

Hortonworks Data Platform

Apache Ambari Operations

(October 30, 2017)

Hortonworks Data Platform: Apache Ambari Operations

Copyright © 2012-2017 Hortonworks, Inc. Some rights reserved.

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, [training](#) and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the [Hortonworks Data Platform](#) page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the [Support](#) or [Training](#) page. Feel free to [Contact Us](#) directly to discuss your specific needs.



Except where otherwise noted, this document is licensed under
Creative Commons Attribution ShareAlike 4.0 License.
<http://creativecommons.org/licenses/by-sa/4.0/legalcode>

Table of Contents

1. Ambari Operations: Overview	1
1.1. Ambari Architecture	1
1.2. Accessing Ambari Web	2
2. Understanding the Cluster Dashboard	5
2.1. Viewing the Cluster Dashboard	5
2.1.1. Scanning Operating Status	6
2.1.2. Viewing Details from a Metrics Widget	7
2.1.3. Linking to Service UIs	7
2.1.4. Viewing Cluster-Wide Metrics	8
2.2. Modifying the Cluster Dashboard	9
2.2.1. Replace a Removed Widget to the Dashboard	10
2.2.2. Reset the Dashboard	10
2.2.3. Customizing Metrics Display	11
2.3. Viewing Cluster Heatmaps	11
3. Managing Hosts	13
3.1. Understanding Host Status	13
3.2. Searching the Hosts Page	14
3.3. Performing Host-Level Actions	17
3.4. Managing Components on a Host	18
3.5. Decommissioning a Master or Slave	19
3.5.1. Decommission a Component	20
3.6. Delete a Component	20
3.7. Deleting a Host from a Cluster	21
3.8. Setting Maintenance Mode	21
3.8.1. Set Maintenance Mode for a Service	22
3.8.2. Set Maintenance Mode for a Host	22
3.8.3. When to Set Maintenance Mode	23
3.9. Add Hosts to a Cluster	24
3.10. Establishing Rack Awareness	25
3.10.1. Set the Rack ID Using Ambari	26
3.10.2. Set the Rack ID Using a Custom Topology Script	27
4. Managing Services	28
4.1. Starting and Stopping All Services	29
4.2. Displaying Service Operating Summary	29
4.2.1. Alerts and Health Checks	30
4.2.2. Modifying the Service Dashboard	30
4.3. Adding a Service	32
4.4. Performing Service Actions	36
4.5. Rolling Restarts	36
4.5.1. Setting Rolling Restart Parameters	37
4.5.2. Aborting a Rolling Restart	38
4.6. Monitoring Background Operations	38
4.7. Removing A Service	40
4.8. Operations Audit	40
4.9. Using Quick Links	40
4.10. Refreshing YARN Capacity Scheduler	41
4.11. Managing HDFS	41
4.11.1. Rebalancing HDFS	42

4.11.2. Tuning Garbage Collection	42
4.11.3. Customizing the HDFS Home Directory	43
4.12. Managing Atlas in a Storm Environment	43
5. Managing Service High Availability	44
5.1. NameNode High Availability	44
5.1.1. Configuring NameNode High Availability	44
5.1.2. Rolling Back NameNode HA	49
5.1.3. Managing Journal Nodes	59
5.2. ResourceManager High Availability	64
5.2.1. Configure ResourceManager High Availability	64
5.2.2. Disable ResourceManager High Availability	65
5.3. HBase High Availability	67
5.4. Hive High Availability	72
5.4.1. Adding a Hive Metastore Component	72
5.4.2. Adding a HiveServer2 Component	72
5.4.3. Adding a WebHCat Server	73
5.5. Storm High Availability	73
5.5.1. Adding a Nimbus Component	73
5.6. Oozie High Availability	74
5.6.1. Adding an Oozie Server Component	74
5.7. Apache Atlas High Availability	75
5.8. Enabling Ranger Admin High Availability	77
6. Managing Configurations	78
6.1. Changing Configuration Settings	78
6.1.1. Adjust Smart Config Settings	79
6.1.2. Edit Specific Properties	80
6.1.3. Review and Confirm Configuration Changes	80
6.1.4. Restart Components	82
6.2. Manage Host Config Groups	82
6.3. Configuring Log Settings	85
6.4. Set Service Configuration Versions	87
6.4.1. Basic Concepts	87
6.4.2. Terminology	88
6.4.3. Saving a Change	88
6.4.4. Viewing History	89
6.4.5. Comparing Versions	90
6.4.6. Reverting a Change	91
6.4.7. Host Config Groups	91
6.5. Download Client Configuration Files	92
7. Administering the Cluster	94
7.1. Using Stack and Versions Information	94
7.2. Viewing Service Accounts	96
7.3. Enabling Kerberos and Regenerating Keytabs	97
7.3.1. Regenerate Key tabs	98
7.3.2. Disable Kerberos	98
7.4. Enable Service Auto-Start	99
8. Managing Alerts and Notifications	102
8.1. Understanding Alerts	102
8.1.1. Alert Types	103
8.2. Modifying Alerts	104
8.3. Modifying Alert Check Counts	104

8.4. Disabling and Re-enabling Alerts	105
8.5. Tables of Predefined Alerts	105
8.5.1. HDFS Service Alerts	106
8.5.2. HDFS HA Alerts	109
8.5.3. NameNode HA Alerts	110
8.5.4. YARN Alerts	111
8.5.5. MapReduce2 Alerts	112
8.5.6. HBase Service Alerts	112
8.5.7. Hive Alerts	113
8.5.8. Oozie Alerts	114
8.5.9. ZooKeeper Alerts	114
8.5.10. Ambari Alerts	114
8.5.11. Ambari Metrics Alerts	115
8.5.12. SmartSense Alerts	116
8.6. Managing Notifications	116
8.7. Creating and Editing Notifications	116
8.8. Creating or Editing Alert Groups	118
8.9. Dispatching Notifications	119
8.10. Viewing the Alert Status Log	119
8.10.1. Customizing Notification Templates	120
9. Using Ambari Core Services	123
9.1. Understanding Ambari Metrics	123
9.1.1. AMS Architecture	123
9.1.2. Using Grafana	124
9.1.3. Grafana Dashboards Reference	129
9.1.4. AMS Performance Tuning	167
9.1.5. AMS High Availability	172
9.1.6. AMS Security	174
9.2. Ambari Log Search (Technical Preview)	179
9.2.1. Log Search Architecture	179
9.2.2. Installing Log Search	180
9.2.3. Using Log Search	180
9.3. Ambari Infra	183
9.3.1. Archiving & Purging Data	184
9.3.2. Performance Tuning for Ambari Infra	190

1. Ambari Operations: Overview

Hadoop is a large-scale, distributed data storage and processing infrastructure using clusters of commodity hosts networked together. Monitoring and managing such complex distributed systems is not simple. To help you manage the complexity, Apache Ambari collects a wide range of information from the cluster's nodes and services and presents it to you in an easy-to-use, centralized interface: Ambari Web.

Ambari Web displays information such as service-specific summaries, graphs, and alerts. You use Ambari Web to create and manage your HDP cluster and to perform basic operational tasks, such as starting and stopping services, adding hosts to your cluster, and updating service configurations. You also can use Ambari Web to perform administrative tasks for your cluster, such as enabling Kerberos security and performing Stack upgrades. Any user can view Ambari Web features. Users with administrator-level roles can access more options than operator-level or view-only users can. For example, an Ambari administrator can manage cluster security, an operator user can monitor the cluster, but a view-only user can only access features to which an administrator grants required permissions.

More Information

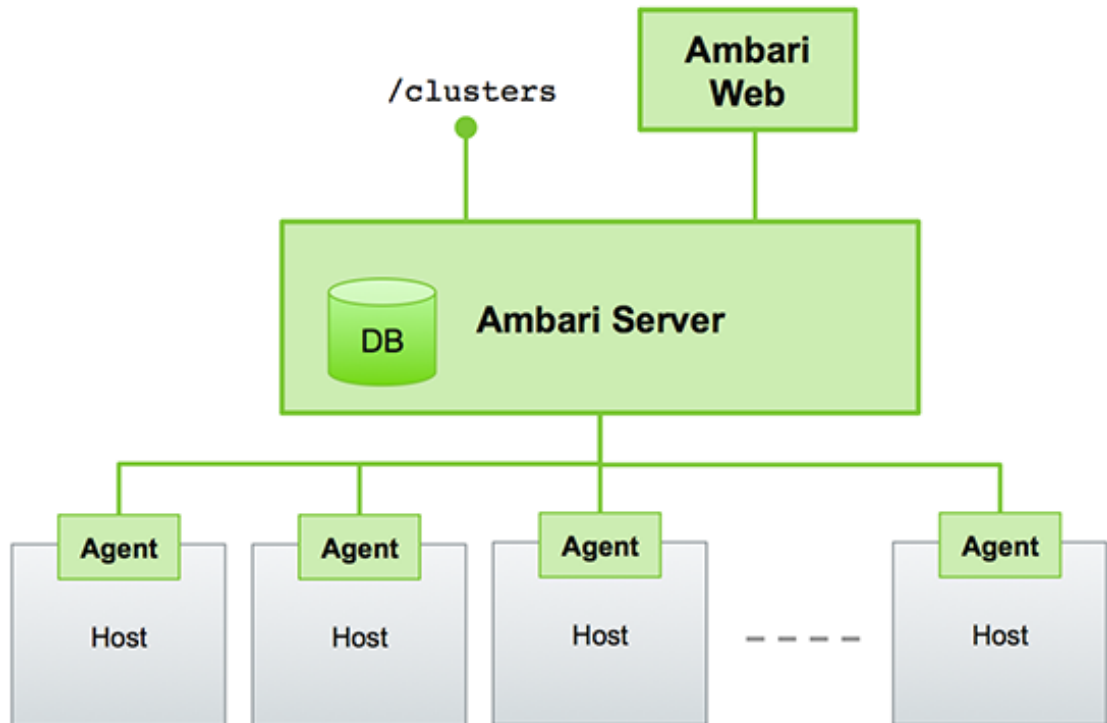
[Hortonworks Data Platform Apache Ambari Administration](#)

[Hortonworks Data Platform Apache Ambari Upgrade](#)

1.1. Ambari Architecture

The Ambari Server collects data from across your cluster. Each host has a copy of the Ambari Agent, which allows the Ambari Server to control each host.

The following graphic is a simplified representation of Ambari architecture:



Ambari Web is a client-side JavaScript application that calls the Ambari REST API (accessible from the Ambari Server) to access cluster information and perform cluster operations. After authenticating to Ambari Web, the application authenticates to the Ambari Server. Communication between the browser and server occurs asynchronously using the REST API.

The Ambari Web UI periodically accesses the Ambari REST API, which resets the session timeout. Therefore, by default, Ambari Web sessions do not timeout automatically. You can configure Ambari to timeout after a period of inactivity.

More Information

[Ambari Web Inactivity Timeout](#)

1.2. Accessing Ambari Web

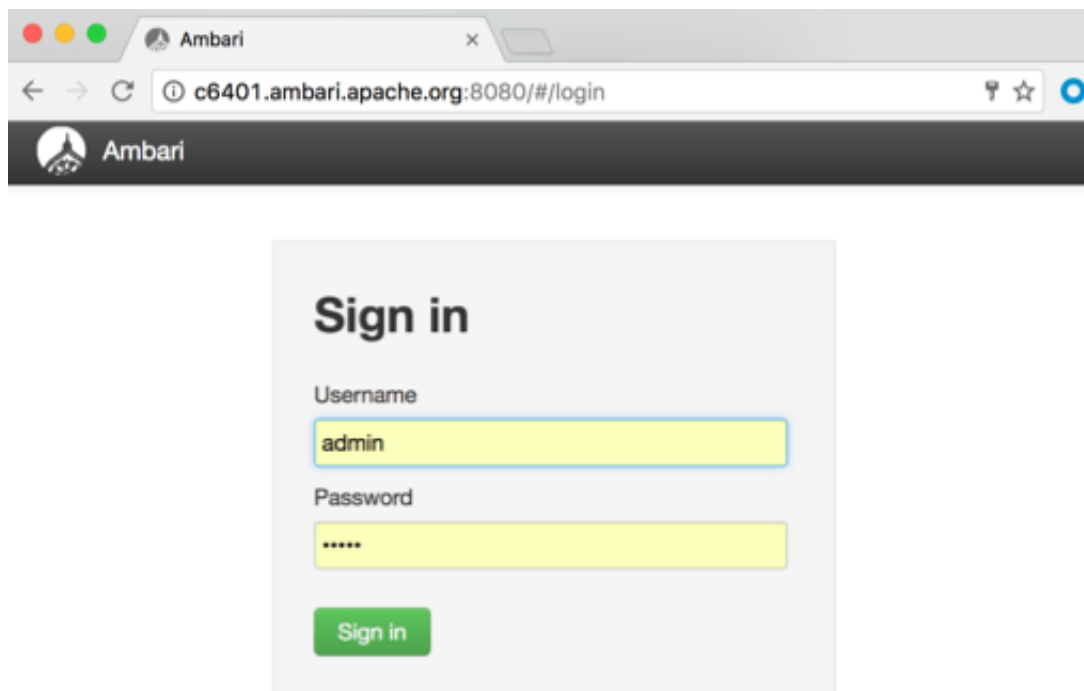
To access Ambari Web:

Steps

1. Open a supported browser.
2. Enter the Ambari Web URL:

`http://<your.ambari.server>:8080`

The Ambari Web login page displays in your browser.



3. Enter your user name and password.

If you are an Ambari administrator accessing the Ambari Web UI for the first time, use the default Ambari administrator account

```
admin/admin
```

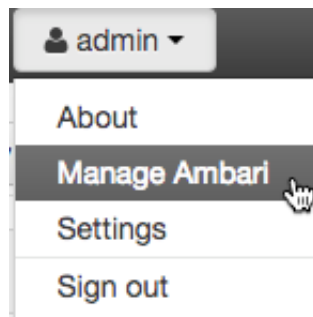
4. Click **Sign In**.

If Ambari Server is stopped, you can restart it using a command line editor on the Ambari Server host machine:

```
ambari-server start
```

Typically, you start the Ambari Server and Ambari Web as part of the installation process.

Ambari administrators access the Ambari Admin page from the **Manage Ambari** option in Ambari Web:



More Information

[Ambari Administration Overview](#)

[Hortonworks Data Platform Apache Ambari Installation](#)

2. Understanding the Cluster Dashboard

You monitor your Hadoop cluster using the Ambari Web Cluster dashboard. You access the Cluster dashboard by clicking **Dashboard** at the top of the Ambari Web UI *main window*:



More Information

- [Viewing the Cluster Dashboard \[5\]](#)
- [Modifying the Cluster Dashboard \[9\]](#)
- [Viewing Cluster Heatmaps \[11\]](#)

2.1. Viewing the Cluster Dashboard

Ambari Web UI displays the **Dashboard** page as the home page. Use **Dashboard** to view the operating status of your cluster.

The left side of Ambari Web displays the list of Hadoop services currently running in your cluster. **Dashboard** includes **Metrics**, **Heatmaps**, and **Config History** tabs; by default, the **Metrics** tab is displayed. On the **Metrics** page, multiple *widgets*, represent operating status information of *services* in your HDP cluster. Most widgets display a single metric: for example, **HDFS Disk Usage** represented by a load chart and a percentage figure:



Metrics Widgets and Descriptions

HDFS metrics

HDFS Disk Usage	The percentage of distributed file system (DFS) used, which is a combination of DFS and non-DFS used
Data Nodes Live	The number of DataNodes operating, as reported from the NameNode
NameNode Heap	The percentage of NameNode Java Virtual Machine (JVM) heap memory used
NameNode RPC	The average RPC queue latency
NameNode CPU WIO	The percentage of CPU wait I/O
NameNode Uptime	The NameNode uptime calculation

YARN metrics (HDP 2.1 or later stacks)

ResourceManager Heap	The percentage of ResourceManager JVM heap memory used
ResourceManager Uptime	The ResourceManager uptime calculation
NodeManagers Live	The number of DataNodes operating, as reported from the ResourceManager
YARN Memory	The percentage of available YARN memory (used versus total available)

HBase metrics

HBase Master Heap	The percentage of NameNode JVM heap memory used
HBase Ave Load	The average load on the HBase server
HBase Master Uptime	The HBase master uptime calculation
Region in Transition	The number of HBase regions in transition

Storm metrics (HDP 2.1 or later stacks)

Supervisors Live	The number of supervisors operating as reported by the Nimbus server
------------------	--

More Information

[Modifying the Service Dashboard \[30\]](#)

[Scanning Operating Status \[6\]](#)

2.1.1. Scanning Operating Status

The service summary list on the left side of Ambari Web lists all of the Apache component services that are currently monitored. The icon shape, color, and action to the left of each item indicates the operating status of that item:

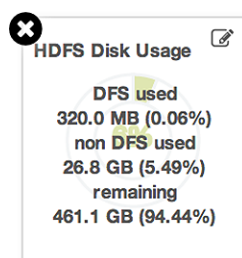
Status Indicators

Color	Status
solid green	All masters are running.
blinking green	Starting up
solid red	At least one master is down.
blinking red	Stopping

Click a service name to open the **Services** page, on which you can see more detailed information about that service.

2.1.2. Viewing Details from a Metrics Widget

To see more detailed information about a service, hover your cursor over a Metrics widget:



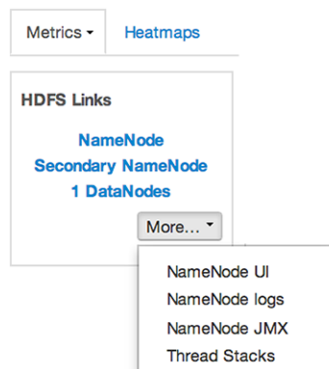
- To remove a widget, click the white X
- To edit the display of information in a widget, click the edit (pencil) icon.

More Information

[Customizing Metrics Display \[11\]](#)

2.1.3. Linking to Service UIs

The **HDFS Links** and **HBase Links** widgets list HDP components for which links to more metrics information, such as thread stacks, logs, and native component UIs, are available. For example, you can link to NameNode, Secondary NameNode, and DataNode components for HDFS by using the links shown in the following example:



Choose the **More** drop-down to select from the list of links available for each service. The Ambari Dashboard includes additional links to metrics for the following services:

HDFS

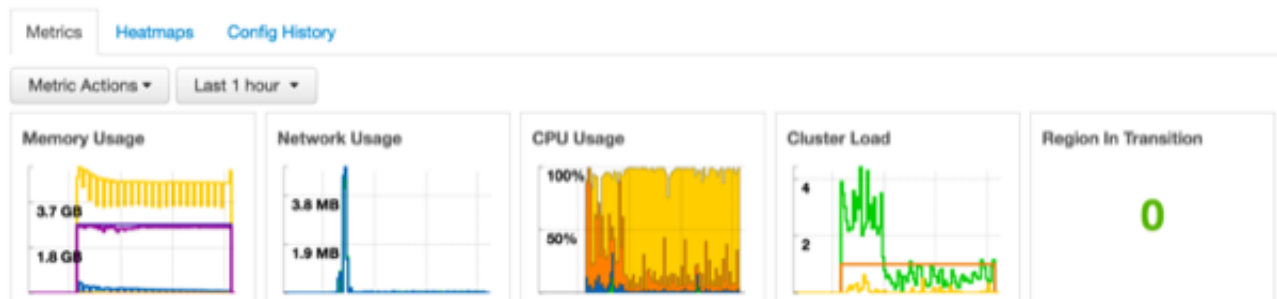
NameNode UI	Links to the NameNode UI
NameNode Logs	Links to the NameNode logs
NameNode JMX	Links to the NameNode JMX servlet
Thread Stacks	Links to the NameNode thread stack traces

HBase

HBase Master UI	Links to the HBase Master UI
HBase Logs	Links to the HBase logs
ZooKeeper Info	Links to ZooKeeper information
HBase Master JMX	Links to the HBase Master JMX servlet
Debug Dump	Links to debug information
Thread Stacks	Links to the HBase Master thread stack traces

2.1.4. Viewing Cluster-Wide Metrics

From the **Metrics** tab, you can also view the following cluster-wide metrics:



These metrics widgets show the following information:

Memory usage	Cluster-wide memory used, including memory that is cached, swapped, used, and shared
Network usage	The cluster-wide network utilization, including in-and-out
CPU Usage	Cluster-wide CPU information, including system, user and wait IO
Cluster Load	Cluster-wide Load information, including total number of nodes. total number of CPUs, number of running processes and 1-min Load

You can customize this display as follows:

- To remove a widget
Click the white X.
- To magnify the chart or itemize the widget display
Hover your cursor over the widget.
- To remove or add metrics
Select the item on the widget legend.
- To see a larger view of the chart
Select the magnifying glass icon.

Ambari displays a larger version of the widget in a separate window:



You can use the larger view in the same ways that you use the dashboard.

To close the larger view, click **OK**.

2.2. Modifying the Cluster Dashboard

You can modify the content of the Ambari Cluster dashboard in the following ways:

- [Replace a Removed Widget to the Dashboard \[10\]](#)

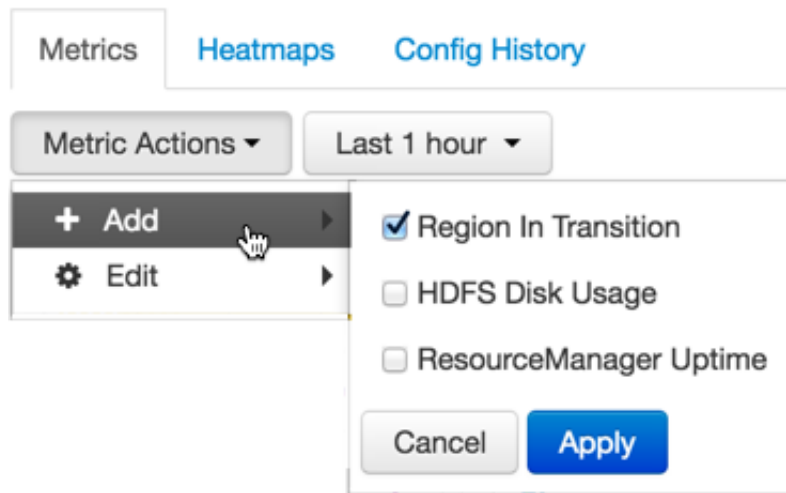
- [Reset the Dashboard \[10\]](#)
- [Customizing Metrics Display \[11\]](#)

2.2.1. Replace a Removed Widget to the Dashboard

To replace a widget that has been removed from the dashboard:

Steps

1. Select **Metric Actions**:



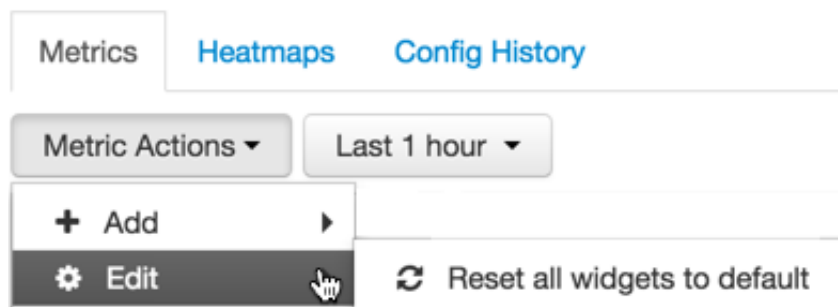
2. Click **Add**.
3. Select a metric, such as **Region in Transition**.
4. Click **Apply**.

2.2.2. Reset the Dashboard

To reset all widgets on the dashboard to display default settings:

Steps

1. Click **Metric Actions**:



2. Click **Edit**.

3. Click **Reset all widgets to default**.

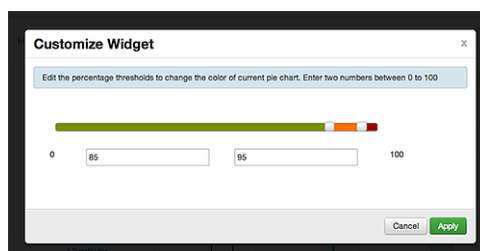
2.2.3. Customizing Metrics Display

Although not all widgets can be edited, you can customize the way that some of them display metrics by using the Edit (pencil) icon, if one is displayed.

Steps

1. Hover your cursor over a widget.
2. Click **Edit**.

The Customize Widget window appears:



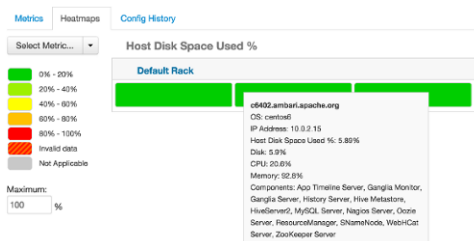
3. Follow the instructions in **Customize Widget** to customize widget appearance.

In this example, you can adjust the thresholds at which the **HDFS Capacity** bar chart changes color, from green to orange to red.

4. To save your changes and close the editor, click **Apply**.
5. To close the editor without saving any changes, choose **Cancel**.

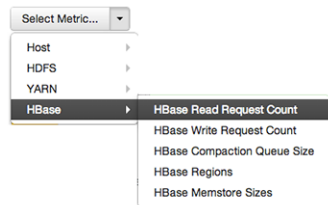
2.3. Viewing Cluster Heatmaps

As described earlier, the Ambari web interface home page is divided into a status summary panel on the left, and **Metrics**, **Heatmaps**, and **Config History** tabs at the top, with the **Metrics** page displayed by default. When you want to view a graphical representation of your overall cluster utilization, clicking **Heatmaps** provides you with that information, using simple color coding known as a *heatmap*:



A colored block represents each host in your cluster. You can see more information about a specific host by hovering over its block, which causes a separate window to display metrics about HDP components installed on that host.

