find JAVA_HOME, you may see an error such as:

```
/usr/share/opscenter-agent/bin/opscenter-agent: line 98: exec: -X: invalid option
exec: usage: exec [-cl ] [-a name ] [ command [arguments ... ]]
[redirection ... ]
```

To correct this problem, install Java or set JAVA_HOME:

```
export JAVA_HOME = <path_to_java>
```

Insufficient user resource limits errors

Insufficient resource limits may result in an insufficient nofiles error:

```
2012-08-13 11:22:51-0400 [] INFO: Could not accept new connection (EMFILE )
```

See Recommended settings under Insufficient user resource limits errors in the Cassandra documentation.

Problems with provisioning

General troubleshooting steps

Ensure firewalls are properly opened between the opscenterd machine and each node.

Check the following files (on any of the nodes having problems) for any errors:

- `/var/log/cassandra/system.log`
- `/var/log/opscenter-agent/agent.log`

Verify that Cassandra (or DSE) was not previously installed on any of the machines; if it was, all binaries, configuration files, logs, etc. must be cleared out first.

Invalid repository credentials

Invalid repository credentials

**Debian**

When installing DSE, if you enter invalid values for DataStax Credentials, an error dialog displays text along the lines of:

```
Installation stage failed: Installation failed on node 172.16.1.2: 'apt-get update' failed.
```

Clicking Details displays the entire output from both stdout and stderr. If, in this output, you see the "401 Unauthorized", it means that the credentials entered were invalid.

**RHEL**

When installing DSE, if you enter invalid values for DataStax Credentials, an error dialog displays text along the lines of:

```
Installation failed on node 172.16.1.2: 'yum install' failed.
```

Clicking Details displays the entire output from both stdout and stderr. If, in this output, you see "The requested URL returned error: 401", it means that the credentials used were invalid.
Troubleshooting

Timed out waiting for Cassandra to start

If you receive this error, it most likely means that the Cassandra process failed to start on one or more nodes. You should look in `/var/log/cassandra/system.log` for any errors that may need to be resolved.

The following packages are already installed

If you receive an error that starts with this message, it means Cassandra (or DSE) is already installed on the system. OpsCenter provisioning requires that any instances of Cassandra (or DSE) be completely removed or purged before provisioning a new cluster.

Agents cannot connect to opscenterd

If you receive an error message that includes "The installed agent doesn't seem to be responding", there is most likely a firewall issue preventing the installed agent from connecting to the opscenterd machine. You should ensure that port 61620 is open on the opscenterd machine and check the agent logs.

Removing all Cassandra or DSE files after failed provisioning

If you provision a new cluster or add a node to an existing cluster via the OpsCenter UI and an error occurs, you can retry the request, but before you retry you must **remove all traces** of Cassandra or DSE packages from the nodes in the cluster. You can also remove simply remove the files if you do not wish to retry the provisioning.
OpsCenter API reference

The OpsCenter API facilitates the development of websites and programs to retrieve data and perform Cassandra administrative actions. The OpsCenter API includes RESTful requests for programmatically performing the same set of operations as the OpsCenter GUI.
Release Notes

Information about new features and resolved issues in the following releases.

3.2.2

Resolved issues

• Fixed issues with collecting and displaying Solr search metrics.

3.2.1

Resolved issues

• Fixed the primary range option in the Repair action.
• Fixed an issue that caused metrics data to never expire. (How to fix the problem.)
• Fixed editing column families with column metadata.
• Other minor bug fixes.

3.2

New features

• Improved support for large clusters (~100 nodes).
• Faster graph loading times (~10x, or more, in most cases).
• Added support for CQL3 column families (limited Data Explorer support); data is only viewable if the column family was created with the COMPACT STORAGE option).
• Improved support for vnodes.
• Added the ability to jump to a specific time range in the Performance section.
• Added the ability to submit forms via the Enter key.
• Increased the number of nodes that can be rebalanced from 100 to 200.
• Added hostname of the problem node to alert emails.