Colors displayed in the block represent usage in a unit appropriate for the selected set of metrics. If any data necessary to determine usage is not available, the block displays **Invalid data**. You can solve this issue by changing the default maximum values for the heatmap, using the **Select Metric** menu:



Heatmaps supports the following metrics:

Host/Disk Space Used %	disk.disk_free and disk.disk_total
Host/Memory Used %	memory.mem_free and memory.mem_total
Host/CPU Wait I/O %	cpu.cpu_wio
HDFS/Bytes Read	dfs.datanode.bytes_read
HDFS/Bytes Written	dfs.datanode.bytes_written
HDFS/Garbage Collection Time	jvm.gcTimeMillis
HDFS/JVM Heap MemoryUsed	jvm.memHeapUsedM
YARN/Garbage Collection Time	jvm.gcTimeMillis
YARN / JVM Heap Memory Used	jvm.memHeapUsedM
YARN / Memory used %	UsedMemoryMB and AvailableMemoryMB
HBase/RegionServer read request count	hbase.regionserver.readRequestsCount
HBase/RegionServer write request count	hbase.regionserver.writeRequestsCount
HBase/RegionServer compaction queue size	hbase.regionserver.compactionQueueSize
HBase/RegionServer regions	hbase.regionserver.regions
HBase/RegionServer memstore sizes	hbase.regionserver.memstoreSizeMB

3. Managing Hosts

As a Cluster administrator or Cluster operator, you need to know the operating status of each hosts. Also, you need to know which hosts have issues that require action. You can use the Ambari Web **Hosts** page to manage multiple Hortonworks Data Platform (HDP) components, such as DataNodes, NameNodes, NodeManagers, and RegionServers, running on hosts throughout your cluster. For example, you can restart all DataNode components, optionally controlling that task with rolling restarts. Ambari Hosts enables you to filter your selection of host components to manage, based on operating status, host health, and defined host groupings.

The **Hosts** tab enables you to perform the following tasks:

- [Understanding Host Status \[13\]](#)
- [Searching the Hosts Page \[14\]](#)
- [Performing Host-Level Actions \[17\]](#)
- [Managing Components on a Host \[18\]](#)
- [Decommissioning a Master or Slave \[19\]](#)
- [Delete a Component \[20\]](#)
- [Deleting a Host from a Cluster \[21\]](#)
- [Setting Maintenance Mode \[21\]](#)
- [Add Hosts to a Cluster \[24\]](#)
- [Establishing Rack Awareness \[25\]](#)

3.1. Understanding Host Status

You can view the individual hosts in your cluster on the Ambari Web **Hosts** page. The hosts are listed by fully qualified domain name (FDQN) and accompanied by a colored icon that indicates the host's operating status:

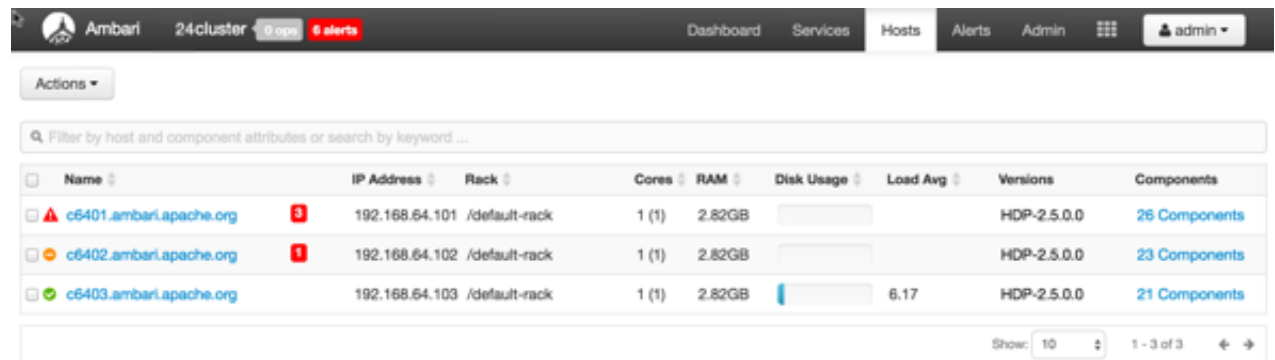
Red Triangle	At least one master component on that host is down. You can hover your cursor over the host name to see a tooltip that lists affected components.
Orange	Orange - At least one slave component on that host is down. Hover to see a tooltip that lists affected components.
Yellow	Ambari Server has not received a heartbeat from that host for more than 3 minutes.
Green	Normal running state.








Maintenance Mode Black "medical bag" icon indicates a host in maintenance mode.

Alert Red square with white number indicates the number of alerts generated on a host.

A red icon overrides an orange icon, which overrides a yellow icon. In other words, a host that has a master component down is accompanied by a red icon, even though it might have slave component or connection issues as well. Hosts in maintenance mode or are experiencing alerts, are accompanied by an icon to the right of the host name.

The following example **Hosts** page shows three hosts, one having a master component down, one having a slave component down, one running normally, and two with alerts:



Name	IP Address	Rack	Cores	RAM	Disk Usage	Load Avg	Versions	Components
  c6401.ambari.apache.org	192.168.64.101	/default-rack	1 (1)	2.82GB			HDP-2.5.0.0	26 Components
 c6402.ambari.apache.org	192.168.64.102	/default-rack	1 (1)	2.82GB			HDP-2.5.0.0	23 Components
 c6403.ambari.apache.org	192.168.64.103	/default-rack	1 (1)	2.82GB		6.17	HDP-2.5.0.0	21 Components

More Information

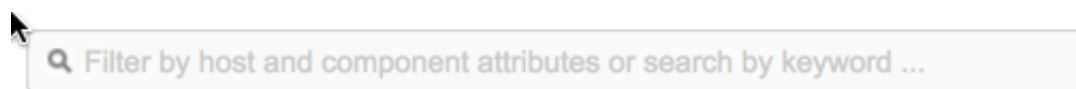
[Maintenance Mode](#)

[Alerts](#)

3.2. Searching the Hosts Page

You can search the full list of hosts, filtering your search by host name, component attribute, and component operating status. You can also search by keyword, simply by typing a word in the search box.

The Hosts search tool appears above the list of hosts:



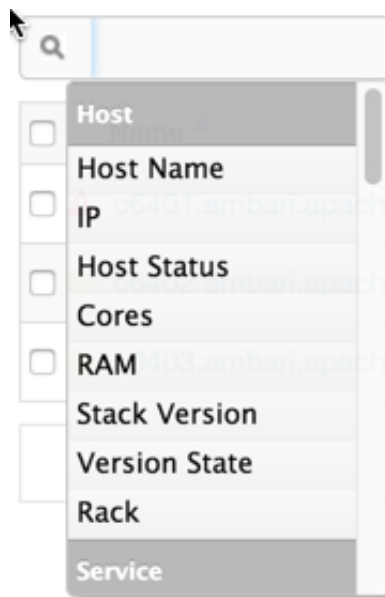
Steps

1. Click the search box.

Available search types appear, including:

Search by Host Attribute

Search by host name, IP, host status, and other attributes, including:

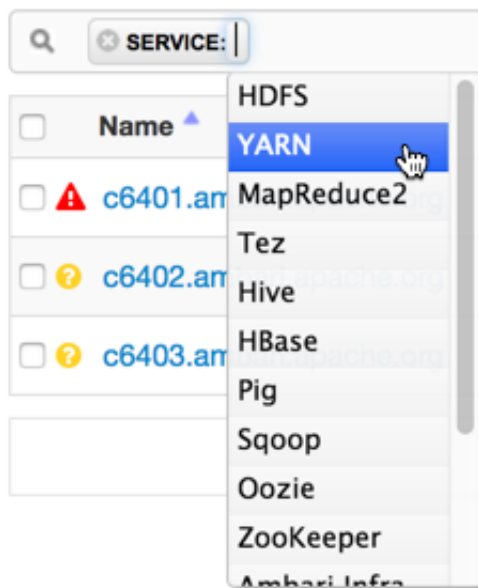


- | | |
|---------------------|---|
| Search by Service | Find hosts that are hosting a component from a given service. |
| Search by Component | Find hosts that are hosting a components in a given state, such as started, stopped, maintenance mode, and so on. |
| Search by keyword | Type any word that describes what you are looking for in the search box. This becomes a text filter. |

2. Click a Search type.

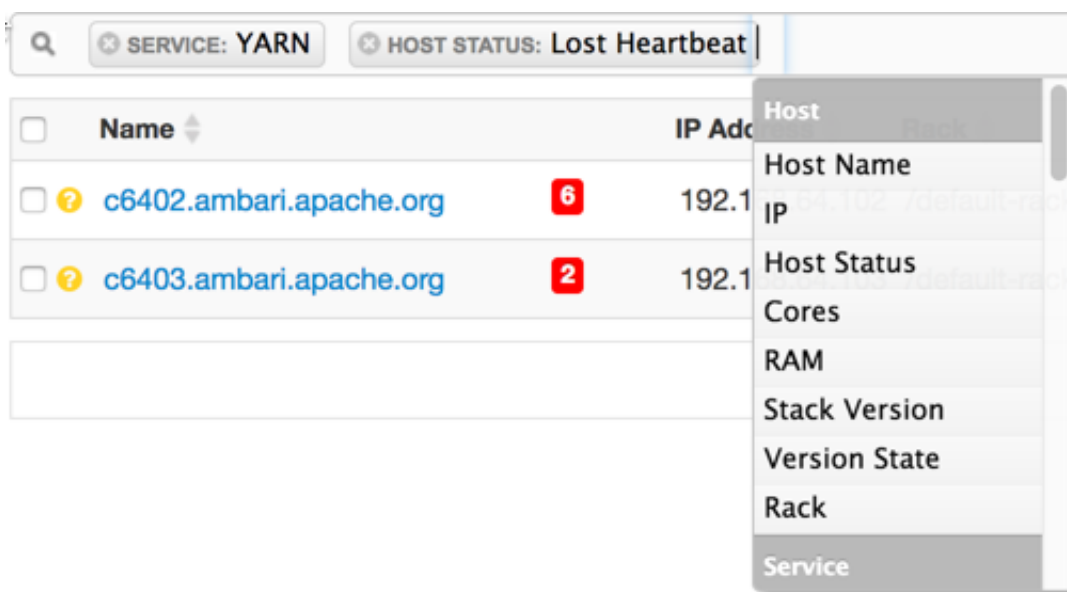
A list of available options appears, depending on your selection in step 1.

For example, if you click Service, current services appear:



3. Click an option, (in this example, the YARN service).

The list of hosts that match your current search criteria display on the Hosts page.



4. Click option(s) to further refine your search.

Examples of searches that you can perform, based on specific criteria, and which interface controls to use:

Find all hosts with a DataNode



Find all the hosts with a DataNode that are stopped



Find all the hosts with an HDFS component

✕ SERVICE: HDFS

Find all the hosts with an HDFS or HBase component

✕ SERVICE: HDFS

✕ SERVICE: HBase

3.3. Performing Host-Level Actions

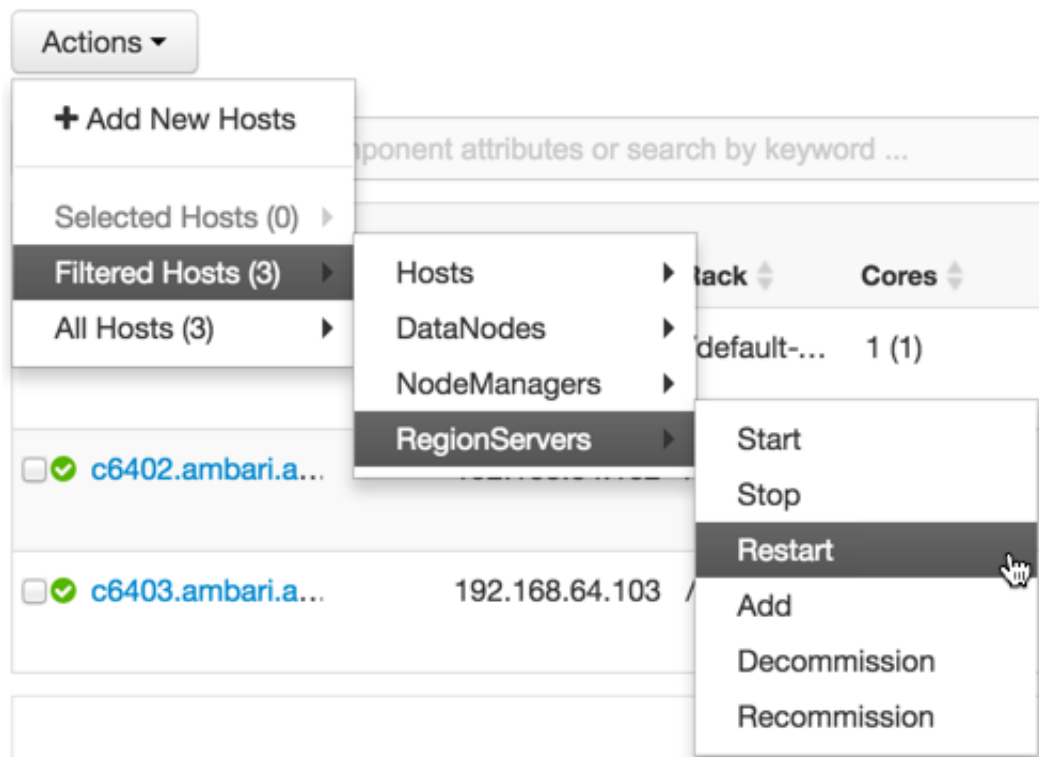
Use the **Actions** UI control to act on hosts in your cluster. Actions that you perform that comprise more than one operation, possibly on multiple hosts, are also known as *bulk operations*.

The **Actions** control comprises a workflow that uses a sequence of three menus to refine your search: a hosts menu, a menu of objects based on your host choice, and a menu of actions based on your object choice.

For example, if you want to restart the RegionServers on any host in your cluster on which a RegionServer exists:

Steps

1. In the **Hosts** page, select or search for hosts running a RegionServer:
2. Using the **Actions** control, click **Filtered Hosts > RegionServers > Restart**:



3. Click **OK** to start the selected operation.
4. Optionally, monitor background operations to follow, diagnose, or troubleshoot the restart operation.

More Information

[Monitoring Background Operations \[38\]](#)

3.4. Managing Components on a Host

To manage components running on a specific host, click one of the FQDNs listed on the **Hosts** page. For example, if you click `c6403.ambari.apache.org`, that host's page appears. Clicking the Summary tab displays a Components pane that lists all components installed on that host:

The screenshot displays the Ambari interface for the host `c6403.ambari.apache.org`. The interface is divided into several sections:

- Components:** A list of installed services with their status:
 - Metrics Collector / Ambari Metrics: Started
 - ZooKeeper Server / ZooKeeper: Started
 - DataNode / HDFS: Started
 - RegionServer / HBase: Started
 - SmartSense HST A... / SmartSense: Started
 - Log Feeder / Log Search: Started
 - Metrics Monitor / Ambari Metrics: Started
 - NodeManager / YARN: Started
 - Clients / HBase Client, HCat Client, HDFS Client, Hive Client, Log Search Solr Client, MapReduce2 Client, Oozie Client, Pig Client, Slider Client, Tez Client, YARN Client, ZooKeeper Client: Installed
- Host Metrics:** A section with six charts showing performance metrics over the last hour:
 - CPU Usage: 100% (orange bar chart)
 - Disk Usage: 372.5 GB / 196.2 GB (blue line chart)
 - Load: 4 / 2 (orange line chart)
 - Memory Usage: 3.7 GB / 1.6 GB (yellow line chart)
 - Network Usage: 78.1 KB / 58.6 KB / 19.5 KB (green bar chart)
 - Processes: 50 (blue line chart)
- Summary:** Host details including:
 - Hostname: c6403.ambari.apache.org
 - IP Address: 192.168.64.103
 - Rack: /default-rack
 - OS: centos6 (x86_64)
 - Cores [CPU]: 1 (1)
 - Disk: 34.5GB/489.66GB (7.05% used)
 - Memory: 2.82GB
 - Load Avg: 2.42
 - Heartbeat: 19 minutes ago
 - Current Version: 2.5.0.0-687

To manage all of the components on a single host, you can use the **Host Actions** control at the top right of the display to start, stop, restart, delete, or turn on maintenance mode for all components installed on the selected host.

Alternatively, you can manage components individually, by using the drop-down menu shown next to an individual component in the **Components** pane. Each component's menu is labeled with the component's current operating status. Opening the menu displays your available management options, based on that status: for example, you can decommission, restart, or stop the DataNode component for HDFS, as shown here:

The screenshot displays the Ambari Components page. At the top right, there is a '+ Add' button. The main area is a table of components, each with a green checkmark icon, a name, a link to its configuration page, and a status dropdown menu. The components listed are:

- Metrics Collector / Ambari Metrics (Started)
- ZooKeeper Server / ZooKeeper (Started)
- Accumulo TServer / Accumulo (Started)
- DataNode / HDFS (Started)
- Flume / Flume (Started)
- RegionServer / HBase (Started)
- SmartSense HST Agent / SmartSense (Started)
- Metrics Monitor / Ambari Metrics (Started)
- NodeManager / YARN (Started)
- Supervisor / Storm (Started)
- Clients / Accumulo Client, Falcon Client, HBase Client, HCat Client, HDFS Client, Hive Client, Mahout, MapReduce2 Client, Oozie Client, Pig, Slider, Spark Client, Sqoop, Tez Client, YARN Client, ZooKeeper Client (Installed)

A context menu is open over the 'Started' dropdown for the 'Metrics Collector' component, showing the following options: Decommission, Restart, Stop, Turn On Maintenance Mode, and Delete.

3.5. Decommissioning a Master or Slave

Decommissioning is a process that supports removing components and their hosts from the cluster. You must decommission a master or slave running on a host before removing it or its host from service. Decommissioning helps you to prevent potential loss of data or disruption of service. Decommissioning is available for the following component types:

- DataNodes
- NodeManagers
- RegionServers

Decommissioning executes the following tasks:

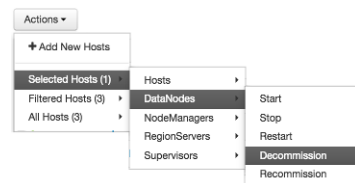
- | | |
|-------------------|---|
| For DataNodes | Safely replicates the HDFS data to other DataNodes in the cluster |
| For NodeManagers | Stops accepting new job requests from the masters and stops the component |
| For RegionServers | Turns on drain mode and stops the component |

3.5.1. Decommission a Component

To decommission a component (a DataNode, in the following example):

Steps

1. Using Ambari Web, browse the **Hosts** page.
2. Find and click the FQDN of the host on which the component resides.
3. Using the **Actions** control, click **Selected Hosts > DataNodes > Decommission**:



The UI shows Decommissioning status while in process:



When this DataNode decommissioning process is finished, the status display changes to Decommissioned (shown here for NodeManager).

3.6. Delete a Component

To delete a component:

Steps

1. Using Ambari Web, browse the **Hosts** page.
2. Find and click the FQDN of the host on which the component resides.
3. In **Components**, find a decommissioned component.
4. If the component status is Started, stop it.

A decommissioned slave component may restart in the decommissioned state.

5. Click **Delete** from the component drop-down menu.

Deleting a slave component, such as a DataNode does not automatically inform a master component, such as a NameNode to remove the slave component from its exclusion list. Adding a deleted slave component back into the cluster presents the following issue; the added slave remains decommissioned from the master's perspective. Restart the master component, as a work-around.

6. To enable Ambari to recognize and monitor only the remaining components, restart services.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

3.7. Deleting a Host from a Cluster

Deleting a host removes the host from the cluster.

Prerequisites

Before deleting a host, you must complete the following prerequisites:

- Stop all components running on the host.
- Decommission any DataNodes running on the host.
- Move from the host any master components, such as NameNode or ResourceManager, running on the host.
- Turn off host Maintenance Mode, if it is on.

To delete a component:

Steps

1. Using Ambari Web, browse the hosts page to find and click the FQDN of the host that you want to delete.
2. On the **Host-Details** page, click **Host Actions**.
3. Click Delete.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

3.8. Setting Maintenance Mode

Setting Maintenance Mode enables you to suppress alerts and omit bulk operations for specific services, components, and hosts in an Ambari-managed cluster when you want to

focus on performing hardware or software maintenance, changing configuration settings, troubleshooting, decommissioning, or removing cluster nodes.

Explicitly setting Maintenance Mode for a service implicitly sets Maintenance Mode for components and hosts that run the service. While Maintenance Mode prevents bulk operations being performed on the service, component, or host, you may explicitly start and stop a service, component, or host while in Maintenance Mode.

The following sections provide examples of how to use Maintenance Mode in a three-node, Ambari-managed cluster installed using default options and having one data node, on host c6403. They describe how to explicitly turn on Maintenance Mode for the HDFS service, alternative procedures for explicitly turning on Maintenance Mode for a host, and the implicit effects of turning on Maintenance Mode for a service, a component, and a host.

More Information

[Set Maintenance Mode for a Service \[22\]](#)

[Set Maintenance Mode for a Host \[22\]](#)

[When to Set Maintenance Mode \[23\]](#)

3.8.1. Set Maintenance Mode for a Service

1. Using Services, select **HDFS**.
2. Select Service Actions, then choose **Turn On Maintenance Mode**.
3. Choose **OK** to confirm.

Notice, on Services Summary that Maintenance Mode turns on for the NameNode and SNameNode components.

3.8.2. Set Maintenance Mode for a Host

To set Maintenance Mode for a host by using the **Host Actions** control:

Steps

1. Using Hosts, select `c6401.ambari.apache.org`.
2. Select **Host Actions**, then choose **Turn On Maintenance Mode**.
3. Choose **OK** to confirm.

Notice on Components, that Maintenance Mode turns on for all components.

To set Maintenance Mode for a host, using the **Actions** control:

Steps

1. Using Hosts, click `c6403.ambari.apache.org`.
2. In **Actions > Selected Hosts > Hosts**, choose **Turn On Maintenance Mode**.
3. Choose **OK**.

Your list of hosts shows that Maintenance Mode is set for hosts c6401 and c6403:

Name	IP Address	Cores (CPU)	RAM	Disk Usage	Load Avg	Components
Any		Any	Any		Any	Filter
c6401.ambari.apache.org	10.0.2.15	1 (1)	1.83GB		0.14	6 Components
c6402.ambari.apache.org	10.0.2.15	1 (1)	1.83GB		0.11	27 Components
c6403.ambari.apache.org	10.0.2.15	1 (1)	1.83GB		0.02	17 Components

3 of 3 hosts showing - clear filters 1 host selected - clear selection Show: 10 1 - 3 of 3

If you hover your cursor over each Maintenance Mode icon appearing in the hosts list, you see the following information:

- Hosts c6401 and c6403 are in Maintenance Mode.
- On host c6401, HBaseMaster, HDFS client, NameNode, and ZooKeeper Server are also in Maintenance Mode.
- On host c6403, 15 components are in Maintenance Mode.
- On host c6402, HDFS client and Secondary NameNode are in Maintenance Mode, even though the host is not.

Notice also how the DataNode is affected by setting Maintenance Mode on this host:

- Alerts are suppressed for the DataNode.
- DataNode is omitted from HDFS Start/Stop/Restart All, Rolling Restart.
- DataNode is omitted from all Bulk Operations except Turn Maintenance Mode ON/OFF.
- DataNode is omitted from Start All and / Stop All components.
- DataNode is omitted from a host-level restart/restart all/stop all/start.

3.8.3. When to Set Maintenance Mode

Four common instances in which you might want to set Maintenance Mode are to perform maintenance, to test a configuration change, to delete a service completely, and to address alerts.:

You want to perform hardware, firmware, or OS maintenance on a host.

While performing maintenance, you want to be able to do the following:

- Prevent alerts generated by all components on this host.
- Be able to stop, start, and restart each component on the host.
- Prevent host-level or service-level bulk operations from starting, stopping, or restarting components on this host.

You want to test a service configuration change. You will stop, start, and restart the service using a "rolling" restart to test whether restarting activates the change.

To achieve these goals, explicitly set Maintenance Mode for the host. Putting a host in Maintenance Mode implicitly puts all components on that host in Maintenance Mode.

To test configuration changes, you want to ensure the following conditions:

- No alerts are generated by any components in this service.
- No host-level or service-level bulk operations start, stop, or restart components in this service.

To achieve these goals, explicitly set Maintenance Mode for the service. Putting a service in Maintenance Mode implicitly turns on Maintenance Mode for all components in the service.

You want to stop a service.

To stop a service completely, you want to ensure the following conditions:

- No warnings are generated by the service.
- No components start, stop, or restart due to host-level actions or bulk operations.

To achieve these goals, explicitly set Maintenance Mode for the service. Putting a service in Maintenance Mode implicitly turns on Maintenance Mode for all components in the service.

You want to stop a host component from generating alerts.

To stop a host component from generating alerts, you must be able to do the following:

- Check the component.
- Assess warnings and alerts generated for the component.
- Prevent alerts generated by the component while you check its condition.

To achieve these goals, explicitly set Maintenance Mode for the host component. Putting a host component in Maintenance Mode prevents host-level and service-level bulk operations from starting or restarting the component. You can restart the component explicitly while Maintenance Mode is on.

3.9. Add Hosts to a Cluster

To add new hosts to your cluster:

Steps

1. Browse to the Hosts page and select **Actions > +Add New Hosts**.

The **Add Host** wizard provides a sequence of prompts similar to those in the Ambari Cluster Install wizard.

2. Follow the prompts, providing information similar to that provided to define the first set of hosts in your cluster:

Add Host Wizard

Next Steps

Review and confirm all recommended configuration changes.

Note that if you are adding a new host to your cluster, then you must know that the previous HDP and Ambari components are not added to the new host of your cluster.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

[Install Options](#)

3.10. Establishing Rack Awareness

You can establish rack awareness in two ways. Either you can set the rack ID using Ambari or you can set the rack ID using a custom topology script.

More Information

[Set the Rack ID Using Ambari \[26\]](#)

[Set the Rack ID Using a Custom Topology Script \[27\]](#)

3.10.1. Set the Rack ID Using Ambari

By setting the Rack ID, you can enable Ambari to manage rack information for hosts, including displaying the hosts in heatmaps by Rack ID, enabling users to filter and find hosts on the **Hosts** page by using that Rack ID.

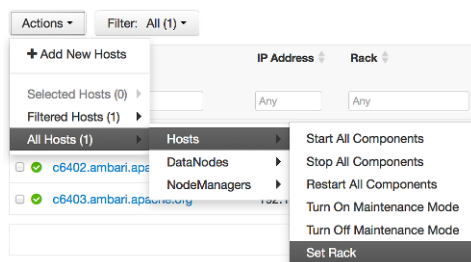
If HDFS is installed in your cluster, Ambari passes this Rack ID information to HDFS by using a topology script. Ambari generates a topology script at `/etc/hadoop/conf/topology.py` and sets the `net.topology.script.file.name` property in `core-site` automatically. This topology script reads a mappings file `/etc/hadoop/conf/topology_mappings.data` that Ambari automatically generates. When you make changes to Rack ID assignment in Ambari, this mappings file will be updated when you push out the HDFS configuration. HDFS uses this topology script to obtain Rack information about the DataNode hosts.

There are two ways using Ambari Web to set the Rack ID: for multiple hosts, using **Actions**, or for individual hosts, using **Host Actions**.

To set the Rack ID for multiple hosts:

Steps

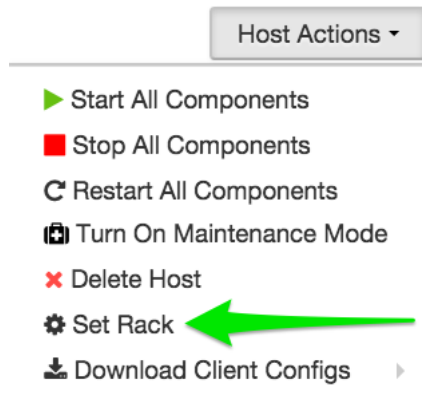
1. Using **Actions**, click selected, filtered, or all hosts.
2. Click **Hosts**.
3. Click **Set Rack**.



To set the Rack ID on an individual host:

Steps

1. Browse to the Host page.
2. Click **Host Actions**.
3. Click **Set Rack**.



3.10.2. Set the Rack ID Using a Custom Topology Script

If you do not want to have Ambari manage the rack information for hosts, you can use a custom topology script. To do this, you must *create your own topology script and manage distributing the script to all hosts*. Note also that because Ambari will have no access to host rack information, heatmaps will not display by rack in Ambari Web.

To set the Rack ID using a custom topology script:

Steps

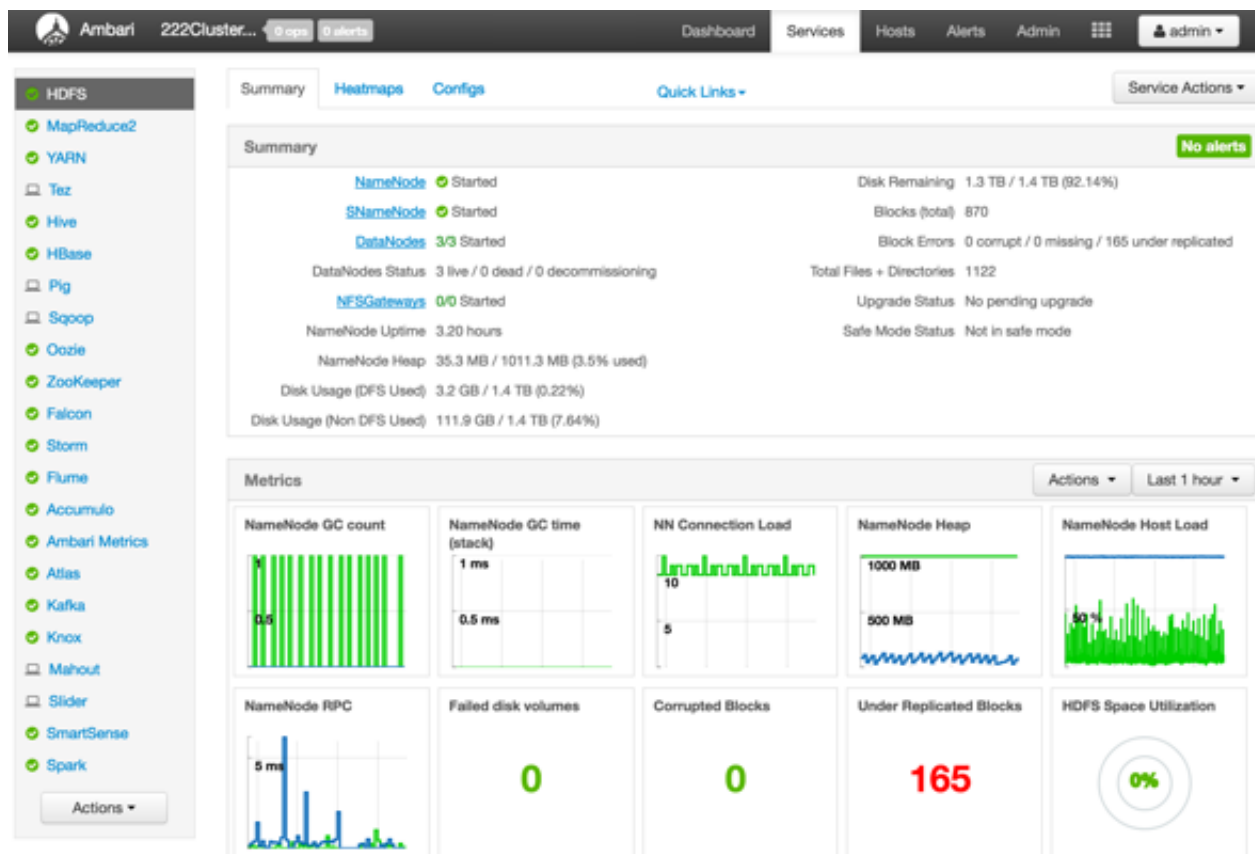
1. Browse to **Services > HDFS > Configs**.
2. Modify `net.topology.script.file.name` to your own custom topology script.
For example: `/etc/hadoop/conf/topology.sh`.
3. Distribute that topology script to your hosts.

You can now manage the rack mapping information for your script outside of Ambari.

4. Managing Services

You use the **Services** tab of the Ambari Web UI home page to monitor and manage selected services running in your Hadoop cluster.

All services installed in your cluster are listed in the leftmost panel:



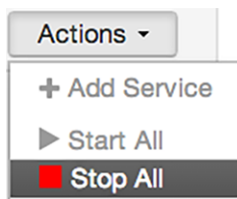
The **Services** tab enables you to perform the following tasks:

- [Starting and Stopping All Services \[29\]](#)
- [Displaying Service Operating Summary \[29\]](#)
- [Adding a Service \[32\]](#)
- [Changing Configuration Settings \[78\]](#)
- [Performing Service Actions \[36\]](#)
- [Rolling Restarts \[36\]](#)
- [Monitoring Background Operations \[38\]](#)
- [Removing A Service \[40\]](#)
- [Operations Audit \[40\]](#)

- [Using Quick Links \[40\]](#)
- [Refreshing YARN Capacity Scheduler \[41\]](#)
- [Managing HDFS \[41\]](#)
- [Managing Atlas in a Storm Environment \[43\]](#)

4.1. Starting and Stopping All Services

To start or stop all listed services simultaneously, click **Actions** and then click **Start All** or **Stop All**:



4.2. Displaying Service Operating Summary

Clicking the name of a service from the list displays a **Summary** tab containing basic information about the operational status of that service, including any alerts. To refresh the monitoring panels and display information about a different service, you can click a different name from the list.

Notice the colored icons next to each service name, indicating service operating status and any alerts generated for the service.

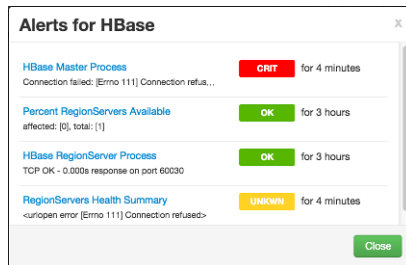
A screenshot of a 'Summary' tab for a service. The tab is titled 'Summary' and has a 'No alerts' indicator in the top right corner. The summary is organized into two columns. The left column lists service components and their status: NameNode (Started), SNameNode (Started), DataNodes (3/3 Started), NFSGateways (0/0 Started), NameNode Uptime (3.56 hours), NameNode Heap (111.1 MB / 1011.3 MB (11.0% used)), Disk Usage (DFS Used) (3.2 GB / 1.4 TB (0.22%)), and Disk Usage (Non DFS Used) (111.9 GB / 1.4 TB (7.64%)). The right column shows system metrics: Disk Remaining (1.3 TB / 1.4 TB (92.15%)), Blocks (total) (882), Block Errors (0 corrupt / 0 missing / 179 under replicated), Total Files + Directories (1134), Upgrade Status (No pending upgrade), and Safe Mode Status (Not in safe mode).

You can click one of the **View Host** links, as shown in the following example, to view components and the host on which the selected service is running:

[NameNode](#) ✔ Started
[SNameNode](#) ✔ Started
[DataNodes](#) 1/1 DataNodes Live

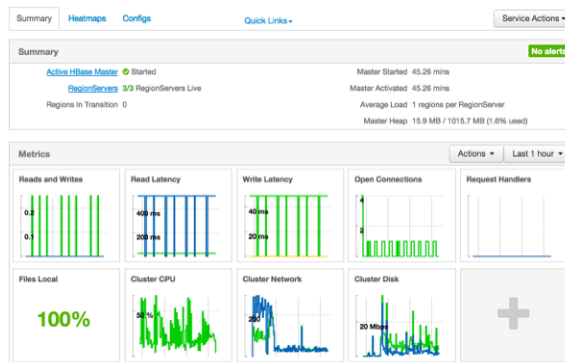
4.2.1. Alerts and Health Checks

In the **Summary** tab, you can click **Alerts** to see a list of all health checks and their status for the selected service. Critical alerts are shown first. To see alert definitions, you can click the text title of each alert message in the list to see the alert definition. The following example shows the results when you click **HBase > Services > Alerts > HBase Master Process**:



4.2.2. Modifying the Service Dashboard

Depending on the service, the **Summary** tab includes a Metrics dashboard that is by default populated with important service metrics to monitor:



If you have the Ambari Metrics service installed and are using Apache HDFS, Apache Hive, Apache HBase, or Apache YARN, you can customize the Metrics dashboard. You can add and remove widgets from the Metrics dashboard, and you can create new widgets and delete widgets. Widgets can be *private* to you and your dashboard, or they can be shared in a Widget Browser library.

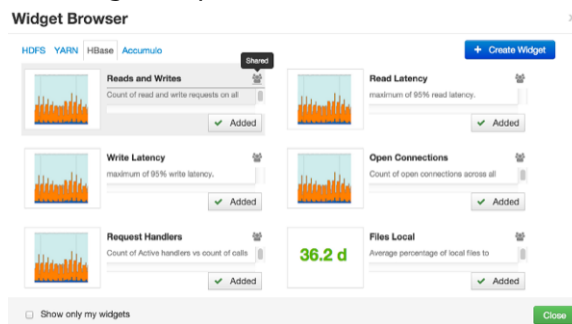
You must have the Ambari Metrics service installed to be able to view, create, and customize the Metrics dashboard.

4.2.2.1. Adding or Removing a Widget

To add or remove a widget in the HDFS, Hive, HBase, or YARN service Metrics dashboard:

1. Either click + to launch the Widget Browser, or click **Browse Widgets** from **Actions > Metrics**.
2. The Widget Browser displays the widgets available to add to your service dashboard, including widgets already included in your dashboard, shared widgets, and widgets

you have created. Widgets that are shared are identified by the icon highlighted in the following example:



3. If you want to display only the widgets you have created, select the “Show only my widgets” check box.
4. If you want to remove a widget shown as added to your dashboard, click to remove it.
5. If you want to add an available widget that is not already added, click **Add**.

4.2.2.2. Creating a Widget

1. Click + to launch the Widget Browser.
2. Either click the **Create Widget** button or **Create Widget** in the Actions menu Metrics header.
3. Select the type of widget to create.
4. Depending on the service and type of widget, you can select metrics and use operators to create an expression to be displayed in the widget.

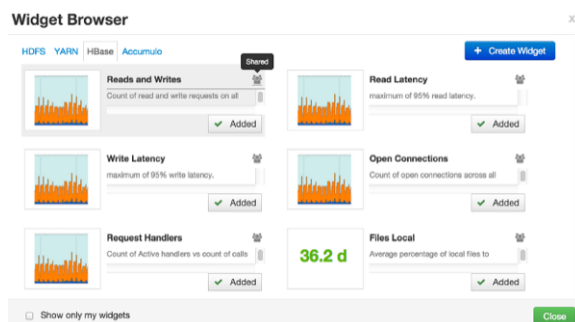
A preview of the widget is displayed as you build the expression.

5. Enter the widget name and description.
6. Optionally, choose to share the widget.

Sharing the widget makes the widget available to all Ambari users for this cluster. After a widget is shared, other Ambari Admins or Cluster Operators can modify or delete the widget. This cannot be undone.

4.2.2.3. Deleting a Widget

1. Click on the “+” to launch the Widget Browser. Alternatively, you can choose the Actions menu in the Metrics header to **Browse Widgets**.
2. The Widget Browser displays the available widgets to add to your Service Dashboard. This is a combination of shared widgets and widgets you have created. Widgets that are shared are identified by the icon highlighted in the following example.



3. If a widget is already added to your dashboard, it is shown as **Added**. Click to remove.
4. For widgets that you created, you can select the **More...** option to delete.
5. For widgets that are shared, if you are an Ambari Admin or Cluster Operator, you will also have the option to delete.

Deleting a shared widget removes the widget from all users. This cannot be undone.

4.2.2.4. Export Widget Graph Data

You can export the metrics data from widget graphs using the Export capability.

1. Hover your cursor over the widget graph, or click the graph to zoom, to display the **Export** icon.
2. Click the icon and specify either CSV or JSON format.

4.2.2.5. Setting Display Timezone

You can set the timezone used for displaying metrics data in widget graphs.

1. In Ambari Web, click your user name and select **Settings**.
2. In the **Locale** section, select the **Timezone**.
3. Click **Save**.

The Ambari Web UI reloads and graphs are displayed using the timezone you have set.

4.3. Adding a Service

The Ambari installation wizard installs all available Hadoop services by default. You can choose to deploy only some services initially, and then add other services as you need them. For example, many customers deploy only core Hadoop services initially. The **Add Service** option of the **Actions** control enables you to deploy additional services without interrupting operations in your Hadoop cluster. When you have deployed all available services, the **Add Service** control display is dimmed, indicating that it is unavailable.

To add a service, follow the steps shown in this example of adding the Apache Falcon service to your Hadoop cluster:

1. Click **Actions > Add Service**.

The Add Service wizard opens.

2. Click **Choose Services**.

The Choose Services pane displays, showing a table of those services that are already active in a green background and with their checkboxes checked.

Choose Services

Choose which services you want to install on your cluster.

<input type="checkbox"/> Service	Version	Description
<input checked="" type="checkbox"/> HDFS	2.7.3	Apache Hadoop Distributed File System
<input checked="" type="checkbox"/> YARN + MapReduce2	2.7.3	Apache Hadoop NextGen MapReduce (YARN)
<input checked="" type="checkbox"/> Tez	0.7.0	Tez is the next generation Hadoop Query Processing framework written on top of YARN.
<input checked="" type="checkbox"/> Hive	1.2.1000	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service
<input checked="" type="checkbox"/> HBase	1.1.2	A Non-relational distributed database, plus Phoenix, a high performance SQL layer for low latency applications.
<input checked="" type="checkbox"/> Pig	0.16.0	Scripting platform for analyzing large datasets
<input checked="" type="checkbox"/> Sqoop	1.4.6	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases
<input checked="" type="checkbox"/> Oozie	4.2.0	System for workflow coordination and execution of Apache Hadoop jobs. This also includes the installation of the optional Oozie Web Console which relies on and will install the ExUS Library .
<input checked="" type="checkbox"/> ZooKeeper	3.4.6	Centralized service which provides highly reliable distributed coordination
<input checked="" type="checkbox"/> Falcon	0.10.0	Data management and processing platform
<input checked="" type="checkbox"/> Storm	1.1.0	Apache Hadoop Stream processing framework
<input checked="" type="checkbox"/> Flume	1.5.2	A distributed service for collecting, aggregating, and moving large amounts of streaming data into HDFS
<input checked="" type="checkbox"/> Accumulo	1.7.0	Robust, scalable, high performance distributed key/value store.
<input checked="" type="checkbox"/> Ambari Infra	0.1.0	Core shared service used by Ambari managed components.
<input checked="" type="checkbox"/> Ambari Metrics	0.1.0	A system for metrics collection that provides storage and retrieval capability for metrics collected from the cluster
<input checked="" type="checkbox"/> Atlas	0.8.0	Atlas Metadata and Governance platform
<input checked="" type="checkbox"/> Kafka	0.10.0	A high-throughput distributed messaging system
<input checked="" type="checkbox"/> Knox	0.11.0	Provides a single point of authentication and access for Apache Hadoop services in a cluster
<input type="checkbox"/> Log Search	0.5.0	Log aggregation, analysis, and visualization for Ambari managed services. This service is Technical Preview .
<input checked="" type="checkbox"/> SmartSense	1.4.0-1033	SmartSense - Hortonworks SmartSense Tool (HST) helps quickly gather configuration, metrics, logs from common HDP services that aids to quickly troubleshoot support cases and receive cluster-specific recommendations.
<input checked="" type="checkbox"/> Spark	1.6.3	Apache Spark is a fast and general engine for large-scale data processing.
<input checked="" type="checkbox"/> Spark2	2.1.0	Apache Spark is a fast and general engine for large-scale data processing
<input checked="" type="checkbox"/> Zeppelin Notebook	0.7.0	A web-based notebook that enables interactive data analytics. It enables you to make beautiful data-driven, interactive and collaborative documents with SQL, Scala and more.
<input type="checkbox"/> Druid	0.9.2	A fast column-oriented distributed data store. This service is Technical Preview .
<input checked="" type="checkbox"/> Mahout	0.9.0	Project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification
<input checked="" type="checkbox"/> Slider	0.91.0	A framework for deploying, managing and monitoring existing distributed applications on YARN.

3. In the Choose Services pane, select the empty check box next to the service that you want to add, and then click Next.

Notice that you can also select all services listed by selecting the checkbox next to the Service table column heading.

4. In **Assign Masters**, confirm the default host assignment.

The Add Services Wizard indicates hosts on which the master components for a chosen service will be installed. A service chosen for addition shows a grey check mark.

Alternatively, use the drop-down menu to choose a different host machine to which master components for your selected service will be added.

The screenshot shows the 'Assign Masters' step of the 'ADD SERVICE WIZARD'. On the left, a sidebar contains navigation options: 'Choose Services', 'Assign Slaves and Clients', 'Customize Services', 'Review', 'Install, Start and Test', and 'Summary'. The main area is titled 'Assign Masters' and includes instructions: 'Assign master components to hosts you want to run them on. HiveServer2, Hive Metastore, and WebCat Server will be hosted on the same server.' Below this, a list of services is shown with their assigned master hosts. A dropdown menu is open, displaying a list of available hosts: 'c6402.ambar.apache.org (1.8 GB, 1 cores)', 'c6401.ambar.apache.org (1.8 GB, 1 cores)', 'c6403.ambar.apache.org (1.8 GB, 1 cores)', 'HBase Master', 'NameNode', 'SecondaryNameNode', 'Hadoop Distributed Cache (HDC) Server', 'Hive Metastore', 'WebCat Server', 'HiveServer2', 'Oozie Server', 'ZooKeeper', and 'Falcon Server'. The 'Falcon Server' is highlighted in green. At the bottom, there are 'Back' and 'Next' buttons.

5. If you are adding a service that requires slaves and clients, in the **Assign Slaves and Clients** control, accept the default assignment of slave and client components to hosts by clicking **Next**.

Alternatively, select hosts on which you want to install slave and client components (at least one host for the slave of each service being added), and click **Next**.

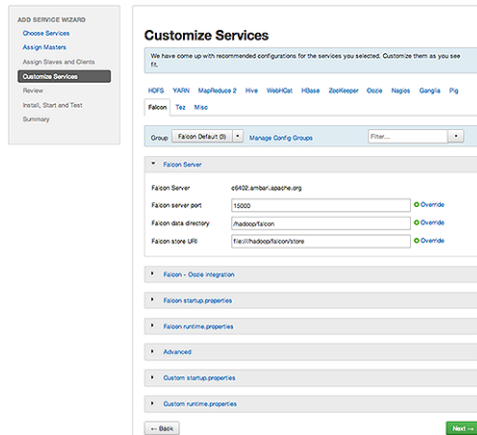
Host Roles Required for Added Services

Service Added	Host Role Required
YARN	NodeManager
HBase	RegionServer

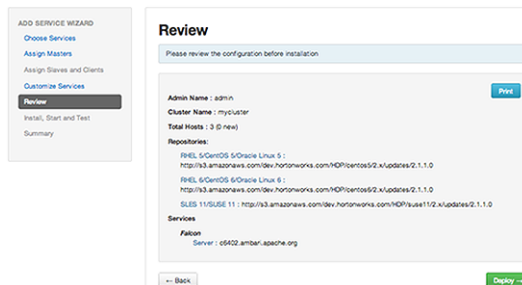
The screenshot shows the 'Assign Slaves and Clients' step of the 'ADD SERVICE WIZARD'. The main area is titled 'Assign Slaves and Clients' and includes instructions: 'Assign slave and client components to hosts you want to run them on. Hosts that are assigned master components are shown with a 'Client' will install HDFS Client, MapReduce 2 Client, YARN Client, Tez Client, Hive Client, HCat Client, HBase Client, Pig, Sqoop, Oozie Client, ZooKeeper Client and Falcon Client.' Below this, a table is shown with columns for 'Host' and 'Role'. The roles are assigned to the hosts. The 'Falcon Server' is highlighted in green. At the bottom, there are 'Back' and 'Next' buttons.

6. In **Customize Services**, accept the default configuration properties.

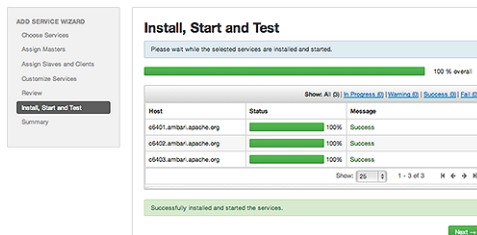
Alternatively, edit the default values for configuration properties, if necessary. Choose **Override** to create a configuration group for this service. Then, choose **Next**:



7. In **Review**, verify that the configuration settings match your intentions, and then, click **Deploy**:



8. Monitor the progress of installing, starting, and testing the service, and when that finishes successfully, click **Next**:



9. When you see the summary display of installation results, click **Complete**:



10. Review and confirm recommended configuration changes.

11. Restart any other components that have stale configurations as a result of adding services.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

[Choose Services](#)

[Apache Spark Component Guide](#)

[Apache Storm Component Guide](#)

[Apache Ambari Apache Storm Kerberos Configuration](#)

[Apache Kafka Component Guide](#)

[Apache Ambari Apache Kafka Kerberos Configuration](#)

[Installing and Configuring Apache Atlas](#)

[Installing Ranger Using Ambari](#)

[Installing Hue](#)

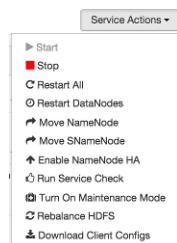
[Apache Solr Search Installation](#)

[Installing Ambari Log Search \(Technical Preview\)](#)

[Installing Druid \(Technical Preview\)](#)

4.4. Performing Service Actions

Manage a selected service on your cluster by performing service actions. In the **Services** tab, click **Service Actions** and click an option. Available options depend on the service you have selected; for example, HDFS service action options include:



Clicking **Turn On Maintenance Mode** suppresses alerts and status indicator changes generated by the service, while allowing you to start, stop, restart, move, or perform maintenance tasks on the service.

More Information

[Setting Maintenance Mode \[21\]](#)

[Enable Service Auto-Start \[99\]](#)

4.5. Rolling Restarts

When you restart multiple services, components, or hosts, use *rolling restarts* to distribute the task. A rolling restart stops and then starts multiple running slave components, such as DataNodes, NodeManagers, RegionServers, or Supervisors, using a batch sequence.



Important

Rolling restarts of DataNodes should be performed only during cluster maintenance.

You set rolling restart parameter values to control the number, time between, tolerance for failures, and limits for restarts of many components across large clusters.

To run a rolling restart, follow these steps:

1. From the service summary pane on the left of the Service display, click a service name.
2. On the service Summary page, click a link, such as DataNodes or RegionServers, of any components that you want to restart.

The Hosts page lists any host names in your cluster on which that component resides.

3. Using the host-level **Actions** menu, click the name of a slave component option, and then click **Restart**.
4. Review and set values for **Rolling Restart Parameters**.
5. Optionally, reset the flag to restart only components with changed configurations.
6. Click **Trigger Restart**.

After triggering the restart, you should monitor the progress of the background operations.

More Information

[Setting Rolling Restart Parameters \[37\]](#)

[Monitoring Background Operations \[38\]](#)

[Performing Host-Level Actions \[17\]](#)

[Aborting a Rolling Restart \[38\]](#)

4.5.1. Setting Rolling Restart Parameters

When you choose to restart slave components, you should use parameters to control how restarts of components roll. Parameter values based on ten percent of the total number of components in your cluster are set as default values. For example, default settings for a rolling restart of components in a three-node cluster restarts one component at a time, waits two minutes between restarts, proceeds if only one failure occurs, and restarts all existing components that run this service. Enter integer, non-zero values for all parameters.

Batch Size	Number of components to include in each restart batch.
Wait Time	Time (in seconds) to wait between queuing each batch of components.
Tolerate up to x failures	Total number of restart failures to tolerate, across all batches, before halting the restarts and not queuing batches.

If you trigger a rolling restart of components, the default value of **Restart components with stale configs** is “true.” If you trigger a rolling restart of services, this value is “false.”

Restart DataNodes x

This will restart a specified number of DataNodes at a time.

Note: This will trigger alerts. To suppress alerts, turn on Maintenance Mode for HDFS prior to triggering a rolling restart

Restart DataNodes at a time

Wait seconds between batches

Tolerate up to restart failures

Only restart DataNodes with stale configs

More Information

[Rolling Restarts \[36\]](#)

4.5.2. Aborting a Rolling Restart

To abort future restart operations in the batch, click **Abort Rolling Restart**:

Rolling Restart of NodeManagers - batch 1 of 1 x

← Operations Hosts Show: All (1)

Future operations of this batch request can be aborted.