Resolved issues

• Added support for restoring column families with thousands of SSTables.
• Fixed memory leak in Performance section which caused some browser crashes.
• Fixed unknown fields being removed from cassandra.yaml when configuring nodes.
• Fixed the partitioner option in cassandra.yaml from being overwritten if the original value was not a full classname.
• Fixed the UI failing to load on mobile devices, including Chromebooks.
• Fixed display of cache hit rate metrics.
• Fixed viewing rows with Int32 types in the Data Explorer.

3.1.1

Resolved issues

• Fixed an issue with opscenterd not starting up when certain types of CQL3 column families exist.
3.1

New features
- Column families created via CQL3 are recognized and monitored.
- Added support for provisioning nodes with vnodes enabled.

Resolved issues
- Fixed OpsCenter not running without the proper SSL library being installed, even if SSL feature has been disabled, has been fixed.
- Fixed an issue with editing keyspace properties if another keyspace was recently updated.
- Fixed loading Performance presets with special characters in the name.
- Fixed agent hitting out of memory exception when cleaning up some backups.
- Fixed the List View rendering when nodes have long hostnames.
- Fixed an issue with adding nodes to an existing DSE cluster when some nodes were down.
- Fixed an issue when provisioning nodes using RackInferringSnitch and hostnames.

3.0.2

Resolved issues
- When adding internal Cassandra authentication to a cluster, the cluster’s edited configuration can now be saved. Previously an error was reported: Unable to connect to cluster.
- Scrolling long lists of column families in the Schema and Data Explorer panes has been implemented.
- Agents now receive configuration changes without having to be restarted.
- Compaction status now displays properly in List View or node popup.
- OpsCenter no longer becomes unresponsive (caused by an agent hanging) if there are too many files in the snapshots directory.

3.0

New features
- Visual cluster creation and management.
  - Managing settings.
  - Restarting a cluster.
- Database restore.
- Customizable backups.
- Software update notifications.
- Improved object creation and management.
- Automated collection of diagnostic data.
- Offloading of metadata repository to secondary clusters.
- DSE security and encryption

Resolved issues
- Fixed node popup in Ring/Physical view going blank.
- Fixed agent data collection being interrupted on error.
- Fixed display of some data in Performance section.
- Fixed list of nodes in List View jumping around when compactions are running.
- Fixed several minor issues in the UI.
- Fixed several issues with installing opscenterd and the agents.
2.1.3

New features
• OpsCenter Enterprise Edition 2.1.3 can monitor and administer Cassandra 1.2 as described in the documentation unless you enable virtual nodes. When you enable virtual nodes, OpsCenter chooses a single token for each node for operations such as collecting metrics. Attempting to move nodes, rebalance nodes, and perform other tasks involving token ranges is not supported.

Resolved issues
• Fixed issue processing keyspaces that only exist in 1 datacenter.
• Fixed data collection issues during compactions on secondary indexes.

2.1.2

New features

Resolved issues

2.1

New features
• Browsing data in the Cassandra database.
• Generation of a PDF report about the cluster.
• Capability to handle thousands of column families efficiently.
• Online feedback form.
• Optional HTTPS support.
• Installation of agents to RHEL and Debian clusters regardless of the location of the opscenterd.conf configuration file.
• Truncation of column families. Truncation removes the data but not the column family itself.

2.0

New features
• Multi-cluster management. You can manage multiple cluster from the same OpsCenter instance.
• New Overview page for monitoring multiple clusters. It shows a condensed view of each cluster’s Dashboard. It is visible only when multiple clusters exist.
• Perform and schedule operations from OpsCenter.
• Point OpsCenter at existing Cassandra clusters, where you can add, modify, or delete clusters from OpsCenter.
• Adds support and metrics for DataStax Enterprise 2.0 Search nodes.
• No longer compatible with Cassandra 0.7.
Send us feedback

Thanks for using OpsCenter Enterprise Edition. Please take a moment to let us know what you think.

1. Start OpsCenter and click **Feedback** at the top of the console.
   The feedback form appears.

2. Enter your name, email address, and feedback.

3. Click **Send Feedback**.
   We appreciate your comments.