✓ 65403.ambari.apache.org ▶

Do not show this dialog again when starting a background operation

More Information

[Rolling Restarts \[36\]](#)

4.6. Monitoring Background Operations

You can use the Background Operations window to monitor progress and completion of a task that comprises multiple operations, such as a rolling restart of components. The Background Operations window opens by default when you run such a task. For example, to monitor the progress of a rolling restart, click elements in the Background Operations window:

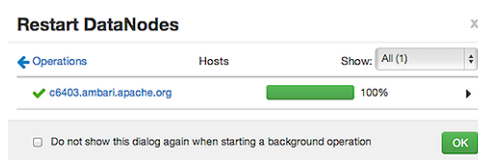
1. Click the right-arrow for each operation to show restart operation progress on each host:

1 Background Operations Running x

Operations	Start Time	Duration	Show: All (10)	
Rolling Restart of DataNodes - batch 1 of 1	Today 16:45	11.12 secs	<input type="text" value="35%"/>	▶
Rolling Restart of NodeManagers - batch 2 of 2	Today 10:06	38.45 secs	<input type="text" value="100%"/>	▶
Rolling Restart of NodeManagers - batch 1 of 2	Today 10:05	25.87 secs	<input type="text" value="100%"/>	▶

Do not show this dialog again when starting a background operation

- After restart operations are complete, you can click either the right-arrow or host name to view log files and any error messages generated on the selected host:



- Optionally, you can use the **Copy**, **Open**, or **Host Logs** icons located at the upper-right of the Background Operations window to copy, open, or view logs for the rolling restart.

For example, choose **Host Logs** to view error and output logs information for host c6403.ambari.apache.org:

c6403.ambari.apache.org

← Tasks ✓ Restart DataNode 📄 Copy 📄 Open 📄 Host Logs

Ambari stdout/stderr hadoop-hdfs-datanode-c6403.ambari.apache.org.log

stderr: /var/lib/ambari-agent/data/errors-127.txt

None

stdout: /var/lib/ambari-agent/data/output-127.txt

```

2016-06-09 19:53:07,171 - The hadoop conf dir /usr/hdp/current/hadoop-client/conf exists,
will call conf-select on it for version 2.5.0.0-687
2016-06-09 19:53:07,172 - Checking if need to create versioned conf dir /etc/hadoop/2.5.0.0-
687/0
2016-06-09 19:53:07,172 - call[({'ambari-python-wrap', '/usr/bin/conf-select', 'create-conf-
dir', '--package', 'hadoop', '--stack-version', '2.5.0.0-687', '--conf-version', '0'})]
{'logoutput': False, 'sudo': True, 'quiet': False, 'stderr': -1}
2016-06-09 19:53:07,249 - call returned (1, '/etc/hadoop/2.5.0.0-687/0 exist already', '')
2016-06-09 19:53:07,250 - checked_call[({'ambari-python-wrap', '/usr/bin/conf-select', 'set-
conf-dir', '--package', 'hadoop', '--stack-version', '2.5.0.0-687', '--conf-version', '0'})]
{'logoutput': False, 'sudo': True, 'quiet': False}
2016-06-09 19:53:07,291 - checked_call returned (0, '')
2016-06-09 19:53:07,299 - Ensuring that hadoop has the correct symlink structure
2016-06-09 19:53:07,300 - Using hadoop conf dir: /usr/hdp/current/hadoop-client/conf
2016-06-09 19:53:07,492 - The hadoop conf dir /usr/hdp/current/hadoop-client/conf exists,
will call conf-select on it for version 2.5.0.0-687
2016-06-09 19:53:07,493 - Checking if need to create versioned conf dir /etc/hadoop/2.5.0.0-
687/0
2016-06-09 19:53:07,493 - call[({'ambari-python-wrap', '/usr/bin/conf-select', 'create-conf-
dir', '--package', 'hadoop', '--stack-version', '2.5.0.0-687', '--conf-version', '0'})]
{'logoutput': False, 'sudo': True, 'quiet': False, 'stderr': -1}
2016-06-09 19:53:07,536 - call returned (1, '/etc/hadoop/2.5.0.0-687/0 exist already', '')
2016-06-09 19:53:07,536 - checked_call[({'ambari-python-wrap', '/usr/bin/conf-select', 'set-
conf-dir', '--package', 'hadoop', '--stack-version', '2.5.0.0-687', '--conf-version', '0'})]
{'logoutput': False, 'sudo': True, 'quiet': False}
2016-06-09 19:53:07,562 - checked_call returned (0, '')

```

Do not show this dialog again when starting a background operation OK

As shown here, you can also select the check box at the bottom of the Background Operations window to hide the window when performing tasks in the future.

4.7. Removing A Service



Important

Removing a service is not reversible and all configuration history will be lost.

To remove a service:

1. Click the name of the service from the left panes of the **Services** tab.
2. Click **Service Actions > Delete**.
3. As prompted, remove any dependent services.
4. As prompted, stop all components for the service.
5. Confirm the removal.

After the service is stopped, you **must confirm** the removal to proceed.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

4.8. Operations Audit

When you perform an operation using Ambari, such as user login or logout, stopping or starting a service, and adding or removing a service, Ambari creates an entry in an audit log. By reading the audit log, you can determine who performed the operation, when the operation occurred, and other, operation-specific information. You can find the Ambari audit log on your Ambari server host, at:

```
/var/log/ambari-server/ambari-audit.log
```

When you change configuration for a service, Ambari creates an entry in the audit log, and creates a specific log file, at:

```
ambari-config-changes.log
```

By reading the configuration change log, you can find out even more information about each change. For example:

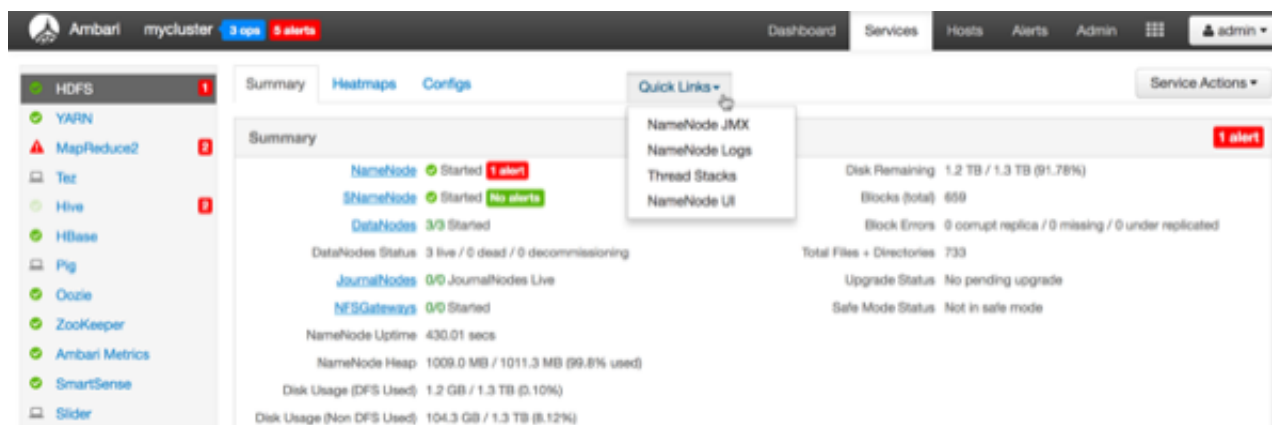
```
2016-05-25 18:31:26,242 INFO - Cluster 'MyCluster' changed by: 'admin';  
service_name='HDFS' config_group='default' config_group_id='-1' version='2'
```

More Information

[Changing Configuration Settings](#)

4.9. Using Quick Links

Select **Quick Links** options to access additional sources of information about a selected service. For example, HDFS Quick Links options include the following:



Quick Links are not available for every service.

4.10. Refreshing YARN Capacity Scheduler

This topic describes how to “refresh” the Capacity Scheduler from Ambari when you add or modify existing queues. After you modify the Capacity Scheduler configuration, YARN enables you to refresh the queues without restarting your ResourceManager, if you have made no destructive changes (such as completely removing a queue) to your configuration. The Refresh operation will fail with the following message: `Failed to re-init queues` if you attempt to refresh queues in a case where you performed a destructive change, such as removing a queue. In cases where you have made destructive changes, you must perform a ResourceManager restart for the capacity scheduler change to take effect.

To refresh the Capacity Scheduler, follow these steps:

1. In Ambari Web, browse to **Services > YARN > Summary**.
2. Click **Service Actions**, and then click **Refresh YARN Capacity Scheduler**.
3. Confirm that you want to perform this operation.

The refresh operation is submitted to the YARN ResourceManager.

More Information

[ResourceManager High Availability \[64\]](#)

4.11. Managing HDFS

This section contains information specific to rebalancing and tuning garbage collection in Hadoop Distributed File System (HDFS).

More Information

[Rebalancing HDFS \[42\]](#)

[Tuning Garbage Collection \[42\]](#)

[Customizing the HDFS Home Directory \[43\]](#)

[NameNode High Availability \[44\]](#)

4.11.1. Rebalancing HDFS

HDFS provides a “balancer” utility to help balance the blocks across DataNodes in the cluster. To initiate a balancing process, follow these steps:

1. In Ambari Web, browse to **Services > HDFS > Summary**.
2. Click **Service Actions**, and then click **Rebalance HDFS**.
3. Enter the **Balance Threshold** value as a percentage of disk capacity.
4. Click **Start**.

You can monitor or cancel a rebalance process by opening the **Background Operations** window in Ambari.

More Information

[Monitoring Background Operations \[38\]](#)

[Tuning Garbage Collection \[42\]](#)

4.11.2. Tuning Garbage Collection

The Concurrent Mark Sweep (CMS) garbage collection (GC) process includes a set of heuristic rules used to trigger garbage collection. This makes garbage collection less predictable and tends to delay collection until capacity is reached, creating a `Full GC` error (which might pause all processes).

Ambari sets default parameter values for many properties during cluster deployment. Within the `export HADOOP_NameNode_Opts=` clause of the `hadoop-env` template, two parameters that affect the CMS GC process have the following default settings:

- `-XX:+UseCMSInitiatingOccupancyOnly`

prevents the use of GC heuristics.

- `-XX:CMSInitiatingOccupancyFraction=<percent>`

tells the Java VM when the CMS collector should be triggered.

If this percent is set too low, the CMS collector runs too often; if it is set too high, the CMS collector is triggered too late, and [concurrent mode failure](#) might occur. The default setting for `-XX:CMSInitiatingOccupancyFraction` is `70`, which means that the application should utilize less than 70% of capacity.

To tune garbage collection by modifying the NameNode CMS GC parameters, follow these steps:

1. In Ambari Web, browse to **Services > HDFS**.

2. Open the **Configs** tab and browse to **Advanced > Advanced hadoop-env**.
3. Edit the `hadoop-env` template.
4. Save your configurations and restart, as prompted.

More Information

[Rebalancing HDFS \[42\]](#)

4.11.3. Customizing the HDFS Home Directory

By default, the HDFS home directory is set to `/user/<user_name>`. You can use the `dfs.user.home.base.dir` property to customize the HDFS home directory.

1. In Ambari Web, browse to **Services > HDFS > Configs > Advanced**.
2. Click **Custom hdfs-site**, then click **Add Property**.
3. On the Add Property pop-up, add the following property:

```
dfs.user.home.base.dir=<home_directory>
```

Where `<home_directory>` is the path to the new home directory.

4. Click **Add**, then save the new configuration and restart, as prompted.

4.12. Managing Atlas in a Storm Environment

When you update the Apache Atlas configuration settings in Ambari, Ambari marks the services that require a restart. To restart these services, follow these steps:

1. In Ambari Web, click the **Actions** control.
2. Click **Restart All Required**.



Important

Apache Oozie requires a restart after an Atlas configuration update, but might not be marked as requiring restart in Ambari. If Oozie is not included, follow these steps to restart Oozie:

1. In Ambari Web, click **Oozie** in the services summary pane on the left of the display.
2. Click **Service Actions > Restart All**.

More Information

[Installing and Configuring Atlas Using Ambari](#)

[Storm Guide](#)

5. Managing Service High Availability

Ambari web provides a wizard-driven user experience that enables you to configure high availability of the components in many Hortonworks Data Platform (HDP) stack services. High availability is assured through establishing primary and secondary components. In the event that the primary component fails or becomes unavailable, the secondary component is available. After configuring high availability for a service, Ambari enables you to manage and disable (roll back) high availability of components in that service.

- [NameNode High Availability \[44\]](#)
- [ResourceManager High Availability \[64\]](#)
- [HBase High Availability \[67\]](#)
- [Hive High Availability \[72\]](#)
- [Oozie High Availability \[74\]](#)
- [Apache Atlas High Availability \[75\]](#)
- [Enabling Ranger Admin High Availability \[77\]](#)

5.1. NameNode High Availability

To ensure that another NameNode in your cluster is always available if the primary NameNode host fails, you should enable and configure NameNode high availability on your cluster using Ambari Web.

More Information

[Configuring NameNode High Availability \[44\]](#)

[Rolling Back NameNode HA \[49\]](#)

[Managing Journal Nodes \[59\]](#)

5.1.1. Configuring NameNode High Availability

Prerequisites

- Verify that you have at least three hosts in your cluster and are running at least three Apache ZooKeeper servers.
- Verify that the Hadoop Distributed File System (HDFS) and ZooKeeper services are not in Maintenance Mode.

HDFS and ZooKeeper must stop and start when enabling NameNode HA. Maintenance Mode will prevent those start and stop operations from occurring. If the HDFS or

ZooKeeper services are in Maintenance Mode the NameNode HA wizard will not complete successfully.

Steps

1. In Ambari Web, select **Services > HDFS > Summary**.
2. Click **Service Actions**, then click **Enable NameNode HA**.
3. The Enable HA wizard launches. This wizard describes the set of automated and manual steps you must take to set up NameNode high availability.
4. On the **Get Started** page, type in a Nameservice ID and click **Next**.

ENABLE NAMEDNODE HA WIZARD

- Get Started**
- Select Hosts
- Review
- Create Checkpoint
- Configure Components
- Initialize JournalNodes
- Start Components
- Initialize Metadata
- Finalize HA Setup

Get Started

This wizard will walk you through enabling NameNode HA on your cluster. Once enabled, you will be running a Standby NameNode in addition to your Active NameNode. This allows for an Active-Standby NameNode configuration that automatically performs failover.

The process to enable HA involves a combination of **automated steps** (that will be handled by the wizard) and **manual steps** (that you must perform in sequence as instructed by the wizard).

You should plan a cluster maintenance window and prepare for cluster downtime when enabling NameNode HA.

If you have HBase running, please exit this wizard and stop HBase first.

Nameservice ID:

Next →

You use this Nameservice ID instead of the NameNode FQDN after HA is set up.

5. On the **Select Hosts** page, select a host for the additional NameNode and the JournalNodes, and then click **Next**:

Select Hosts

Select a host that will be running the additional NameNode.
In addition, select the hosts to run JournalNodes, which store NameNode edit logs in a fault tolerant manner.

Current NameNode:

Additional NameNode:

JournalNode:

JournalNode:

JournalNode:

c6401.ambari.apache.org (1.8 GB, 1 cores)

NameNode HBase Master ZooKeeper

JournalNode

c6402.ambari.apache.org (1.8 GB, 1 cores)

SNameNode History Server

ResourceManager App Timeline Server

Nagios Server Ganglia Server

HiveServer2 Hive Metastore

WebHCat Server Oozie Server

ZooKeeper Falcon Server Nimbus

Storm UI Server Logviewer Server

DRPC Server Storm REST API Server

JournalNode **NameNode**

c6403.ambari.apache.org (1.8 GB, 1 cores)

ZooKeeper **JournalNode**

6. On the **Review** page, confirm your host selections and click **Next**:

Review

Confirm your host selections.

Current NameNode: c6401.ambari.apache.org

Secondary NameNode: c6402.ambari.apache.org **- TO BE DELETED**

Additional NameNode: c6402.ambari.apache.org **+ TO BE INSTALLED**

JournalNode: c6401.ambari.apache.org **+ TO BE INSTALLED**
c6402.ambari.apache.org **+ TO BE INSTALLED**
c6403.ambari.apache.org **+ TO BE INSTALLED**

Review Configuration Changes.
The following lists the configuration changes that will be made by the Wizard to enable NameNode HA. This information is for review only and is not editable except for the `dfs.journalnode.edits.dir` property

- ▶ HDFS
- ▶ HBase

7. Follow the directions on the **Manual Steps Required: Create Checkpoint on NameNode** page, and then click **Next**:

Manual Steps Required: Create Checkpoint on NameNode

1. Login to the NameNode host c6401.ambari.apache.org.
2. Put the NameNode in Safe Mode (read-only mode):

```
sudo su -l hdfs -c 'hdfs dfsadmin -safemode enter'
```
3. Once in Safe Mode, create a Checkpoint:

```
sudo su -l hdfs -c 'hdfs dfsadmin -saveNamespace'
```
4. You will be able to proceed once Ambari detects that the NameNode is in Safe Mode and the Checkpoint has been created successfully.

If the Next button is enabled before you run the "Step 3: Create a Checkpoint" command, it means there is a recent Checkpoint already and you may proceed without running the "Step 3: Create a Checkpoint" command.

Checkpoint created [Next →](#)

You must log in to your *current* NameNode host and run the commands to put your NameNode into safe mode and create a checkpoint.

8. When Ambari detects success and the message on the bottom of the window changes to `Checkpoint created`, click **Next**.
9. On the **Configure Components** page, monitor the configuration progress bars, then click **Next**:

Configure Components

Please proceed to the next step.

- ✓ Stop All Services
- ✓ Install Additional NameNode
- ✓ Install JournalNodes
- ✓ Reconfigure HDFS
- ✓ Start JournalNodes
- ✓ Disable Secondary NameNode

[Next](#)

10. Follow the instructions on the **Manual Steps Required: Initialize JournalNodes** page and then click **Next**:

Manual Steps Required: Initialize JournalNodes

1. Login to the NameNode host c6401.ambari.apache.org.
2. Initialize the JournalNodes by running:

```
sudo su -l hdfs -c 'hdfs namenode -initializeSharedEdits'
```
3. You will be able to proceed once Ambari detects that the JournalNodes have been initialized successfully.

JournalNodes initialized [Next →](#)

You must log in to your *current* NameNode host to run the command to initialize the JournalNodes.

11. When Ambari detects success and the message on the bottom of the window changes to `JournalNodes initialized`, click **Next**.
12. On the **Start Components** page, monitor the progress bars as the ZooKeeper servers and NameNode start; then click **Next**:

Start Components

Please proceed to the next step.

- ✓ Start ZooKeeper Servers
- ✓ Start NameNode

[Next](#)



Note

In a cluster with Ranger enabled, and with Hive configured to use MySQL, Ranger will fail to start if MySQL is stopped. To work around this issue, start the Hive MySQL database and then retry starting components.

13. On the **Manual Steps Required: Initialize NameNode HA Metadata** page: Complete each step, using the instructions on the page, and then click **Next**.

Manual Steps Required: Initialize NameNode HA Metadata

1. Login to the NameNode host `c6401.ambari.apache.org`.
2. Initialize the metadata for NameNode automatic failover by running:


```
sudo su -i hdfs -c "hdfs zkfc -formatzk"
```
3. Login to the Additional NameNode host `c6402.ambari.apache.org`.

Important! Be sure to login to the Additional NameNode host. This is a different host from the Steps 1 and 2 above.
4. Initialize the metadata for the Additional NameNode by running:


```
sudo su -i hdfs -c "hdfs namenode -bootstrapstandby"
```

Please proceed once you have completed the steps above.

[Next →](#)

For this step, you must log in to both the *current* NameNode and the *additional* NameNode. Make sure you are logged in to the correct host for each command. Click **OK** to confirm, after you complete each command.

14. On the **Finalize HA Setup** page, monitor the progress bars as the wizard completes HA setup, then click **Done** to finish the wizard.

Finalize HA Setup

Please wait while the wizard finalizes the HA setup.

- ✓ Start Additional NameNode
- ✓ Install Failover Controllers
- ✓ Start Failover Controllers
- ✓ Reconfigure HBase
- ✓ Delete Secondary NameNode
- ⚙ Start All Services 72%

[Done](#)

After the Ambari Web UI reloads, you may see some alert notifications. Wait a few minutes until all the services restart.

15. Restart any components using Ambari Web, if necessary.

16. If you are using Hive, you must manually change the Hive Metastore FS root to point to the Nameservice URI instead of the NameNode URI. You created the Nameservice ID in the Get Started step.

Steps

a. Find the current FS root on the Hive host:

```
hive --config /etc/hive/conf/conf.server --service metatool -listFSRoot
```

The output should look similar to Listing FS Roots... `hdfs://<namenode-host>/apps/hive/warehouse`.

b. Change the FS root:

```
$ hive --config /etc/hive/conf/conf.server --service metatool
-updateLocation <new-location><old-location>
```

For example, if your Nameservice ID is `mycluster`, you input:

```
$ hive --config /etc/hive/conf/conf.server --service metatool
-updateLocation hdfs://mycluster/apps/hive/warehouse hdfs://
c6401.ambari.apache.org/apps/hive/warehouse.
```

The output looks similar to:

```
Successfully updated the following locations...Updated X
records in SDS table
```



Important

The Hive configuration path for a default HDP 2.3.x or later stack is /
`etc/hive/conf/conf.server`

The Hive configuration path for a default HDP 2.2.x or earlier stack is /
`etc/hive/conf`

17 Adjust the ZooKeeper Failover Controller retries setting for your environment:

- a. Browse to **Services > HDFS > Configs > Advanced core-site**.
- b. Set `ha.failover-controller.active-standby-electector.zk.op.retries=120`.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

5.1.2. Rolling Back NameNode HA

To disable (roll back) NameNode high availability, perform these tasks (depending on your installation):

1. [Stop HBase \[50\]](#)
2. [Checkpoint the Active NameNode \[50\]](#)
3. [Stop All Services \[51\]](#)
4. [Prepare the Ambari Server Host for Rollback \[51\]](#)

5. [Restore the HBase Configuration \[52\]](#)
6. [Delete ZooKeeper Failover Controllers \[53\]](#)
7. [Modify HDFS Configurations \[53\]](#)
8. [Re-create the Secondary NameNode \[55\]](#)
9. [Re-enable the Secondary NameNode \[56\]](#)
10. [Delete All JournalNodes \[57\]](#)
11. [Delete the Additional NameNode \[58\]](#)
12. [Verify the HDFS Components \[58\]](#)
13. [Start HDFS \[59\]](#)

More Information

[Configuring NameNode High Availability \[44\]](#)

5.1.2.1. Stop HBase

1. In the Ambari Web cluster dashboard, click the **HBase** service.
2. Click **Service Actions > Stop**.
3. Wait until HBase has stopped completely before continuing.

5.1.2.2. Checkpoint the Active NameNode

If HDFS is used *after* you enable NameNode HA, but you want to revert to a non-HA state, you must checkpoint the HDFS state before proceeding with the rollback.

If the **Enable NameNode HA wizard** failed and you need to revert, you can omit this step and proceed to stop all services.

Checkpointing the HDFS state requires different syntax, depending on whether Kerberos security is enabled on the cluster or not:

- If Kerberos security has *not* been enabled on the cluster, use the following command on the active NameNode host and as the HDFS service user, to save the namespace:

```
sudo su -l <HDFS_USER> -c 'hdfs dfsadmin -safemode enter' sudo su -l <HDFS_USER> -c 'hdfs dfsadmin -saveNamespace'
```

- If Kerberos security *has* been enabled on the cluster, use the following commands to save the namespace:

```
sudo su -l <HDFS_USER> -c 'kinit -kt /etc/security/keytabs/nn.service.keytab nn/<HOSTNAME>@<REALM>;hdfs dfsadmin -safemode enter' sudo su -l <HDFS_USER> -c 'kinit -kt /etc/security/
```

```
keytabs/nn.service.keytab nn/<HOSTNAME>@<REALM>;hdfs dfsadmin -
saveNamespace'
```

In this example, **<HDFS_USER>** is the HDFS service user (for example, `hdfs`), **<HOSTNAME>** is the Active NameNode hostname, and **<REALM>** is your Kerberos realm.

More Information

[Stop All Services \[51\]](#)

5.1.2.3. Stop All Services

After stopping HBase and, if necessary, checkpointing the Active NameNode, stop all services:

1. In Ambari Web, click the **Services** tab.
2. Click **Stop All**.
3. Wait for all services to stop completely before continuing.

5.1.2.4. Prepare the Ambari Server Host for Rollback

To prepare for the rollback procedure:

Steps

1. Log in to the Ambari server host.
2. Set the following environment variables

```
export AMBARI_USER=AMBARI_USERNAME
Substitute the value of the administrative user for Ambari Web. The default value is admin.
```

```
export AMBARI_PW=AMBARI_PASSWORD
Substitute the value of the administrative password for Ambari Web. The default value is admin.
```

```
export AMBARI_PORT=AMBARI_PORT
Substitute the Ambari Web port. The default value is 8080.
```

```
export AMBARI_PROTO=AMBARI_PROTOCOL
Substitute the value of the protocol for connecting to Ambari Web. Options are http or https. The default value is http.
```

```
export CLUSTER_NAME=CLUSTER_NAME
Substitute the name of your cluster, which you set during installation: for example, mycluster.
```

```
export NAMENODE_HOSTNAME=NN_HOSTNAME
Substitute the FQDN of the host for the non-HA NameNode: for example, nn01.mycompany.com.
```

```
export ADDITIONAL_NAMENODE_HOSTNAME=ADDITIONAL_NAMENODE_HOSTNAME
Substitute the FQDN of the host for the additional NameNode in a non-HA setup.
```

```
export SECONDARY_NAMENODE_HOSTNAME=SECONDARY_NAMENODE_HOSTNAME
Substitute the FQDN of the host for the secondary NameNode in a non-HA setup.
```



```
export JOURNALNODE1_HOSTNAME=JOURNAL1_HOSTNAME
```

Substitute the FQDN of the host for the first Journal Node

```
export JOURNALNODE2_HOSTNAME=JOURNAL2_HOSTNAME
```

Substitute the FQDN of the host for the second Journal Node

```
export JOURNALNODE3_HOSTNAME=JOURNAL3_HOSTNAME
```

Substitute the FQDN of the host for the third Journal Node

3. Double check that these environment variables are set correctly.

5.1.2.5. Restore the HBase Configuration

If you have installed HBase, you might need to restore a configuration to its pre-HA state:



Note

For Ambari 2.6.0 and higher, `config.sh` is not supported and will fail. Use `config.py` instead.

1. From the Ambari server host, determine whether your current HBase configuration must be restored:

```
/var/lib/ambari-server/resources/scripts/configs.py -u
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> get localhost
<CLUSTER_NAME> hbase-site
```

Use the environment variables that you set up when preparing the Ambari server host for rollback for the variable names.

If `hbase.rootdir` is set to the NameService ID you set up using the **Enable NameNode HA** wizard, you must revert `hbase-site` to non-HA values. For example, in `"hbase.rootdir": "hdfs://<name-service-id>:8020/apps/hbase/data"`, the `hbase.rootdir` property points to the NameService ID and the value must be rolled back.

If `hbase.rootdir` points instead to a specific NameNode host, it does not need to be rolled back. For example, in `"hbase.rootdir": "hdfs://<nn01.mycompany.com>:8020/apps/hbase/data"`, the `hbase.rootdir` property points to a specific NameNode host and not a NameService ID. This does not need to be rolled back; you can proceed to delete ZooKeeper failover controllers.

2. If you must roll back the `hbase.rootdir` value, on the Ambari Server host, use the `config.sh` script to make the necessary change:

```
/var/lib/ambari-server/resources/scripts/configs.py -
u <AMBARI_USER> -p<AMBARI_PW> -port <AMBARI_PORT> set
localhost <CLUSTER_NAME> hbase-site hbase.rootdir hdfs://
<NAMENODE_HOSTNAME>:8020/apps/hbase/data
```

Use the environment variables that you set up when preparing the Ambari server host for rollback for the variable names.

3. On the Ambari server host, verify that the `hbase.rootdir` property has been restored properly:

```
/var/lib/ambari-server/resources/scripts/configs.py -u  
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> get localhost  
<CLUSTER_NAME> hbase-site
```

The `hbase.rootdir` property should now be the same as the NameNode hostname, not the NameService ID.

More Information

[Prepare the Ambari Server Host for Rollback \[51\]](#)

[Delete ZooKeeper Failover Controllers \[53\]](#)

5.1.2.6. Delete ZooKeeper Failover Controllers

Prerequisites

If the following command on the Ambari server host returns an empty `items` array then you must delete ZooKeeper (ZK) Failover Controllers:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari"  
-i <AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/  
<CLUSTER_NAME>/host_components?HostRoles/component_name=ZKFC
```

To delete the failover controllers:

Steps

1. On the Ambari Server host, issue the following DELETE commands:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari"  
-i -X DELETE <AMBARI_PROTO>://localhost:<AMBARI_PORT>/  
api/v1/clusters/<CLUSTER_NAME>/hosts/<NAMENODE_HOSTNAME>/  
host_components/ZKFC curl -u <AMBARI_USER>:<AMBARI_PW> -  
H "X-Requested-By: ambari" -i -X DELETE <AMBARI_PROTO>://  
localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>/hosts/  
<ADDITIONAL_NAMENODE_HOSTNAME>/host_components/ZKFC
```

2. Verify that the controllers are gone:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari"  
-i <AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/  
<CLUSTER_NAME>/host_components?HostRoles/component_name=ZKFC
```

This command should return an empty `items` array.

5.1.2.7. Modify HDFS Configurations

You may need to modify your `hdfs-site` configuration and/or your `core-site` configuration.



Note

For Ambari 2.6.0 and higher, `config.sh` is not supported and will fail. Use `config.py` instead.

Prerequisites

Check whether you need to modify your `hdfs-site` configuration, by executing the following command on the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.py -u  
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> get localhost  
<CLUSTER_NAME> hdfs-site
```

If you see **any** of the following properties, you must delete them from your configuration.

- `dfs.nameservices`
- `dfs.client.failover.proxy.provider.<NAMESERVICE_ID>`
- `dfs.ha.namenodes.<NAMESERVICE_ID>`
- `dfs.ha.fencing.methods`
- `dfs.ha.automatic-failover.enabled`
- `dfs.namenode.http-address.<NAMESERVICE_ID>.nn1`
- `dfs.namenode.http-address.<NAMESERVICE_ID>.nn2`
- `dfs.namenode.rpc-address.<NAMESERVICE_ID>.nn1`
- `dfs.namenode.rpc-address.<NAMESERVICE_ID>.nn2`
- `dfs.namenode.shared.edits.dir`
- `dfs.journalnode.edits.dir`
- `dfs.journalnode.http-address`
- `dfs.journalnode.kerberos.internal.spnego.principal`
- `dfs.journalnode.kerberos.principal`
- `dfs.journalnode.keytab.file`

Where `<NAMESERVICE_ID>` is the NameService ID you created when you ran the **Enable NameNode HA** wizard.

To modify your `hdfs-site` configuration:

Steps

1. On the Ambari Server host, execute the following *for each property* you found:

```
/var/lib/ambari-server/resources/scripts/configs.py -u  
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> delete  
localhost <CLUSTER_NAME> hdfs-site property_name
```

Replace `property_name` with the name of each of the properties to be deleted.

2. Verify that all of the properties have been deleted:

```
/var/lib/ambari-server/resources/scripts/configs.py -u  
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> get localhost  
<CLUSTER_NAME> hdfs-site
```

None of the properties listed above should be present.

3. Determine whether you must modify your `core-site` configuration:

```
/var/lib/ambari-server/resources/scripts/configs.py -u  
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> get localhost  
<CLUSTER_NAME> core-site
```

4. If you see the property `ha.zookeeper.quorum`, delete it:

```
/var/lib/ambari-server/resources/scripts/configs.py -u  
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> delete  
localhost <CLUSTER_NAME> core-site ha.zookeeper.quorum
```

5. If the property `fs.defaultFS` is set to the NameService ID, revert it to its non-HA value:

```
"fs.defaultFS":"hdfs://<name-service-id>" The property  
fs.defaultFS needs to be modified as it points to a NameService  
ID "fs.defaultFS":"hdfs://<nn01.mycompany.com>"
```

You need not change the property `fs.defaultFS`, because it points to a specific NameNode, not to a NameService ID.

6. Revert the property `fs.defaultFS` to the NameNode host value:

```
/var/lib/ambari-server/resources/scripts/configs.py -u  
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> set localhost  
<CLUSTER_NAME> core-site fs.defaultFS hdfs://<NAMENODE_HOSTNAME>
```

7. Verify that the `core-site` properties are now properly set:

```
/var/lib/ambari-server/resources/scripts/configs.py -u  
<AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> get localhost  
<CLUSTER_NAME> core-site
```

The property `fs.defaultFS` should be the NameNode host and the property `ha.zookeeper.quorum` should not appear.

5.1.2.8. Re-create the Secondary NameNode

You may need to recreate your secondary NameNode.

Prerequisites

Check whether you need to recreate the secondary NameNode, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By:
ambari" -i -X GET <AMBARI_PROTO>://localhost:<AMBARI_PORT>/
api/v1/clusters/<CLUSTER_NAME>/host_components?HostRoles/
component_name=SECONDARY_NAMENODE
```

If this returns an empty `items` array, you must *recreate* your secondary NameNode. Otherwise you can proceed to *re-enable* your secondary NameNode.

To recreate your secondary NameNode:

Steps

1. On the Ambari Server host, run the following command:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By:
ambari" -i -X POST -d '{"host_components" : [{"HostRoles":
{"component_name": "SECONDARY_NAMENODE"}]}]' <AMBARI_PROTO>://
localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>/hosts?
Hosts/host_name=<SECONDARY_NAMENODE_HOSTNAME>
```

2. Verify that the secondary NameNode now exists. On the Ambari Server host, run the following command:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By:
ambari" -i -X GET <AMBARI_PROTO>://localhost:<AMBARI_PORT>/
api/v1/clusters/<CLUSTER_NAME>/host_components?HostRoles/
component_name=SECONDARY_NAMENODE
```

This should return a non-empty `items` array containing the secondary NameNode.

More Information

[Re-enable the Secondary NameNode \[56\]](#)

5.1.2.9. Re-enable the Secondary NameNode

To re-enable the secondary NameNode:

Steps

1. Run the following commands on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-
By: ambari" -i -X PUT -d '{"RequestInfo":
{"context": "Enable Secondary NameNode"}, "Body":
{"HostRoles": {"state": "INSTALLED"} }]' <AMBARI_PROTO>://
localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>/hosts/
<SECONDARY_NAMENODE_HOSTNAME>/host_components/SECONDARY_NAMENODE
```

2. Analyze the output:

- If this returns 200, proceed to delete all JournalNodes.
- If this input returns the value 202, wait a few minutes and then run the following command:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By:
ambari" -i -X GET "<AMBARI_PROTO>://localhost:<AMBARI_PORT>/
api/v1/clusters/<CLUSTER_NAME>/host_components?HostRoles/
component_name=SECONDARY_NAMENODE&fields=HostRoles/state"
```

Wait for the response "state" : "INSTALLED" before proceeding.

More Information

[Delete All JournalNodes \[57\]](#)

5.1.2.10. Delete All JournalNodes

You may need to delete any JournalNodes.

Prerequisites

Check to see if you need to delete JournalNodes, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By:
ambari" -i -X GET <AMBARI_PROTO>://localhost:<AMBARI_PORT>/
api/v1/clusters/<CLUSTER_NAME>/host_components?HostRoles/
component_name=JOURNALNODE
```

If this returns an empty `items` array, you can go on to [Delete the Additional NameNode](#). Otherwise you must delete the JournalNodes.

To delete the JournalNodes:

Steps

1. On the Ambari Server host, run the following command:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari"
-i -X DELETE <AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/
v1/clusters/<CLUSTER_NAME>/hosts/<JOURNALNODE1_HOSTNAME>/
host_components/JOURNALNODE curl -u <AMBARI_USER>:<AMBARI_PW>
-H "X-Requested-By: ambari" -i -X DELETE <AMBARI_PROTO>://
localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>/hosts/
<JOURNALNODE2_HOSTNAME>/host_components/JOURNALNODE
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari"
-i -X DELETE <AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/
v1/clusters/<CLUSTER_NAME>/hosts/<JOURNALNODE3_HOSTNAME>/
host_components/JOURNALNODE
```

2. Verify that all the JournalNodes have been deleted. On the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By:
ambari" -i -X GET <AMBARI_PROTO>://localhost:<AMBARI_PORT>/
```

```
api/v1/clusters/<CLUSTER_NAME>/host_components?HostRoles/  
component_name=JOURNALNODE
```

This should return an empty `items` array.

More Information

[Delete the Additional NameNode \[58\]](#)

[Delete All JournalNodes \[57\]](#)

5.1.2.11. Delete the Additional NameNode

You may need to delete your Additional NameNode.

Prerequisites

Check to see if you need to delete your Additional NameNode, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i  
-X GET <AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/  
<CLUSTER_NAME>/host_components?HostRoles/component_name=NAMENODE
```

If the `items` array contains two NameNodes, the Additional NameNode must be deleted.

To delete the Additional NameNode that was set up for HA:

Steps

1. On the Ambari Server host, run the following command:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari"  
-i -X DELETE <AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/  
clusters/<CLUSTER_NAME>/hosts/<ADDITIONAL_NAMENODE_HOSTNAME>/  
host_components/NAMENODE
```

2. Verify that the Additional NameNode has been deleted:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i  
-X GET <AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/  
<CLUSTER_NAME>/host_components?HostRoles/component_name=NAMENODE
```

This should return an `items` array that shows only one NameNode.

5.1.2.12. Verify the HDFS Components

Before starting HDFS, verify that you have the correct components:

1. Go to **Ambari Web UI > Services**; then select **HDFS**.
2. Check the Summary panel and ensure that the first three lines look like this:
 - NameNode
 - SNameNode

- DataNodes

You should *not* see a line for JournalNodes.

5.1.2.13. Start HDFS

1. In the **Ambari Web UI**, click **Service Actions**, then click **Start**.
2. If the progress bar does not show that the service has completely started and has passed the service checks, repeat Step 1.
3. To start all of the other services, click **Actions > Start All** in the **Services** navigation panel.

5.1.3. Managing Journal Nodes

After you enable NameNode high availability in your cluster, you must maintain at least three, active JournalNodes in your cluster. You can use the Manage JournalNode wizard to assign, add, or remove JournalNodes on hosts in your cluster. The Manage JournalNode wizard enables you to assign JournalNodes, review and confirm required configuration changes, and will restart all components in the cluster to take advantage of the changes made to JournalNode placement and configuration.

Please note that this wizard will restart all cluster services.

Prerequisites

- NameNode high availability must be enabled in your cluster

To manage JournalNodes in your cluster:

Steps

1. In Ambari Web, select **Services > HDFS > Summary**.
2. Click **Service Actions**, then click **Manage JournalNodes**.

The screenshot shows the Ambari Web UI for the HDFS service. The 'Summary' tab is active, displaying the following information:

- Standby NameNode:** Started, No alerts, Disk Remaining: 36.7 GB
- ZooKeeper:** Started, No alerts, Blocks (total): 15
- Active NameNode:** Started, No alerts, Block Errors: 0 corrupt replicate
- ZooKeeper:** Started, No alerts
- DataNodes:** 3/3 Started, Total Files + Directories: 51
- DataNodes Status:** 3 live / 0 dead / 0 decommissioning, Upgrade Status: No pending
- JournalNodes:** 3/3 JournalNodes Live, Safe Mode Status: Not in safe mode
- NFS Gateways:** 0/0 Started
- NameNode Uptime:** 23.34 hours
- NameNode Heap:** 94.9 MB / 1011.3 MB (9.4% used)
- Disk Usage (DFS Used):** 741.0 MB / 44.9 GB (1.61%)

The 'Service Actions' menu is open, showing the following options:

- Start
- Stop
- Restart All
- Restart DataNodes
- Restart JournalNodes
- Restart ZooKeeper
- Restart ZooKeeper
- Move NameNode
- Manage JournalNodes** (highlighted)
- Run Service Check
- Turn On Maintenance Mode
- Rebalance HDFS
- Download Client Configs
- Delete Service

3. On the **Assign JournalNodes** page, make assignments by clicking the + and - icons and selecting host names in the drop-down menus. The **Assign JournalNodes** page

enables you to maintain three, current JournalNodes by updating each time you make an assignment.

Manage JournalNode Wizard

When you complete your assignments, click **Next**.

4. On the **Review** page, verify the summary of your JournalNode host assignments and the related configuration changes. When you are satisfied that all assignments match your intentions, click **Next**:

Manage JournalNode Wizard

5. Using a remote shell, complete the steps on the **Save Namespace** page. When you have successfully created a checkpoint, click **Next**:

Manage JournalNode Wizard

6. On the **Add/Remove JournalNodes** page, monitor the progress bars, then click **Next**:

Manage JournalNode Wizard

7. Follow the instructions on the **Manual Steps Required: Format JournalNodes** page and then click **Next**:

Manage JournalNode Wizard

The screenshot shows the 'Manage JournalNode Wizard' interface. On the left is a sidebar with a list of steps: 'MANAGE JOURNALNODE WIZARD', 'Assign JournalNodes', 'Review', 'Save Namespace', 'Add/Remove JournalNodes', 'Format JournalNodes' (highlighted), 'Start Active NameNode', 'BootStrap StandBy NameNode', and 'Start All Services'. The main content area is titled 'Manual Steps Required: Format JournalNodes' and contains the following instructions:

1. Login to the NameNode host `revo2.hortonworks.local`.
2. Initialize the JournalNodes by running:


```
sudo su hdfs -l -c "hdfs namenode -initializeSharedDir"
```
3. You will be able to proceed once Ambari detects that the JournalNodes have been initialized successfully.

At the bottom right of the main content area, it says 'JournalNodes initialized' and has a green 'Next →' button.

8. In the remote shell, confirm that you want to initialize JournalNodes, by entering `Y`, at the following prompt:

```
Re-format filesystem in QJM to [host.ip.address.1,
host.ip.address.2, host.ip.address.3,] ? (Y or N) Y
```

9. On the **Start Active NameNodes** page, monitor the progress bars as services re-start; then click **Next**:

Manage JournalNode Wizard

The screenshot shows the 'Manage JournalNode Wizard' interface. On the left is a sidebar with a list of steps: 'MANAGE JOURNALNODE WIZARD', 'Assign JournalNodes', 'Review', 'Save Namespace', 'Add/Remove JournalNodes', 'Format JournalNodes', 'Start Active NameNode' (highlighted), 'BootStrap StandBy NameNode', and 'Start All Services'. The main content area is titled 'Start Active NameNode' and contains the following instructions:

Please proceed to the next step.

- ✓ Start Zookeeper Server
- ✓ Start Active NameNode

At the bottom right of the main content area, there is a green 'Next' button.

10. On the **Manual Steps Required: Bootstrap Standby NameNode** page: Complete each step, using the instructions on the page, and then click **Next**.

Manage JournalNode Wizard

The screenshot shows the 'Manage JournalNode Wizard' interface. On the left is a sidebar with a list of steps: MANAGE JOURNALNODE WIZARD, Assign JournalNodes, Review, Save Namespace, Add/Remove JournalNodes, Format JournalNodes, Start Active NameNode, **BootStrap StandBy NameNode** (highlighted), and Start All Services. The main content area is titled 'Manual Steps Required: BootStrap Standby NameNode'. It contains the following instructions:

3. Login to the Additional NameNode host `rev01.hortonworks.local`.

Important! Be sure to login to the Additional NameNode host. This is a different host from previous steps.

4. Initialize the metadata for the Additional NameNode by running:


```
sudo su -hdfs -s /bin/bash -c "hdfs namenode -bootstrapstandby"
```

Please proceed once you have completed the steps above.

A green 'Next ->' button is visible at the bottom right.

11. In the remote shell, confirm that you want to bootstrap the standby NameNode, by entering `Y`, at the following prompt:

```
RE-format filesystem in Storage Directory /grid/0/hadoop/hdfs/
namenode ? (Y or N) Y
```

12. On the **Start All Services** page, monitor the progress bars as the wizard starts all services, then click **Done** to finish the wizard.

Manage JournalNode Wizard

The screenshot shows the 'Manage JournalNode Wizard' interface at the 'Start All Services' step. The sidebar on the left is the same as in the previous screenshot, but the 'Start All Services' button is highlighted. The main content area is titled 'Start All Services' and contains the following information:

Wait all services to be started

- ✔ Stop HDFS
- ⚙ Start All Services (73% progress bar)

A green 'Done' button is visible at the bottom right.

After the Ambari Web UI reloads, you may see some alert notifications. Wait a few minutes until all the services restart and alerts clear.

13. Restart any components using Ambari Web, if necessary.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

[Configuring NameNode High Availability \[44\]](#)

5.2. ResourceManager High Availability

If you are working in an HDP 2.2 or later environment, you can configure high availability for ResourceManager by using the Enable ResourceManager HA wizard.

Prerequisites

You must have at least three:

- hosts in your cluster
- Apache ZooKeeper servers running

More Information

[Configure ResourceManager High Availability \[64\]](#)

[Disable ResourceManager High Availability \[65\]](#)

5.2.1. Configure ResourceManager High Availability

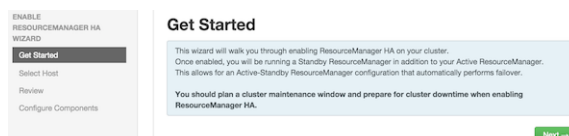
To access the wizard and configure ResourceManager high availability:

Steps

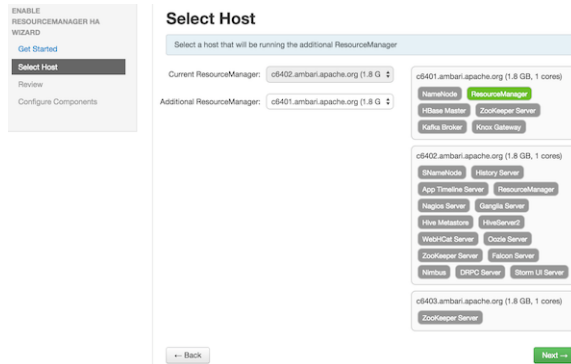
1. In Ambari Web, browse to **Services > YARN > Summary**.
2. Select **Service Actions** and choose **Enable ResourceManager HA**.

The **Enable ResourceManager HA** wizard launches, describing a set of automated and manual steps that you must take to set up ResourceManager high availability.

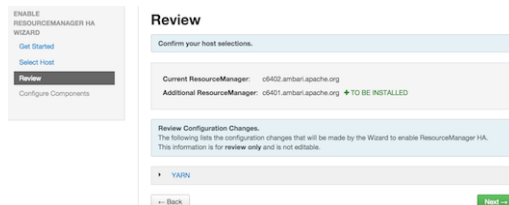
3. On the **Get Started** page, read the overview of enabling ResourceManager HA; then click **Next** to proceed:



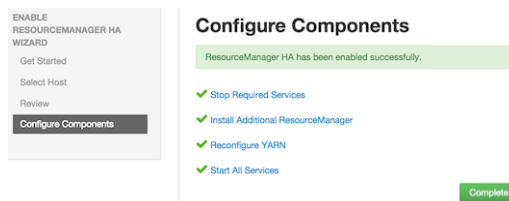
4. On the **Select Host** page, accept the default selection, or choose an available host, then click **Next** to proceed.



5. On the **Review Selections** page, expand YARN, if necessary, to review all the configuration changes proposed for YARN. Click **Next** to approve the changes and start automatically configuring ResourceManager HA.



6. On the **Configure Components** page, click **Complete** when all the progress bars finish tracking:



More Information

[Disable ResourceManager High Availability \[65\]](#)

5.2.2. Disable ResourceManager High Availability

To disable ResourceManager high availability, you must delete one ResourceManager and keep one ResourceManager. This requires using the Ambari API to modify the cluster configuration to delete the ResourceManager and using the ZooKeeper client to update the znode permissions.

Prerequisites

Because these steps involve using the Ambari REST API, you should test and verify them in a test environment prior to executing against a production environment.

To disable ResourceManager high availability:

Steps

1. In **Ambari Web**, stop YARN and ZooKeeper services.
2. On the Ambari Server host, use the Ambari API to retrieve the YARN configurations into a JSON file:



Note

For Ambari 2.6.0 and higher, `config.sh` is not supported and will fail. Use `config.py` instead.

```
/var/lib/ambari-server/resources/scripts/configs.py get <ambari.server>  
<cluster.name> yarn-site yarn-site.json
```

In this example, `ambari.server` is the hostname of your Ambari Server and `cluster.name` is the name of your cluster.

3. In the `yarn-site.json` file, change `yarn.resourcemanager.ha.enabled` to `false` and delete the following properties:
 - `yarn.resourcemanager.ha.rm-ids`
 - `yarn.resourcemanager.hostname.rm1`
 - `yarn.resourcemanager.hostname.rm2`
 - `yarn.resourcemanager.webapp.address.rm1`
 - `yarn.resourcemanager.webapp.address.rm2`
 - `yarn.resourcemanager.webapp.https.address.rm1`
 - `yarn.resourcemanager.webapp.https.address.rm2`
 - `yarn.resourcemanager.cluster-id`
 - `yarn.resourcemanager.ha.automatic-failover.zk-base-path`
4. Verify that the following properties in the `yarn-site.json` file are set to the ResourceManager hostname you are keeping:
 - `yarn.resourcemanager.hostname`
 - `yarn.resourcemanager.admin.address`
 - `yarn.resourcemanager.webapp.address`
 - `yarn.resourcemanager.resource-tracker.address`
 - `yarn.resourcemanager.scheduler.address`
 - `yarn.resourcemanager.webapp.https.address`
 - `yarn.timeline-service.webapp.address`

- yarn.timeline-service.webapp.https.address
 - yarn.timeline-service.address
 - yarn.log.server.url
5. Search the `yarn-site.json` file and remove any references to the ResourceManager hostname that you are removing.
 6. Search the `yarn-site.json` file and remove any properties that might still be set for ResourceManager IDs: for example, `rm1` and `rm2`.
 7. Save the `yarn-site.json` file and set that configuration against the Ambari Server:

```
/var/lib/ambari-server/resources/scripts/configs.py set  
ambari.server cluster.name yarn-site yarn-site.json
```

8. Using the Ambari API, delete the ResourceManager host component for the host that you are deleting:

```
curl --user admin:admin -i -H "X-Requested-By: ambari" -X DELETE  
http://ambari.server:8080/api/v1/clusters/cluster.name/hosts/  
hostname/host_components/RESOURCEMANAGER
```

9. In Ambari Web, start the ZooKeeper service.
10. On a host that has the ZooKeeper client installed, use the ZooKeeper client to change znode permissions:

```
/usr/hdp/current/zookeeper-client/bin/zkCli.sh  
getAcl /rmstore/ZKRMStateRoot  
setAcl /rmstore/ZKRMStateRoot world:anyone:rwcd
```

11. In Ambari Web, restart ZooKeeper service and start YARN service.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

5.3. HBase High Availability

To help you achieve redundancy for high availability in a production environment, Apache HBase supports deployment of multiple HBase Masters in a cluster. If you are working in a Hortonworks Data Platform (HDP) 2.2 or later environment, Apache Ambari enables simple setup of multiple HBase Masters.

During the Apache HBase service installation and depending on your component assignment, Ambari installs and configures one HBase Master component and multiple RegionServer components. To configure high availability for the HBase service, you can run two or more HBase Master components. HBase uses ZooKeeper for coordination of the

active Master in a cluster running two or more HBase Masters. This means, when primary HBase Master fails, the client will be automatically routed to secondary Master.

Set Up Multiple HBase Masters Through Ambari

Hortonworks recommends that you use Ambari to configure multiple HBase Masters. Complete the following tasks:

Add a Secondary HBase Master to a New Cluster

When installing HBase, click the “+” sign that is displayed on the right side of the name of the existing HBase Master to add and select a node on which to deploy a secondary HBase Master:



Add a New HBase Master to an Existing Cluster

1. Log in to the Ambari management interface as a cluster administrator.
2. In **Ambari Web**, browse to **Services > HBase**.
3. In **Service Actions**, click **+ Add HBase Master**.
4. Choose the host on which to install the additional HBase master; then click **Confirm Add**.

Ambari installs the new HBase Master and reconfigures HBase to manage multiple Master instances.

Set Up Multiple HBase Masters Manually

Before you can configure multiple HBase Masters manually, you must configure the first node (node-1) on your cluster by following the instructions in the Installing, [Configuring, and Deploying a Cluster](#) section in [Apache Ambari Installation Guide](#). Then, complete the following tasks:

1. Configure Passwordless SSH Access
2. Prepare node-1
3. Prepare node-2 and node-3
4. Start and test your HBase Cluster

Configure Passwordless SSH Access

The first node on the cluster (node-1) must be able to log in to other nodes on the cluster and then back to itself in order to start the daemons. You can accomplish this by using the same user name on all hosts and by using passwordless Secure Socket Shell (SSH) login:

1. On node-1, stop HBase service.
2. On node-1, log in as an HBase user and generate an SSH key pair:

```
$ ssh-keygen -t rsa
```

The system prints the location of the key pair to standard output. The default name of the public key is `id_rsa.pub`.

3. Create a directory to hold the shared keys on the other nodes:
 - On node-2, log in as an HBase user and create an `.ssh/` directory in your home directory.
 - On node-3, repeat the same procedure.
4. Use Secure Copy (`scp`) or any other standard secure means to copy the public key from node-1 to the other two nodes.

On each node in the cluster, create a new file called `.ssh/authorized_keys` (if it does not already exist) and append the contents of the `id_rsa.pub` file to it:

```
$ cat id_rsa.pub >> ~/.ssh/authorized_keys
```

Ensure that you do not overwrite your existing `.ssh/authorized_keys` files by concatenating the new key onto the existing file using the `>>` operator rather than the `>` operator.

5. Use Secure Shell (SSH) from node-1 to either of the other nodes using the same user name.

You should not be prompted for password.

6. On node-2, repeat Step 5, because it runs as a backup Master.

Prepare node-1

Because node-1 should run your primary Master and ZooKeeper processes, you must stop the RegionServer from starting on node-1:

1. Edit `conf/regionservers` by removing the line that contains `localhost` and adding lines with the host name or IP addresses for node-2 and node-3.



Note

If you want to run a RegionServer on node-1, you should refer to it by the hostname the other servers would use to communicate with it. For example, for node-1, it is called as `node-1.test.com`.

2. Configure HBase to use node-2 as a backup Master by creating a new file in `conf/` called `backup-Masters`, and adding a new line to it with the host name for node-2: for example, `node-2.test.com`.
3. Configure ZooKeeper on node-1 by editing `conf/hbase-site.xml` and adding the following properties:

```
<property>
<name>hbase.zookeeper.quorum</name>
<value>node-1.test.com,node-2.test.com,node-3.test.com</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/usr/local/zookeeper</value>
</property>
```

This configuration directs HBase to start and manage a ZooKeeper instance on each node of the cluster. You can learn more about configuring ZooKeeper in [ZooKeeper](#).

4. Change every reference in your configuration to node-1 as localhost to point to the host name that the other nodes use to refer to node-1: in this example, node-1.test.com.

Prepare node-2 and node-3

Before preparing node-2 and node-3, each node of your cluster must have the same configuration information.

node-2 runs as a backup Master server and a ZooKeeper instance.

1. Download and unpack HBase on node-2 and node-3.
2. Copy the configuration files from node-1 to node-2 and node-3.
3. Copy the contents of the conf/ directory to the conf/ directory on node-2 and node-3.

Start and Test your HBase Cluster

1. Use the jps command to ensure that HBase is not running.
2. Kill HMaster, HRegionServer, and HQuorumPeer processes, if they are running.
3. Start the cluster by running the start-hbase.sh command on node-1.

Your output is similar to this:

```
$ bin/start-hbase.sh
node-3.test.com: starting zookeeper, logging to /home/hbuser/hbase-0.98.3-
hadoop2/bin/./logs/hbase-hbuser-zookeeper-node-3.test.com.out
node-1.example.com: starting zookeeper, logging to /home/hbuser/hbase-0.98.
3-hadoop2/bin/./logs/hbase-hbuser-zookeeper-node-1.test.com.out
node-2.example.com: starting zookeeper, logging to /home/hbuser/hbase-0.98.
3-hadoop2/bin/./logs/hbase-hbuser-zookeeper-node-2.test.com.out
starting master, logging to /home/hbuser/hbase-0.98.3-hadoop2/bin/./logs/
hbase-hbuser-master-node-1.test.com.out
node-3.test.com: starting regionserver, logging to /home/hbuser/hbase-0.98.
3-hadoop2/bin/./logs/hbase-hbuser-regionserver-node-3.test.com.out
node-2.test.com: starting regionserver, logging to /home/hbuser/hbase-0.98.
3-hadoop2/bin/./logs/hbase-hbuser-regionserver-node-2.test.com.out
node-2.test.com: starting master, logging to /home/hbuser/hbase-0.98.3-
hadoop2/bin/./logs/hbase-hbuser-master-node2.test.com.out
```

ZooKeeper starts first, followed by the Master, then the RegionServers, and finally the backup Masters.

4. Run the `jps` command on each node to verify that the correct processes are running on each server.

You might see additional Java processes running on your servers as well, if they are used for any other purposes.

Example 1. node-1 jps Output

```
$ jps
20355 Jps
20071 HQuorumPeer
20137 HMaster
```

Example 2. node-2 jps Output

```
$ jps
15930 HRegionServer
16194 Jps
15838 HQuorumPeer
16010 HMaster
```

Example 3. node-3 jps Output

```
$ jps
13901 Jps
13639 HQuorumPeer
13737 HRegionServer
```

ZooKeeper Process Name



Note

The HQuorumPeer process is a ZooKeeper instance which is controlled and started by HBase. If you use ZooKeeper this way, it is limited to one instance per cluster node and is appropriate for testing only. If ZooKeeper is run outside of HBase, the process is called QuorumPeer. For more about ZooKeeper configuration, including using an external ZooKeeper instance with HBase, see [zookeeper](#) section.

5. Browse to the Web UI and test your new connections.

You should be able to connect to the UI for the Master `http://node-1.test.com:16010/` or the secondary master at `http://node-2.test.com:16010/`. If you can connect through localhost but not from another host, check your firewall rules. You can see the web UI for each of the RegionServers at port 16030 of their IP addresses, or by clicking their links in the web UI for the Master.

Web UI Port Changes



Note

In HBase newer than 0.98.x, the HTTP ports used by the HBase Web UI changed from 60010 for the Master and 60030 for each RegionServer to 16010 for the Master and 16030 for the RegionServer.

5.4. Hive High Availability

The Apache Hive service has multiple, associated components. The primary Hive components are Hive Metastore and HiveServer2. You can configure high availability for the Hive service in HDP 2.2 or later by running two or more of each of those components. The relational database that backs the Hive Metastore itself should also be made highly available using best practices defined for the database system in use and should be done after consultation with your in-house DBA.

More Information

[Adding a Hive Metastore Component \[72\]](#)

5.4.1. Adding a Hive Metastore Component

Prerequisites

If you have ACID enabled in Hive, ensure that the **Run Compactor** setting is enabled (set to True) **on only one** Hive metastore host.

Steps

1. In **Ambari Web**, browse to **Services > Hive**.
2. In **Service Actions**, click the + **Add Hive Metastore** option.
3. Choose the host to install the additional Hive Metastore; then click **Confirm Add**.
4. Ambari installs the component and reconfigures Hive to handle multiple Hive Metastore instances.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

[Using Host Config Groups](#)

5.4.2. Adding a HiveServer2 Component

Steps

1. In Ambari Web, browse to the host on which you want to install another HiveServer2 component.

2. On the Host page, click **+Add**.
3. Click **HiveServer2** from the list.

Ambari installs the additional HiveServer2.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

5.4.3. Adding a WebHCat Server

Steps

1. In Ambari Web, browse to the host on which you want to install another WebHCat Server.
2. On the **Host** page, click **+Add**.
3. Click **WebHCat** from the list.

Ambari installs the new server and reconfigures Hive to manage multiple Metastore instances.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

5.5. Storm High Availability

In HDP 2.3 or later, you can configure high availability for the Apache Storm Nimbus server by adding a Nimbus component from Ambari.

5.5.1. Adding a Nimbus Component

Steps

1. In Ambari Web, browse to **Services > Storm**.
2. In **Service Actions**, click the **+ Add Nimbus** option.
3. Click the host on which to install the additional Nimbus; then click **Confirm Add**.

Ambari installs the component and reconfigures Storm to handle multiple Nimbus instances.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

5.6. Oozie High Availability

To set up high availability for the Apache Oozie service in HDP 2.2 or later, you can run two or more instances of the Oozie Server component.

Prerequisites

- The relational database that backs the Oozie Server should also be made highly available using best practices defined for the database system in use and should be done after consultation with your in-house DBA. Using the default installed Derby database instance is not supported with multiple Oozie Server instances; therefore, you must use an existing relational database. When using Apache Derby for the Oozie Server, you do not have the option to add Oozie Server components to your cluster.
- High availability for Oozie requires the use of an external virtual IP address or load balancer to direct traffic to the Oozie servers.

More Information

[Adding an Oozie Server Component \[74\]](#)

5.6.1. Adding an Oozie Server Component

Steps

1. In **Ambari Web**, browse to the host on which you want to install another Oozie Server.
2. On the **Host** page, click the **+Add** button.
3. Click **Oozie Server** from the list.
Ambari installs the new Oozie Server.
4. Configure your external load balancer and then update the Oozie configuration.
5. Browse to **Services > Oozie > Configs**.
6. In `oozie-site`, add the following property values:

`oozie.zookeeper.connection.string` List of ZooKeeper hosts with ports: for example,

`c6401.ambari.apache.org:2181,`

`c6402.ambari.apache.org:2181,`

`c6403.ambari.apache.org:2181`

```
oozie.services.ext      org.apache.oozie.service.ZKLocksService,
                        org.apache.oozie.service.ZKXLogStreamingService,
                        org.apache.oozie.service.ZKJobsConcurrencyService

oozie.base.url          http://<Cloudbalancer.hostname>:11000/oozie
```

7. In `oozie-env`, uncomment the `oozie_base_url` property and change its value to point to the load balancer:

```
export oozie_base_url="http://<loadbalance.hostname>:11000/
oozie"
```

8. Restart Oozie.
9. Update the HDFS configuration properties for the Oozie proxy user:
 - a. Browse to **Services > HDFS > Configs**.
 - b. In `core-site`, update the `hadoop.proxyuser.oozie.hosts` property to include the newly added Oozie Server host.

Use commas to separate multiple host names.

10. Restart services.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

5.7. Apache Atlas High Availability

Prerequisites

In Ambari 2.4.0.0, adding or removing Atlas Metadata Servers requires manually editing the `atlas.rest.address` property.

Steps

1. Click **Hosts** on the Ambari dashboard; then select the host on which to install the standby Atlas Metadata Server.
2. On the Summary page of the new Atlas Metadata Server host, click **Add > Atlas Metadata Server** and add the new Atlas Metadata Server.

Ambari adds the new Atlas Metadata Server in a Stopped state.

3. Click **Atlas > Configs > Advanced**.

4. Click **Advanced application-properties** and append the `atlas.rest.address` property with a comma and the value for the new Atlas Metadata Server:
`,http(s):<host_name>:<port_number>`.

The default protocol is "http". If the `atlas.enableTLS` property is set to `true`, use "https". Also, the default HTTP port is 21000 and the default HTTPS port is 21443. These values can be overridden using the `atlas.server.http.port` and `atlas.server.https.port` properties, respectively.

5. Stop all of the Atlas Metadata Servers that are currently running.



Important

You must use the **Stop** command to stop the Atlas Metadata Servers. Do not use a **Restart** command: this attempts to first stop the newly added Atlas Server, which at this point does not contain any configurations in `/etc/atlas/conf`.

6. On the Ambari dashboard, click **Atlas > Service Actions > Start**.

Ambari automatically configures the following Atlas properties in the `/etc/atlas/conf/atlas-application.properties` file:

- `atlas.server.ids`
- `atlas.server.address.$id`
- `atlas.server.ha.enabled`

7. To refresh the configuration files, restart the following services that contain Atlas hooks:

- Hive
- Storm
- Falcon
- Sqoop
- Oozie

8. Click **Actions > Restart All Required** to restart all services that require a restart.

When you update the Atlas configuration settings in Ambari, Ambari marks the services that require restart.

9. Click **Oozie > Service Actions > Restart All** to restart Oozie along with the other services.

Apache Oozie requires a restart after an Atlas configuration update, but may not be included in the services marked as requiring restart in Ambari.

Next Steps

Review and confirm all recommended configuration changes.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

5.8. Enabling Ranger Admin High Availability

You can configure Ranger Admin high availability (HA) with or without SSL on an Ambari-managed cluster. Please note that the configuration settings used in this section are sample values. You should adjust these settings to reflect your environment (folder locations, passwords, file names, and so on).

Prerequisites**Steps**

- [HTTPD setup for HTTP](#) - Enable Ranger Admin HA with Ambari, begins at step 16.
- [HTTPD setup for HTTPS](#) - Enable Ranger Admin HA with Ambari, begins at step 14.

6. Managing Configurations

You can optimize performance of Hadoop components in your cluster by adjusting configuration settings and property values. You can also use Ambari Web to set up and manage groups and versions of configuration settings in the following ways:

- [Changing Configuration Settings \[78\]](#)
- [Manage Host Config Groups \[82\]](#)
- [Configuring Log Settings \[85\]](#)
- [Set Service Configuration Versions \[87\]](#)
- [Download Client Configuration Files \[92\]](#)

More Information

[Adjust Smart Config Settings \[79\]](#)

[Edit Specific Properties \[80\]](#)

[Review and Confirm Configuration Changes \[80\]](#)

[Restart Components \[82\]](#)

6.1. Changing Configuration Settings

You can optimize service performance using the **Configs** page for each service. The **Configs** page includes several tabs you can use to manage configuration versions, groups, settings, properties and values. You can adjust settings, called "Smart Configs" that control at a macro-level, memory allocation for each service. Adjusting Smart Configs requires related configuration settings to change throughout your cluster. Ambari prompts you to review and confirm all recommended changes and restart affected services.

Steps

1. In Ambari Web, click a service name in the service summary list on the left.
2. From the the service **Summary** page, click the **Configs** tab, then use one of the following tabs to manage configuration settings.

Use the **Configs** tab to manage configuration versions and groups.

Use the **Settings** tab to manage "Smart Configs" by adjusting the green, slider buttons.

Use the **Advanced** tab to edit specific configuration properties and values.

3. Click **Save**.

The screenshot displays the configuration interface for MapReduce memory settings. At the top, there are tabs for 'Summary' and 'Configs', along with 'Quick Links' and 'Service Actions'. Below this, a 'Group' dropdown is set to 'Default (3)' with a 'Manage Config Groups' link and a 'Filter...' input field. A version control indicator shows 'V1' by 'admin' 4 hours ago for 'HDP-2.4'. A status bar indicates 'V1' is 'admin authored on Tue, Apr 12, 2016 09:58' with 'Discard' and 'Save' buttons. The 'Settings' section is set to 'Advanced'.

The main content area is titled 'MapReduce' and contains four memory configuration sliders:

- MapReduce Framework**
 - Map Memory:** A slider ranging from 0.166 GB to 0.5 GB, with a green slider button positioned at 0.17 GB.
 - Reduce Memory:** A slider ranging from 0.166 GB to 0.5 GB, with a green slider button positioned at 0.30 GB.
 - Sort Allocation Memory:** A slider ranging from 0 MB to 2047 MB, with a green slider button positioned at 66 MB.
- MapReduce AppMaster**
 - AppMaster Memory:** A slider ranging from 0.166 GB to 0.5 GB, with a green slider button positioned at 0.17 GB.

Next Steps

Enter a description for this version that includes your current changes, review and confirm each recommended change, and then restart all affected services.

More Information

[Adjust Smart Config Settings \[79\]](#)

[Edit Specific Properties \[80\]](#)

[Review and Confirm Configuration Changes \[80\]](#)

[Restart Components \[82\]](#)

6.1.1. Adjust Smart Config Settings

Use the **Settings** tab to manage "Smart Configs" by adjusting the green, slider buttons.

Steps

1. On the **Settings** tab, click and drag a green-colored slider button to the desired value.

2. Edit values for any properties that display the Override option.

Edited values, also called *stale configs*, show an Undo option.

3. Click **Save**.

Next Steps

Enter a description for this version that includes your current changes, review and confirm each recommended change, and then restart all affected services.

More Information

[Edit Specific Properties \[80\]](#)

[Review and Confirm Configuration Changes \[80\]](#)

[Restart Components \[82\]](#)

6.1.2. Edit Specific Properties

Use the **Advanced** tab of the Configs page for each service to access groups of individual properties that affect performance of that service.

Steps

1. On a service **Configs** page, click **Advanced**.
2. On a service **Configs Advanced** page, expand a category.
3. Edit the value for any property.

Edited values, also called *stale configs*, show an Undo option.

4. Click **Save**.

Next Steps

Enter a description for this version that includes your current changes, review and confirm each recommended change, and then restart all affected services.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

[Restart Components \[82\]](#)

6.1.3. Review and Confirm Configuration Changes

When you change a configuration property value, the Ambari Stack Advisor captures and recommends changes to all related configuration properties affected by your original change. Changing a single property, a "Smart Configuration", and other actions, such as

adding or deleting a service, host or ZooKeeper Server, moving a master, or enabling high availability for a component, all require that you review and confirm related configuration changes. For example, if you increase the Minimum Container Size (Memory) setting for YARN, **Dependent Configurations** lists all recommended changes that you must review and (optionally) accept.

Dependent Configurations

Based on your configuration changes, Ambari is recommending the following dependent configuration changes. Ambari will update all checked configuration changes to the Recommended Value. Uncheck any configuration to retain the Current Value.

<input checked="" type="checkbox"/>	Property	Service	Config Group	File Name	Current Value	Recommended Value
<input checked="" type="checkbox"/>	hive.auto.convert.join.noconditionaltask.size	Hive	Default	hive-site	35791394	71582788
<input checked="" type="checkbox"/>	hive.tez.container.size	Hive	Default	hive-site	128	256
<input checked="" type="checkbox"/>	tez.runtime.io.sort.mb	Tez	Default	tez-site	33	67
<input checked="" type="checkbox"/>	tez.runtime.unordered.output.buffer.size.mb	Tez	Default	tez-site	9	19
<input checked="" type="checkbox"/>	tez.task.resource.memory.mb	Tez	Default	tez-site	128	256
<input checked="" type="checkbox"/>	mapreduce.map.java.opts	MapReduce2	Default	mapred-site	-Xmx102m	-Xmx204m
<input checked="" type="checkbox"/>	mapreduce.reduce.java.opts	MapReduce2	Default	mapred-site	-Xmx204m	-Xmx409m
<input checked="" type="checkbox"/>	yarn.app.mapreduce.am.command-opts	MapReduce2	Default	mapred-site	-Xmx102m -Dhdp.version=\${hdp.version}	-Xmx204m -Dhdp.version=\${hdp.version}
<input checked="" type="checkbox"/>	yarn.app.mapreduce.am.resource.mb	MapReduce2	Default	mapred-site	128	256

Cancel

Types of changes are highlighted in the following colors:

Value Changes Yellow

Added Properties Green

Deleted properties Red

To review and confirm changes to configuration properties:

Steps

1. In **Dependent Configurations**, for each listed property review the summary information.
2. If the change is acceptable, proceed to review the next property in the list.
3. If the change is not acceptable, click the check mark in the blue box to the right of the listed property change.

Clicking the check mark clears the box. Changes for which you clear the box are not confirmed and will not occur.

4. After reviewing all listed changes, click **OK** to confirm that all marked changes occur.

Next Steps

You must restart any components marked for restart to utilize the changes you confirmed.

More Information

[Restart Components \[82\]](#)

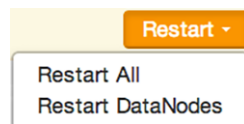
6.1.4. Restart Components

After editing and saving configuration changes, a **Restart** indicator appears next to components that require restarting to use the updated configuration values.

Steps

1. Click the indicated Components or Hosts links to view details about the requested restart.
2. Click **Restart** and then click the appropriate action.

For example, options to restart YARN components include the following:



More Information

[Review and Confirm Configuration Changes \[80\]](#)

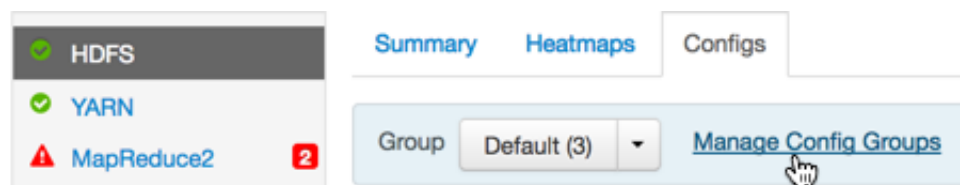
6.2. Manage Host Config Groups

Ambari initially assigns all hosts in your cluster to one default configuration group for each service you install. For example, after deploying a three-node cluster with default configuration settings, each host belongs to one configuration group that has default configuration settings for the HDFS service.

To manage Configuration Groups:

Steps

1. Click a service name, then click **Configs**.
2. In **Configs**, click **Manage Config Groups**.



To create new groups, reassign hosts, and override default settings for host components, you can use the **Manage Configuration Groups** control:

Manage HDFS Configuration Groups

You can apply different sets of HDFS configurations to groups of hosts by managing HDFS Configuration Groups and their host membership. Hosts belonging to a HDFS Configuration Group have the same set of configurations for HDFS. Each host belongs to one HDFS Configuration Group.

Default (3)

c6401.ambari.apache.org
c6402.ambari.apache.org
c6403.ambari.apache.org

+ - ⚙

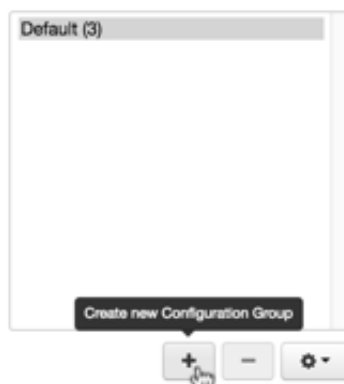
Overrides 0 properties

Cancel Save

To create a new configuration group:

Steps

1. In **Manage Config Groups**, click **Create New Configuration Group**.



2. Name and describe the group; then choose **OK**.

To add hosts to the new configuration group:

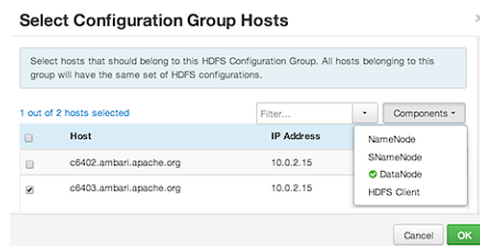
Steps

1. In **Manage Config Groups**, click a configuration group name.
2. Click **Add Hosts to selected Configuration Group**.



- Using **Select Configuration Group Hosts**, click **Components**, then click a component name from the list.

Choosing a component filters the list of hosts to only those on which that component exists for the selected service. To further filter the list of available host names, use the Filter drop-down list. The host list is filtered by IP address, by default.



- After filtering the list of hosts, click the check box next to each host that you want to include in the configuration group.
- Choose **OK**.
- In Manage Configuration Groups, choose **Save**.

To edit settings for a configuration group:

Steps

- In **Configs**, click a group name.
- Click a Config Group; then expand components to expose settings that allow Override.
- Provide a non-default value; then click **Override** or **Save**.

Configuration groups enforce configuration properties that allow override, based on installed components for the selected service and group.

4. Override prompts you to choose one of the following options:

- Either click the name of an existing configuration group (to which the property value override provided in Step 3 applies),
- Or create a new configuration group (which includes default properties, plus the property override provided in Step 3).
- Click **OK**.

5. In **Configs**, choose **Save**.

6.3. Configuring Log Settings

Ambari uses sets of properties called *Log4j properties* to control logging activities for each service running in your Hadoop cluster. Initial, default values for each property reside in a `<service_name>-log4j template` file. Log4j properties and values that limit the size and number of backup log files for each service appear above the log4j template file. To access the default Log4j settings for a service; in Ambari Web, browse to **<Service_name> > Configs > Advanced <service_name>-log4j**. For example, the Advanced yarn-log4j property group for the YARN service looks like:

The screenshot shows the configuration page for 'Advanced yarn-log4j'. At the top, there is a header bar with a user profile icon, 'V2', a checkmark, and the text 'admin authored on Thu, Feb 02, 2017 14:45'. On the right side of the header are 'Discard' and 'Save' buttons. Below the header, there is a breadcrumb trail: 'Advanced yarn-env' > 'Advanced yarn-log4j'. The main configuration area contains two input fields: 'YARN Log: backup file size' with a value of '256' and unit 'MB', and 'YARN Log: # of backup files' with a value of '20'. Below these fields is a large text area containing a log4j configuration template. The template includes comments and property definitions for job summary logging and ResourceManager app summary logging. The text area has a scrollbar on the right and a 'Save' button at the bottom right.

```
#Relative to Yarn Log Dir Prefix
yarn.log.dir=.
#
# Job Summary Appender
#
# Use following logger to send summary to separate file defined by
# hadoop.mapreduce.jobsummary.log.file rolled daily:
# hadoop.mapreduce.jobsummary.logger=INFO,JSA
#
hadoop.mapreduce.jobsummary.logger=${hadoop.root.logger}
hadoop.mapreduce.jobsummary.log.file=hadoop-mapreduce.jobsummary.log
log4j.appender.JSA=org.apache.log4j.DailyRollingFileAppender
# Set the ResourceManager summary log filename
yarn.server.resourcemanager.appsummary.log.file=hadoop-mapreduce.jobsummary.log
# Set the ResourceManager summary log level and appender
yarn.server.resourcemanager.appsummary.logger=${hadoop.root.logger}
#yarn.server.resourcemanager.appsummary.logger=INFO,RMSUMMARY

# To enable AppSummaryLogging for the RM,
```

To change the limits for the size and number of backup log files for a service:

Steps

1. Edit the values for the <service_name> backup file size and <service_name> # of backup files properties.
2. Click **Save**.

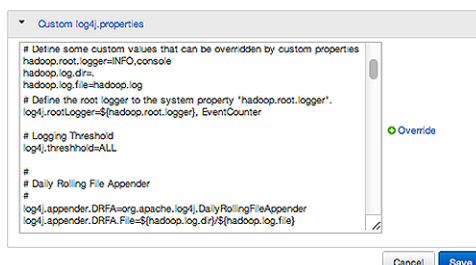
To customize Log4j settings for a service:

Steps

1. Edit values of any properties in the <service_name> **log4j** template.
2. Copy the content of the log4j template file.
3. Browse to the **custom <service_name>log4j** properties group.
4. Paste the copied content into the **custom <service_name>log4j properties**, overwriting, the default content.
5. Click **Save**.
6. Review and confirm any recommended configuration changes, as prompted.
7. Restart affected services, as prompted.

Restarting components in the service pushes the configuration properties displayed in **Custom log4j.properties** to each host running components for that service.

If you have customized logging properties that define how activities for each service are logged, you see refresh indicators next to each service name after upgrading to Ambari 1.5.0 or higher. Ensure that logging properties displayed in **Custom logj4.properties** include any customization.



Optionally, you can create configuration groups that include custom logging properties.

More Information

[Review and Confirm Configuration Changes \[80\]](#)

[Restart Components \[82\]](#)

[Adjust Smart Config Settings \[79\]](#)

[Manage Host Config Groups \[82\]](#)

6.4. Set Service Configuration Versions

Ambari enables you to manage configurations associated with a service. You can make changes to configurations, see a history of changes, compare and revert changes, and push configuration changes to the cluster hosts.

- [Basic Concepts \[87\]](#)
- [Terminology \[88\]](#)
- [Saving a Change \[88\]](#)
- [Viewing History \[89\]](#)
- [Comparing Versions \[90\]](#)
- [Reverting a Change \[91\]](#)
- [Host Config Groups \[91\]](#)

6.4.1. Basic Concepts

It is important to understand how service configurations are organized and stored in Ambari. Properties are grouped into configuration types. A set of *config types* composes the set of configurations for a service.

For example, the Hadoop Distributed File System (HDFS) service includes the `hdfs-site`, `core-site`, `hdfs-log4j`, `hadoop-env`, and `hadoop-policy` config types. If you browse to **Services > HDFS > Configs**, you can edit the configuration properties for these config types.

Ambari performs configuration versioning at the service level. Therefore, when you modify a configuration property in a service, Ambari creates a *service config version*. The following figure shows V1 and V2 of a service config version with a change to a property in Config Type A. After changing a property value in Config Type A in V1, V2 is created.



6.4.2. Terminology

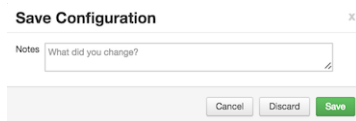
The following table lists configuration versioning terms and concepts that you should know.

configuration property	Configuration property managed by Ambari, such as NameNode heap size or replication factor
configuration type (config type)	Group of configuration properties: for example, <code>hdfs-site</code>
service configurations	Set of configuration types for a particular service: for example, <code>hdfs-site</code> and <code>core-site</code> as part of the HDFS service configuration
change notes	Optional notes to save with a service configuration change
service config version (SCV)	A particular version of a configuration for a specific service
host config group (HCG)	A set of configuration properties to apply to a specific set of hosts

6.4.3. Saving a Change

1. In **Configs**, change the value of a configuration property.
2. Choose **Save**.

3. Optionally, enter notes that describe the change:



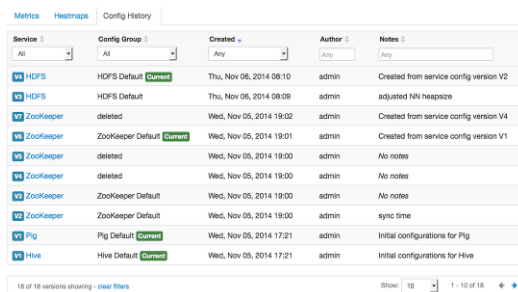
The image shows a 'Save Configuration' dialog box with a text input field for 'Notes' containing the placeholder text 'What did you change?'. Below the input field are three buttons: 'Cancel', 'Discard', and 'Save'.

4. Click **Cancel** to continue editing, **Discard** to leave the control without making any changes, or **Save** to confirm your change.

6.4.4. Viewing History

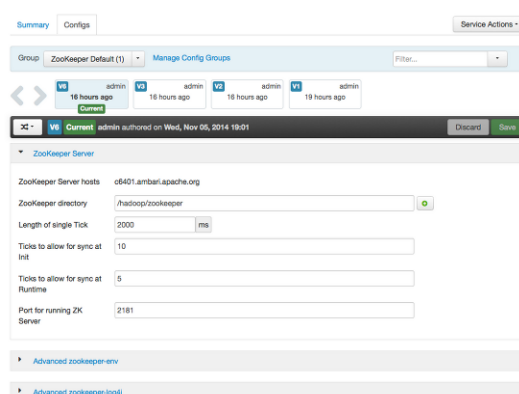
You can view your configuration change history in two places in Ambari Web: on the **Dashboard page, Config History** tab, and on each service page's **Configs** tab.

The **Dashboard > Config History** tab shows a table of all versions across all services, with each version number and the date and time the version was created. You can also see which user authored the change, and any notes about the change. Using this table, you can filter, sort, and search across versions:



Service	Config Group	Created	Author	Notes
HDFS	HDFS Default Current	Thu, Nov 06, 2014 08:10	admin	Created from service config version V2
HDFS	HDFS Default	Thu, Nov 06, 2014 08:08	admin	adjusted NN heapsize
ZooKeeper	deleted	Wed, Nov 05, 2014 19:02	admin	Created from service config version V4
ZooKeeper	ZooKeeper Default Current	Wed, Nov 05, 2014 19:01	admin	Created from service config version V1
ZooKeeper	deleted	Wed, Nov 05, 2014 19:00	admin	No notes
ZooKeeper	deleted	Wed, Nov 05, 2014 19:00	admin	No notes
ZooKeeper	ZooKeeper Default	Wed, Nov 05, 2014 19:00	admin	No notes
ZooKeeper	ZooKeeper Default	Wed, Nov 05, 2014 19:00	admin	sync time
Pig	Pig Default Current	Wed, Nov 05, 2014 17:21	admin	Initial configurations for Pig
Hive	Hive Default Current	Wed, Nov 05, 2014 17:21	admin	Initial configurations for Hive

The **Service > Configs** tab shows you only the most recent configuration change, although you can use the version scrollbar to see earlier versions. Using this tab enables you to quickly access the most recent changes to a service configuration:



The image shows the 'Configs' tab for a ZooKeeper service. It displays a list of configuration versions with a scrollbar. The current version is highlighted. Below the list, the configuration details for the current version are shown, including fields for ZooKeeper Server hosts, directory, tick length, sync ticks, and port.

Using this view, you can click any version in the scrollbar to view it, and hover your cursor over it to display an option menu that enables you to compare versions and perform a revert operation, which makes any config version that you select the current version.

V1

ZooKeeper Default

admin authored on Wed, Nov 05, 2014 16:00

Initial configurations for ZooKeeper

🔍 View

🔄 Compare

↩️ Make Current

6.4.5. Comparing Versions

When browsing the version scroll area on the **Services > Configs** tab, you can hover your cursor over a version to display options to view, compare, or revert (make current):

To compare two service configuration versions:

Steps

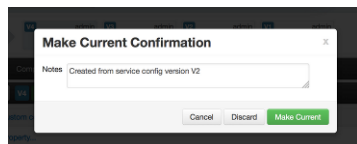
1. Navigate to a specific configuration version: for example, V6.
2. Using the version scrollbar, find the version you want to compare to V6.
For example, if you want to compare V6 to V2, find V2 in the scrollbar.
3. Hover your cursor over V2 to display the option menu, and click **Compare**.

Ambari displays a comparison of V6 to V2, with an option to revert to V2 (**Make V2 Current**). Ambari also filters the display by only **Changed properties**, under the **Filter** control:

6.4.6. Reverting a Change

You can revert to an older service configuration version by using the Make Current feature. Make Current creates a new service configuration version with the configuration properties from the version you are reverting: effectively, a clone.

After initiating the Make Current operation, you are prompted, on the **Make Current Confirmation** control, to enter notes for the clone and save it (**Make Current**). The notes text includes text about the version being cloned:

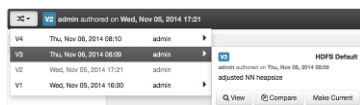


There are multiple methods to revert to a previous configuration version:

- View a specific version and click **Make V* Current**:



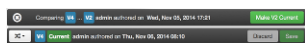
- Use the version navigation menu and click **Make Current**:



- Hover your cursor over a version in the version scrollbar and click **Make Current** :



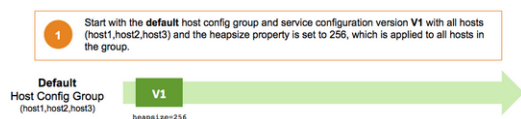
- Perform a comparison and click **Make V* Current**:

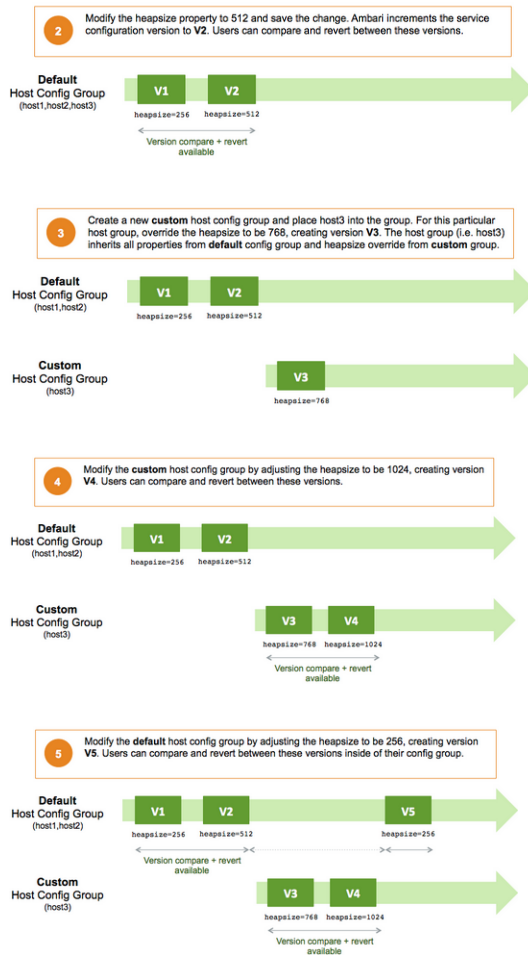


6.4.7. Host Config Groups

Service configuration versions are scoped to a host config group. For example, changes made in the default group can be compared and reverted in that config group. The same applies to custom config groups.

The following workflow shows multiple host config groups and creates service configuration versions in each config group:





6.5. Download Client Configuration Files

Client configuration files include; .xml files, env-sh scripts, and log4j properties used to configure Hadoop services. For services that include client components (most services *except* SmartSense and Ambari Metrics Service), you can download the client configuration files associated with that service. You can also download the client configuration files for your entire cluster as a single archive.

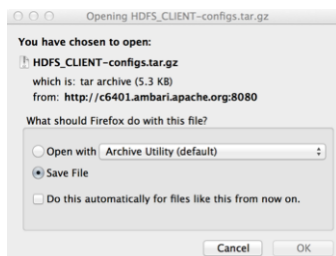
To download client configuration files for a single service:

Steps

1. In Ambari Web, browse to the service for which you want the configurations.
2. Click **Service Actions**.
3. Click **Download Client Configs**.

Your browser downloads a "tarball" archive containing only the client configuration files for that service to your default, local downloads directory.

4. If prompted to save or open the client configs bundle:

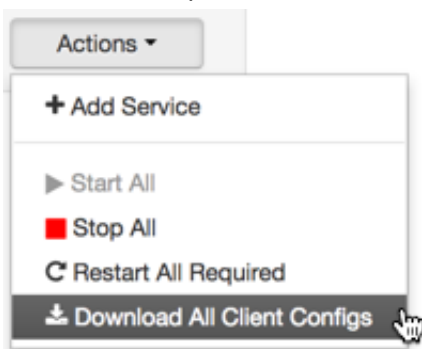


5. Click **Save File**, then click **OK**.

To download all client configuration files for your entire cluster:

Steps

1. In Ambari Web, click **Actions** at the bottom of the service summary list.



2. Click **Download Client Configs**.

Your browser downloads a "tarball" archive containing all client configuration files for your cluster to your default, local downloads directory.

7. Administering the Cluster

Using the Ambari Web **Admin** options:

any user	can view information about the stack and versions of each service added to it
Cluster administrators	can <ul style="list-style-type: none">• enable Kerberos security• regenerate required key tabs• view service user names and values• enable auto-start for services
Ambari administrators	can <ul style="list-style-type: none">• add new services to the stack• upgrade the stack to a new version, by using the link to the Ambari administration interface

Related Topics

[Hortonworks Data Platform Apache Ambari Administration](#)

[Using Stack and Versions Information \[94\]](#)

[Viewing Service Accounts \[96\]](#)

[Enabling Kerberos and Regenerating Keytabs \[97\]](#)

[Enable Service Auto-Start \[99\]](#)

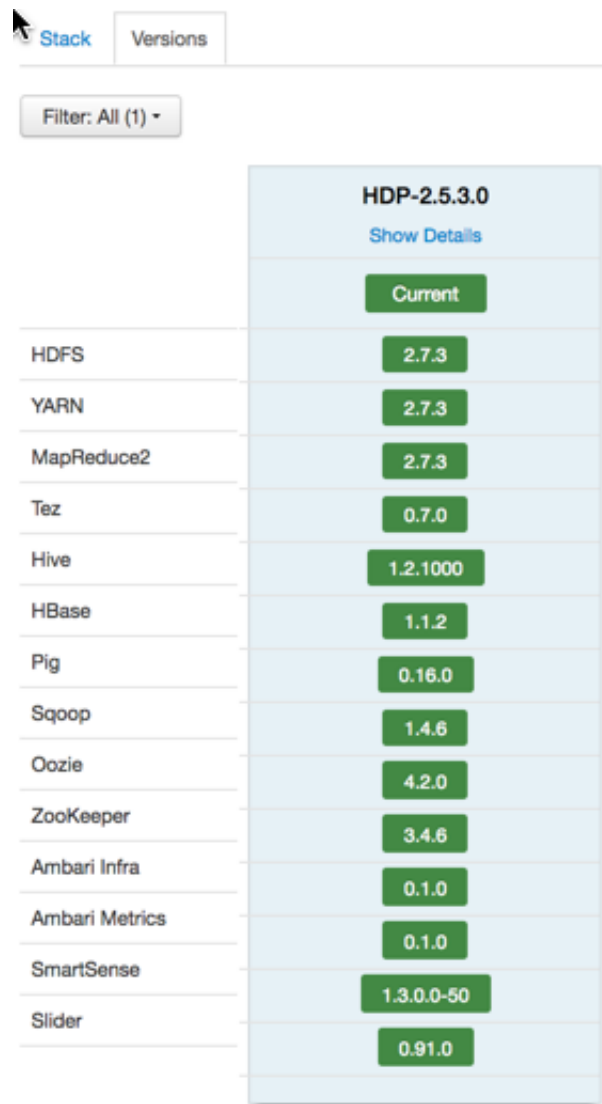
[Managing Versions](#)

7.1. Using Stack and Versions Information

The **Stack** tab includes information about the services installed and available in the cluster stack. Any user can browse the list of services. As an Ambari administrator you can also click **Add Service** to start the wizard to install each service into your cluster.

Service	Version	Status	Description
HDFS	2.7.3	Installed	Apache Hadoop Distributed File System
YARN	2.7.3	Installed	Apache Hadoop NextGen MapReduce (YARN)
MapReduce2	2.7.3	Installed	Apache Hadoop NextGen MapReduce (YARN)
Tez	0.7.0	Installed	Tez is the next generation Hadoop Query Processing framework written on top of YARN.
Hive	1.2.1000	Installed	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service
HBase	1.1.2	Installed	A Non-relational distributed database, plus Phoenix, a high performance SQL layer for low latency applications.
Pig	0.16.0	Installed	Scripting platform for analyzing large datasets
Scoop	1.4.6	Installed	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases
Oozie	4.2.0	Installed	System for workflow coordination and execution of Apache Hadoop jobs. This also includes the installation of the optional Oozie Web Console which relies on and will install the ExtJS Library.
ZooKeeper	3.4.6	Installed	Centralized service which provides highly reliable distributed coordination
Falcon	0.10.0	Add Service	Data management and processing platform
Storm	1.0.1	Add Service	Apache Hadoop Stream processing framework
Flume	1.5.2	Add Service	A distributed service for collecting, aggregating, and moving large amounts of streaming data into HDFS
Accumulo	1.7.0	Add Service	Robust, scalable, high performance distributed key/value store.
Ambari Infra	0.1.0	Installed	Core shared service used by Ambari managed components.
Ambari Metrics	0.1.0	Installed	A system for metrics collection that provides storage and retrieval capability for metrics collected from the cluster
Atlas	0.7.0	Add Service	Atlas Metadata and Governance platform
Kafka	0.10.0	Add Service	A high-throughput distributed messaging system
Knox	0.9.0	Add Service	Provides a single point of authentication and access for Apache Hadoop services in a cluster
Log Search	0.5.0	Add Service	Log aggregation, analysis, and visualization for Ambari managed services. This service is Technical Preview .
Ranger	0.6.0	Add Service	Comprehensive security for Hadoop
Ranger KMS	0.6.0	Add Service	Key Management Server
SmartSense	1.3.0.0-50	Installed	SmartSense - Hortonworks SmartSense Tool (HST) helps quickly gather configuration, metrics, logs from common HDP services that aids to quickly troubleshoot support cases and receive cluster-specific recommendations.
Spark	1.6.2	Add Service	Apache Spark is a fast and general engine for large-scale data processing.
Spark2	2.0.0	Add Service	Apache Spark 2.0 is a fast and general engine for large-scale data processing. This service is Technical Preview .
Zeppelin Notebook	0.6.0	Add Service	A web-based notebook that enables interactive data analytics. It enables you to make beautiful data-driven, interactive and collaborative documents with SQL, Scala and more.
Kerberos	1.10.3-10	Add Service	A computer network authentication protocol which works on the basis of 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.
Mahout	0.9.0	Add Service	Project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification
Slider	0.91.0	Installed	A framework for deploying, managing and monitoring existing distributed applications on YARN.

The **Versions** tab includes information about which version of software is currently installed and running in the cluster. As an Ambari administrator, you can initiate an automated cluster upgrade from this page.



The screenshot shows the 'Stack Versions' page in Ambari. At the top, there are tabs for 'Stack' and 'Versions'. Below the tabs is a filter dropdown set to 'All (1)'. The main content is a table with the following data:

HDP-2.5.3.0	
	Show Details
	Current
HDFS	2.7.3
YARN	2.7.3
MapReduce2	2.7.3
Tez	0.7.0
Hive	1.2.1000
HBase	1.1.2
Pig	0.16.0
Sqoop	1.4.6
Oozie	4.2.0
ZooKeeper	3.4.6
Ambari Infra	0.1.0
Ambari Metrics	0.1.0
SmartSense	1.3.0.0-50
Slider	0.91.0

More Information

[Adding a Service](#)

[Hortonworks Data Platform Apache Ambari Administration](#)

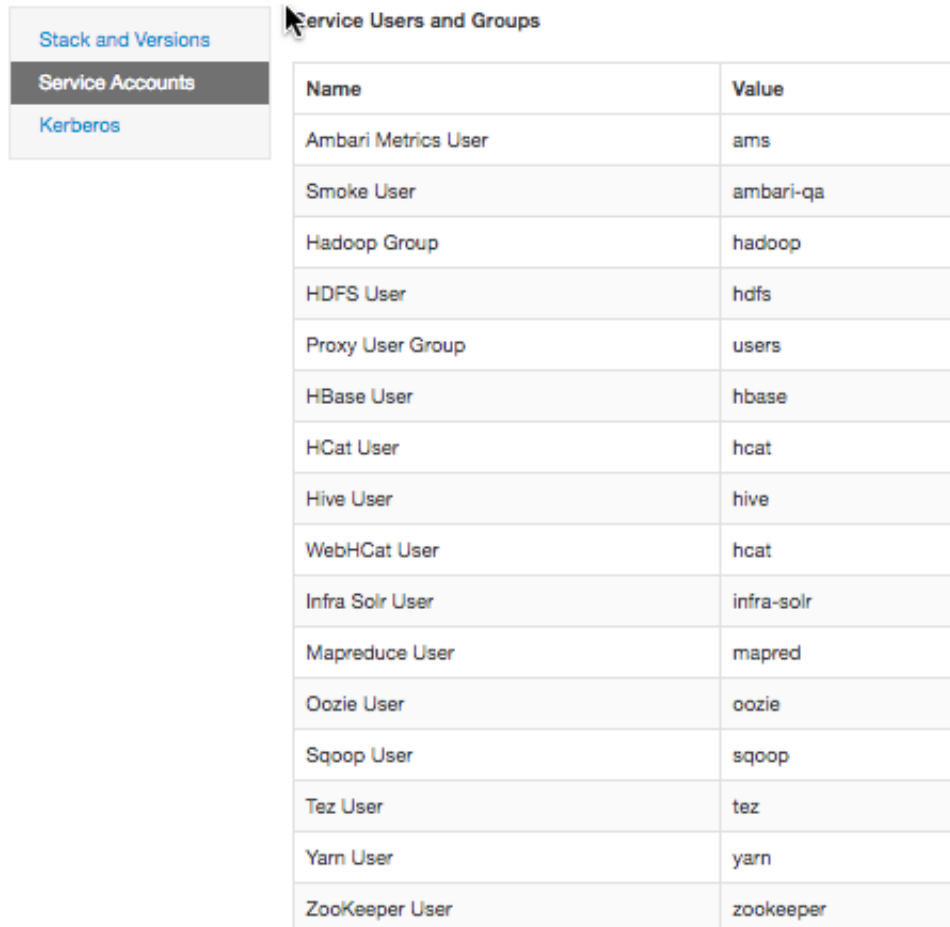
[Hortonworks Data Platform Apache Ambari Upgrade](#)

7.2. Viewing Service Accounts

As a Cluster administrator, you can view the list of Service Users and Group accounts used by the cluster services.

Steps

In Ambari Web UI > **Admin**, click **Service Accounts**.



Name	Value
Ambari Metrics User	ams
Smoke User	ambari-qa
Hadoop Group	hadoop
HDFS User	hdfs
Proxy User Group	users
HBase User	hbase
HCat User	hcat
Hive User	hive
WebHCat User	hcat
Infra Solr User	infra-solr
Mapreduce User	mapred
Oozie User	oozie
Sqoop User	sqoop
Tez User	tez
Yarn User	yarn
ZooKeeper User	zookeeper

More Information

[Defining Users and Groups for an HDP 2.x Stack](#)

7.3. Enabling Kerberos and Regenerating Keytabs

As a Cluster administrator, you can enable and manage Kerberos security in your cluster.

Prerequisites

Before enabling Kerberos in your cluster, you must prepare the cluster, as described in [Configuring Ambari and Hadoop for Kerberos](#).

Steps

In the Ambari web UI > **Admin** menu, click **Enable Kerberos** to launch the Kerberos wizard.

After Kerberos is enabled, you can regenerate key tabs and disable Kerberos from the Ambari web UI > **Admin** menu.

More Information

[Regenerate Key tabs \[98\]](#)

[Disable Kerberos \[98\]](#)

[Configuring Ambari and Hadoop for Kerberos](#)

7.3.1. Regenerate Key tabs

As a Cluster administrator, you can regenerate the key tabs required to maintain Kerberos security in your cluster.

Prerequisites

Before regenerating key tabs in your cluster:

- your cluster must be Kerberos-enabled
- you must have KDC Admin credentials

Steps

1. Browse to **Admin > Kerberos**.
2. Click **Regenerate Kerberos**.
3. Confirm your selection to proceed.
4. Ambari connects to the Kerberos Key Distribution Center (KDC) and regenerates the key tabs for the service and Ambari principals in the cluster. Optionally, you can regenerate key tabs for only those hosts that are missing key tabs: for example, hosts that were not online or available from Ambari when enabling Kerberos.
5. Restart all services.

More Information

[Disable Kerberos \[98\]](#)

[Configuring Ambari and Hadoop for Kerberos](#)

[Managing KDC Admin Credentials](#)

7.3.2. Disable Kerberos

As a Cluster administrator, you can disable Kerberos security in your cluster.

Prerequisites

Before disabling Kerberos security in your cluster, your cluster must be Kerberos-enabled.

Steps

1. Browse to **Admin > Kerberos**.

2. Click **Disable Kerberos**.
3. Confirm your selection.

Cluster services are stopped and the Ambari Kerberos security settings are reset.

4. To re-enable Kerberos, click **Enable Kerberos** and follow the wizard

More Information

[Configuring Ambari and Hadoop for Kerberos](#)

7.4. Enable Service Auto-Start

As a Cluster Administrator or Cluster Operator, you can enable each service in your stack to re-start automatically. Enabling auto-start for a service causes the ambari-agent to attempt re-starting service components in a stopped state without manual effort by a user. Auto-Start Services is enabled by default, but *only* the Ambari Metrics Collector component is set to auto-start by default.

As a first step, you should enable auto-start for the worker nodes in the core Hadoop services, the DataNode and NameNode components in YARN and HDFS, for example. You should also enable auto-start for all components in the SmartSense service. After enabling auto-start, monitor the operating status of your services on the Ambari Web dashboard. Auto-start attempts do not display as background operations. To diagnose issues with service components that fail to start, check the ambari agent logs, located at: `/var/log/ambari-agent.log` on the component host.

To manage the auto-start status for components in a service:

Steps

1. In **Auto-Start Services**, click a service name.
2. Click the grey area in the **Auto-Start Services** control of at least one component, to change its status to **Enabled**.

The screenshot shows the 'Auto-Start Services' control panel. At the top, there is a header 'Auto-Start Services' with a green 'Enabled' button and a 'Save' button. Below this is a table with columns for 'Service', 'Component', and 'Status'. A tooltip indicates '1/4 components enabled'.

Service	Component	Status
SmartSense	Activity Analyzer	Enabled
YARN	Activity Explorer	Disabled
HDFS	HST Agent	Disabled
HBase	HST Server	Disabled
MapReduce2		Enable All Disable All

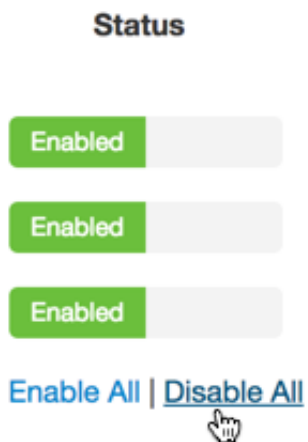
The green icon to the right of the service name indicates the percentage of components with auto-start enabled for the service.

3. To enable auto-start for all components in the service, click **Enable All**.



The green icon fills to indicate all components have auto-start enabled for the service.

4. To disable auto-start for all components in the service, click **Disable All**.



The green icon clears to indicate that all components have auto-start disabled for the service.

5. To clear all pending status changes before saving them, click **Discard**.
6. When you finish changing your auto-start status settings, click **Save**.

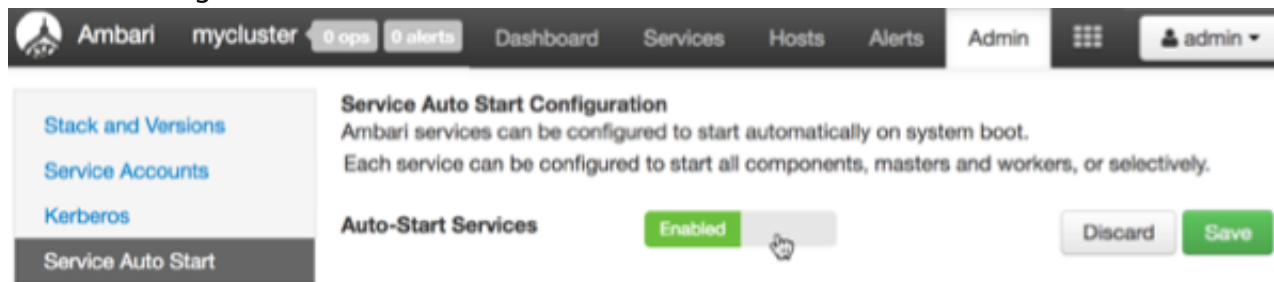
--

To disable Auto-Start Services:

Steps

1. In Ambari Web, click **Admin > Service Auto-Start**.

2. In **Service Auto Start Configuration**, click the grey area in the **Auto-Start Services** control to change its status from **Enabled** to **Disabled**.



3. Click **Save**.

More Information

[Monitoring Background Operations \[38\]](#)

8. Managing Alerts and Notifications

Ambari uses a predefined set of seven types of alerts (web, port, metric, aggregate, script, server, and recovery) for each cluster component and host. You can use these alerts to monitor cluster health and to alert other users to help you identify and troubleshoot problems. You can modify alert names, descriptions, and check intervals, and you can disable and re-enable alerts.

You can also create groups of alerts and setup notification targets for each group so that you can notify different parties interested in certain sets of alerts by using different methods.

This section provides you with the following information:

- [Understanding Alerts \[102\]](#)
- [Modifying Alerts \[104\]](#)
- [Modifying Alert Check Counts \[104\]](#)
- [Disabling and Re-enabling Alerts \[105\]](#)
- [Tables of Predefined Alerts \[105\]](#)
- [Managing Notifications \[116\]](#)
- [Creating and Editing Notifications \[116\]](#)
- [Creating or Editing Alert Groups \[118\]](#)
- [Dispatching Notifications \[119\]](#)
- [Viewing the Alert Status Log \[119\]](#)

8.1. Understanding Alerts

Ambari predefines a set of alerts that monitor the cluster components and hosts. Each alert is defined by an *alert definition*, which specifies the *alert type* check interval and thresholds. When a cluster is created or modified, Ambari reads the alert definitions and creates *alert instances* for the specific items to monitor in the cluster. For example, if your cluster includes Hadoop Distributed File System (HDFS), there is an alert definition to monitor "DataNode Process". An instance of that alert definition is created for each DataNode in the cluster.

Using Ambari Web, you can browse the list of alerts defined for your cluster by clicking the **Alerts** tab. You can search and filter alert definitions by current status, by last status change, and by the service the alert definition is associated with (among other things). You can click **alert definition name** to view details about that alert, to modify the alert properties (such as check interval and thresholds), and to see the list of alert instances associated with that alert definition.

Each alert instance reports an *alert status*, defined by severity. The most common severity levels are OK, WARNING, and CRITICAL, but there are also severities for UNKNOWN and

NONE. Alert notifications are sent when alert status changes (for example, status changes from OK to CRITICAL).

More Information

[Managing Notifications \[116\]](#)

[Tables of Predefined Alerts \[105\]](#)

8.1.1. Alert Types

Alert thresholds and the threshold units depend on the type of the alert. The following table lists the types of alerts, their possible status, and to what units thresholds can be configured if the thresholds are configurable:

WEB Alert Type	<p>WEB alerts watch a web URL on a given component; the alert status is determined based on the HTTP response code. Therefore, you cannot change which HTTP response codes determine the thresholds for WEB alerts. You can customize the response text for each threshold and the overall web connection timeout. A connection timeout is considered a CRITICAL alert. Threshold units are based on seconds.</p> <p>The response code and corresponding status for WEB alerts is as follows:</p> <ul style="list-style-type: none"> • OK status if the web URL responds with a code under 400. • WARNING status if the web URL responds with code 400 and above. • CRITICAL status if Ambari cannot connect to the web URL.
PORT Alert Type	<p>PORT alerts check the response time to connect to a given a port; the threshold units are based on seconds.</p>
METRIC Alert Type	<p>METRIC alerts check the value of a single or multiple metrics (if a calculation is performed). The metric is accessed from a URL endpoint available on a given component. A connection timeout is considered a CRITICAL alert.</p> <p>The thresholds are adjustable and the units for each threshold depend on the metric. For example, in the case of CPU utilization alerts, the unit is percentage; in the case of RPC latency alerts, the unit is milliseconds.</p>
AGGREGATE Alert Type	<p>AGGREGATE alerts aggregate the alert status as a percentage of the alert instances affected. For example, the Percent DataNode Process alert aggregates the DataNode Process alert.</p>
SCRIPT Alert Type	<p>SCRIPT alerts execute a script that determines status such as OK, WARNING, or CRITICAL. You can customize the response</p>

	text and values for the properties and thresholds for the SCRIPT alert.
SERVER Alert Type	SERVER alerts execute a server-side runnable class that determines the alert status such as OK, WARNING, or CRITICAL.
RECOVERY Alert Type	RECOVERY alerts are handled by the Ambari Agents that are monitoring for process restarts. Alert status OK, WARNING, and CRITICAL are based on the number of times a process is restarted automatically. This is useful to know when processes are terminating and Ambari is automatically restarting.

8.2. Modifying Alerts

General properties for an alert include name, description, check interval, and thresholds.

The check interval defines the frequency with which Ambari checks alert status. For example, "1 minute" value means that Ambari checks the alert status every minute.

The configuration options for thresholds depend on the alert type.

To modify the general properties of alerts:

Steps

1. Browse to the **Alerts** section in Ambari Web.
2. Find the alert definition and click to view the definition details.
3. Click **Edit** to modify the name, description, check interval, and thresholds (as applicable).
4. Click **Save**.
5. Changes take effect on all alert instances at the next check interval.

More Information

[Alert Types \[103\]](#)

8.3. Modifying Alert Check Counts

Ambari enables you to set the number of alert checks to perform before dispatching a notification. If the alert state changes during a check, Ambari attempts to check the condition a specified number of times (the *check count*) before dispatching a notification.

Alert check counts are not applicable to AGGREGATE alert types. A state change for an AGGREGATE alert results in a notification dispatch.

If your environment experiences transient issues resulting in false alerts, you can increase the check count. In this case, the alert state change is still recorded, but as a SOFT state change. If the alert condition is still triggered after the specified number of checks, the state change is then considered HARD, and notifications are sent.

You generally want to set the check count value globally for all alerts, but you can also override that value for individual alerts if a specific alert or alerts is experiencing transient issues.

To modify the global alert check count:

Steps

1. Browse to the Alerts section in Ambari Web.
2. In the Ambari Web, **Actions** menu, click **Manage Alert Settings**.
3. Update the **Check Count** value.
4. Click **Save**.

Changes made to the global alert check count might require a few seconds to appear in the Ambari UI for individual alerts.

To override the global alert check count for individual alerts:

Steps

1. Browse to the Alerts section in Ambari Web.
2. Select the alert for which you want to set a specific **Check Count**.
3. On the right, click the Edit icon next to the Check Count property.
4. Update the Check Count value.
5. Click **Save**.

More Information

[Managing Notifications \[116\]](#)

8.4. Disabling and Re-enabling Alerts

You can optionally disable alerts. When an alert is disabled, no alert instances are in effect and Ambari will no longer perform the checks for the alert. Therefore, no alert status changes will be recorded and no notifications (i.e. no emails or SNMP traps) will be dispatched.

1. Browse to the Alerts section in **Ambari Web**.
2. Find the alert definition. Click the **Enabled** or **Disabled** text to enable/disable the alert.
3. Alternatively, you can click on the alert to view the definition details and click **Enabled** or **Disabled** to enable/disable the alert.
4. You will be prompted to confirm enable/disable.

8.5. Tables of Predefined Alerts

- [HDFS Service Alerts \[106\]](#)
- [HDFS HA Alerts \[109\]](#)

- [NameNode HA Alerts \[110\]](#)
- [YARN Alerts \[111\]](#)
- [MapReduce2 Alerts \[112\]](#)
- [HBase Service Alerts \[112\]](#)
- [Hive Alerts \[113\]](#)
- [Oozie Alerts \[114\]](#)
- [ZooKeeper Alerts \[114\]](#)
- [Ambari Alerts \[114\]](#)
- [Ambari Metrics Alerts \[115\]](#)
- [SmartSense Alerts \[116\]](#)

8.5.1. HDFS Service Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
NameNode Blocks Health	METRIC	This service-level alert is triggered if the number of corrupt or missing blocks exceeds the configured critical threshold.	<p>Some DataNodes are down and the replicas that are missing blocks are only on those DataNodes.</p> <p>The corrupt or missing blocks are from files with a replication factor of 1. New replicas cannot be created because the only replica of the block is missing.</p>	<p>For critical data, use a replication factor of 3.</p> <p>Bring up the failed DataNodes with missing or corrupt blocks.</p> <p>Identify the files associated with the missing or corrupt blocks by running the Hadoop <code>fsck</code> command.</p> <p>Delete the corrupt files and recover them from backup, if one exists.</p>
NFS Gateway Process	PORT	This host-level alert is triggered if the NFS Gateway process cannot be confirmed as active.	NFS Gateway is down.	Check for a non-operating NFS Gateway in Ambari Web.
DataNode Storage	METRIC	This host-level alert is triggered if storage capacity is full on the DataNode (90% critical). It checks the DataNode JMX Servlet for the Capacity and Remaining properties.	<p>Cluster storage is full.</p> <p>If cluster storage is not full, DataNode is full.</p>	<p>If the cluster still has storage, use the load balancer to distribute the data to relatively less-used DataNodes.</p> <p>If the cluster is full, delete unnecessary data or add additional storage by adding either more DataNodes or more or larger disks to the DataNodes. After adding more storage, run the load balancer.</p>
DataNode Process	PORT	This host-level alert is triggered if the individual DataNode processes cannot be established to be up and listening on the network for the configured critical threshold, in seconds.	<p>DataNode process is down or not responding.</p> <p>DataNode are not down but is not listening to the correct network port/address.</p>	<p>Check for non-operating DataNodes in Ambari Web.</p> <p>Check for any errors in the DataNode logs (<code>/var/log/hadoop/hdfs</code>) and restart the DataNode, if necessary.</p>

Alert	Alert Type	Description	Potential Causes	Possible Remedies
				Run the <code>netstat-tuplpn</code> command to check if the DataNode process is bound to the correct network port.
DataNode Web UI	WEB	This host-level alert is triggered if the DataNode web UI is unreachable.	The DataNode process is not running.	Check whether the DataNode process is running.
NameNode Host CPU Utilization	METRIC	This host-level alert is triggered if CPU utilization of the NameNode exceeds certain thresholds (200% warning, 250% critical). It checks the NameNode JMX Servlet for the SystemCPULoad property. This information is available only if you are running JDK 1.7.	Unusually high CPU utilization might be caused by a very unusual job or query workload, but this is generally the sign of an issue in the daemon.	Use the <code>top</code> command to determine which processes are consuming excess CPU. Reset the offending process.
NameNode Web UI	WEB	This host-level alert is triggered if the NameNode web UI is unreachable.	The NameNode process is not running.	Check whether the NameNode process is running.
Percent DataNodes with Available Space	AGGREGATE	This service-level alert is triggered if the storage is full on a certain percentage of DataNodes (10% warn, 30% critical).	Cluster storage is full. If cluster storage is not full, DataNode is full.	If the cluster still has storage, use the load balancer to distribute the data to relatively less-used DataNodes. If the cluster is full, delete unnecessary data or increase storage by adding either more DataNodes or more or larger disks to the DataNodes. After adding more storage, run the load balancer.
Percent DataNodes Available	AGGREGATE	This alert is triggered if the number of non-operating DataNodes in the cluster is greater than the configured critical threshold. This aggregates the DataNode process alert.	DataNodes are down. DataNodes are not down but are not listening to the correct network port/address.	Check for non-operating DataNodes in Ambari Web. Check for any errors in the DataNode logs (<code>/var/log/hadoop/hdfs</code>) and restart the DataNode hosts/processes. Run the <code>netstat-tuplpn</code> command to check if the DataNode process is bound to the correct network port.
NameNode RPC Latency	METRIC	This host-level alert is triggered if the NameNode operations RPC latency exceeds the configured critical threshold. Typically an increase in the RPC processing time increases the RPC queue length, causing the average queue wait time to increase for NameNode operations.	A job or an application is performing too many NameNode operations.	Review the job or the application for potential bugs causing it to perform too many NameNode operations.
NameNode Last Checkpoint	SCRIPT	This alert will trigger if the last time that the NameNode performed a checkpoint was too long ago or if the number	Too much time elapsed since last NameNode checkpoint.	Set NameNode checkpoint. Review threshold for uncommitted transactions.

Alert	Alert Type	Description	Potential Causes	Possible Remedies
		of uncommitted transactions is beyond a certain threshold.	Uncommitted transactions beyond threshold.	
Secondary NameNode Process	WEB	If the Secondary NameNode process cannot be confirmed to be up and listening on the network. This alert is not applicable when NameNode HA is configured.	The Secondary NameNode is not running.	Check that the Secondary DataNode process is running.
NameNode Directory Status	METRIC	This alert checks if the NameNode NameDirStatus metric reports a failed directory.	One or more of the directories are reporting as not healthy.	Check the NameNode UI for information about unhealthy directories.
HDFS Capacity Utilization	METRIC	This service-level alert is triggered if the HDFS capacity utilization exceeds the configured critical threshold (80% warn, 90% critical). It checks the NameNode JMX Servlet for the CapacityUsed and CapacityRemaining properties.	Cluster storage is full.	Delete unnecessary data. Archive unused data. Add more DataNodes. Add more or larger disks to the DataNodes. After adding more storage, run the load balancer.
DataNode Health Summary	METRIC	This service-level alert is triggered if there are unhealthy DataNodes.	A DataNode is in an unhealthy state.	Check the NameNode UI for the list of non-operating DataNodes.
HDFS Pending Deletion Blocks	METRIC	This service-level alert is triggered if the number of blocks pending deletion in HDFS exceeds the configured warning and critical thresholds. It checks the NameNode JMX Servlet for the PendingDeletionBlock property.	Large number of blocks are pending deletion.	
HDFS Upgrade Finalized State	SCRIPT	This service-level alert is triggered if HDFS is not in the finalized state.	The HDFS upgrade is not finalized.	Finalize any upgrade you have in process.
DataNode Unmounted Data Dir	SCRIPT	This host-level alert is triggered if one of the data directories on a host was previously on a mount point and became unmounted.	If the mount history file does not exist, then report an error if a host has one or more mounted data directories as well as one or more unmounted data directories on the root partition. This may indicate that a data directory is writing to the root partition, which is undesirable.	Check the data directories to confirm they are mounted as expected.
DataNode Heap Usage	METRIC	This host-level alert is triggered if heap usage goes past thresholds on the DataNode. It checks the DataNode JMXServlet for the MemHeapUsedM and MemHeapMaxM properties. The threshold values are percentages.		
NameNode Client RPC Queue Latency	SCRIPT	This service-level alert is triggered if the deviation of RPC queue latency on client port has grown beyond the		

Alert	Alert Type	Description	Potential Causes	Possible Remedies
		specified threshold within an given period. This alert will monitor Hourly and Daily periods.		
NameNode Client RPC Processing Latency	SCRIPT	This service-level alert is triggered if the deviation of RPC latency on client port has grown beyond the specified threshold within a given period. This alert will monitor Hourly and Daily periods.		
NameNode Service RPC Queue Latency	SCRIPT	This service-level alert is triggered if the deviation of RPC latency on the DataNode port has grown beyond the specified threshold within a given period. This alert will monitor Hourly and Daily periods.		
NameNode Service RPC Processing Latency	SCRIPT	This service-level alert is triggered if the deviation of RPC latency on the DataNode port has grown beyond the specified threshold within a given period. This alert will monitor Hourly and Daily periods.		
HDFS Storage Capacity Usage	SCRIPT	This service-level alert is triggered if the increase in storage capacity usage deviation has grown beyond the specified threshold within a given period. This alert will monitor Daily and Weekly periods.		
NameNode Heap Usage	SCRIPT	This service-level alert is triggered if the NameNode heap usage deviation has grown beyond the specified threshold within a given period. This alert will monitor Daily and Weekly periods.		

8.5.2. HDFS HA Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
JournalNode Web UI	WEB	This host-level alert is triggered if the individual JournalNode process cannot be established to be up and listening on the network for the configured critical threshold, given in seconds.	The JournalNode process is down or not responding. The JournalNode is not down but is not listening to the correct network port/address.	Check if the JournalNode process is running.
NameNode High Availability Health	SCRIPT	This service-level alert is triggered if either the Active NameNode or Standby NameNode are not running.	The Active, Standby or both NameNode processes are down.	On each host running NameNode, check for any errors in the logs (/var/log/hadoop/hdfs/) and restart the NameNode host/process using Ambari Web.

Alert	Alert Type	Description	Potential Causes	Possible Remedies
				On each host running NameNode, run the <code>netstat -tuplpn</code> command to check if the NameNode process is bound to the correct network port.
Percent JournalNodes Available	AGGREGATE	This service-level alert is triggered if the number of down JournalNodes in the cluster is greater than the configured critical threshold (33% warn, 50% crit). It aggregates the results of JournalNode process checks.	JournalNodes are down. JournalNodes are not down but are not listening to the correct network port/address.	Check for dead JournalNodes in Ambari Web.
ZooKeeper Failover Controller Process	PORT	This alert is triggered if the ZooKeeper Failover Controller process cannot be confirmed to be up and listening on the network.	The ZKFC process is down or not responding.	Check if the ZKFC process is running.

8.5.3. NameNode HA Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
JournalNode Process	WEB	This host-level alert is triggered if the individual JournalNode process cannot be established to be up and listening on the network for the configured critical threshold, given in seconds.	The JournalNode process is down or not responding. The JournalNode is not down but is not listening to the correct network port/address.	Check if the JournalNode process is running.
NameNode High Availability Health	SCRIPT	This service-level alert is triggered if either the Active NameNode or Standby NameNode are not running.	The Active, Standby or both NameNode processes are down.	On each host running NameNode, check for any errors in the logs (<code>/var/log/hadoop/hdfs/</code>) and restart the NameNode host/process using Ambari Web. On each host running NameNode, run the <code>netstat -tuplpn</code> command to check if the NameNode process is bound to the correct network port.
Percent JournalNodes Available	AGGREGATE	This service-level alert is triggered if the number of down JournalNodes in the cluster is greater than the configured critical threshold (33% warn, 50% crit). It aggregates the results of JournalNode process checks.	JournalNodes are down. JournalNodes are not down but are not listening to the correct network port/address.	Check for non-operating JournalNodes in Ambari Web.
ZooKeeper Failover Controller Process	PORT	This alert is triggered if the ZooKeeper Failover Controller process cannot be confirmed to be up and listening on the network.	The ZKFC process is down or not responding.	Check if the ZKFC process is running.

8.5.4. YARN Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
App Timeline Web UI	WEB	This host-level alert is triggered if the App Timeline Server Web UI is unreachable.	The App Timeline Server is down. App Timeline Service is not down but is not listening to the correct network port/address.	Check for non-operating App Timeline Server in Ambari Web.
Percent NodeManagers Available	AGGREGATE	This alert is triggered if the number of down NodeManagers in the cluster is greater than the configured critical threshold. It aggregates the results of DataNode process alert checks.	NodeManagers are down. NodeManagers are not down but are not listening to the correct network port/address.	Check for non-operating NodeManagers. Check for any errors in the NodeManager logs (/var/log/hadoop/yarn) and restart the NodeManagers hosts/processes, as necessary. Run the <pre>netstat -tupln</pre> command to check if the NodeManager process is bound to the correct network port.
ResourceManager Web UI	WEB	This host-level alert is triggered if the ResourceManager Web UI is unreachable.	The ResourceManager process is not running.	Check if the ResourceManager process is running.
ResourceManager RPC Latency	METRIC	This host-level alert is triggered if the ResourceManager operations RPC latency exceeds the configured critical threshold. Typically an increase in the RPC processing time increases the RPC queue length, causing the average queue wait time to increase for ResourceManager operations.	A job or an application is performing too many ResourceManager operations.	Review the job or the application for potential bugs causing it to perform too many ResourceManager operations.
ResourceManager CPU Utilization	METRIC	This host-level alert is triggered if CPU utilization of the ResourceManager exceeds certain thresholds (200% warning, 250% critical). It checks the ResourceManager JMX Servlet for the SystemCPULoad property. This information is only available if you are running JDK 1.7.	Unusually high CPU utilization: Can be caused by a very unusual job/query workload, but this is generally the sign of an issue in the daemon.	Use the <pre>top</pre> command to determine which processes are consuming excess CPU. Reset the offending process.
NodeManager Web UI	WEB	This host-level alert is triggered if the NodeManager process cannot be established to be up and listening on the network for the configured critical threshold, given in seconds.	NodeManager process is down or not responding. NodeManager is not down but is not listening to the correct network port/address.	Check if the NodeManager is running. Check for any errors in the NodeManager logs (/var/log/hadoop/yarn) and restart the NodeManager, if necessary.
NodeManager Health Summary	SCRIPT	This host-level alert checks the node health property available from the NodeManager component.	NodeManager Health Check script reports issues or is not configured.	Check in the NodeManager logs (/var/log/hadoop/yarn) for health check errors and

Alert	Alert Type	Description	Potential Causes	Possible Remedies
				restart the NodeManager, and restart if necessary. Check in the ResourceManager UI logs (/var/log/hadoop/yarn) for health check errors.
NodeManager Health	SCRIPT	This host-level alert checks the nodeHealthy property available from the NodeManager component.	The NodeManager process is down or not responding.	Check in the NodeManager logs (/var/log/hadoop/yarn) for health check errors and restart the NodeManager, and restart if necessary.

8.5.5. MapReduce2 Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
History Server Web UI	WEB	This host-level alert is triggered if the HistoryServer Web UI is unreachable.	The HistoryServer process is not running.	Check if the HistoryServer process is running.
History Server RPC latency	METRIC	This host-level alert is triggered if the HistoryServer operations RPC latency exceeds the configured critical threshold. Typically an increase in the RPC processing time increases the RPC queue length, causing the average queue wait time to increase for NameNode operations.	A job or an application is performing too many HistoryServer operations.	Review the job or the application for potential bugs causing it to perform too many HistoryServer operations.
History Server CPU Utilization	METRIC	This host-level alert is triggered if the percent of CPU utilization on the HistoryServer exceeds the configured critical threshold.	Unusually high CPU utilization: Can be caused by a very unusual job/query workload, but this is generally the sign of an issue in the daemon.	Use the <code>top</code> command to determine which processes are consuming excess CPU. Reset the offending process.
History Server Process	PORT	This host-level alert is triggered if the HistoryServer process cannot be established to be up and listening on the network for the configured critical threshold, given in seconds.	HistoryServer process is down or not responding. HistoryServer is not down but is not listening to the correct network port/address.	Check the HistoryServer is running. Check for any errors in the HistoryServer logs (/var/log/hadoop/mapred) and restart the HistoryServer, if necessary.

8.5.6. HBase Service Alerts

Alert	Description	Potential Causes	Possible Remedies
Percent RegionServers Available	This service-level alert is triggered if the configured percentage of Region Server processes cannot be determined to be up and listening on the network for the configured critical threshold. The default setting is 10% to produce a WARN alert and 30% to produce a CRITICAL alert. It aggregates the results of RegionServer process down checks.	Misconfiguration or less-than-ideal configuration caused the RegionServers to crash. Cascading failures brought on by some workload caused the RegionServers to crash. The RegionServers shut themselves own because there were problems	Check the dependent services to make sure they are operating correctly. Look at the RegionServer log files (usually /var/log/hbase/*.log) for further information. If the failure was associated with a particular workload, try to understand the workload better.

Alert	Description	Potential Causes	Possible Remedies
		<p>in the dependent services, ZooKeeper or HDFS.</p> <p>GC paused the RegionServer for too long and the RegionServers lost contact with Zookeeper.</p>	Restart the RegionServers.
HBase Master Process	This alert is triggered if the HBase master processes cannot be confirmed to be up and listening on the network for the configured critical threshold, given in seconds.	<p>The HBase master process is down.</p> <p>The HBase master has shut itself down because there were problems in the dependent services, ZooKeeper or HDFS.</p>	<p>Check the dependent services.</p> <p>Look at the master log files (usually /var/log/hbase/*.log) for further information.</p> <p>Look at the configuration files (/etc/hbase/conf).</p> <p>Restart the master.</p>
HBase Master CPU Utilization	This host-level alert is triggered if CPU utilization of the HBase Master exceeds certain thresholds (200% warning, 250% critical). It checks the HBase Master JMX Servlet for the SystemCPULoad property. This information is only available if you are running JDK 1.7.	Unusually high CPU utilization: Can be caused by a very unusual job/query workload, but this is generally the sign of an issue in the daemon.	<p>Use the <code>top</code> command to determine which processes are consuming excess CPU</p> <p>Reset the offending process.</p>
RegionServers Health Summary	This service-level alert is triggered if there are unhealthy RegionServers.	<p>The RegionServer process is down on the host.</p> <p>The RegionServer process is up and running but not listening on the correct network port (default 60030).</p>	Check for dead RegionServer in Ambari Web.
HBase RegionServer Process	This host-level alert is triggered if the RegionServer processes cannot be confirmed to be up and listening on the network for the configured critical threshold, given in seconds.	<p>The RegionServer process is down on the host.</p> <p>The RegionServer process is up and running but not listening on the correct network port (default 60030).</p>	<p>Check for any errors in the logs (/var/log/hbase/) and restart the RegionServer process using Ambari Web.</p> <p>Run the <code>netstat-tuplpn</code> command to check if the RegionServer process is bound to the correct network port.</p>

8.5.7. Hive Alerts

Alert	Description	Potential Causes	Possible Remedies
HiveServer2 Process	This host-level alert is triggered if the HiveServer cannot be determined to be up and responding to client requests.	<p>HiveServer2 process is not running.</p> <p>HiveServer2 process is not responding.</p>	Using Ambari Web, check status of HiveServer2 component. Stop and then restart.
HiveMetastore Process	This host-level alert is triggered if the Hive Metastore process cannot be determined to be up and listening on the network for the configured critical threshold, given in seconds.	<p>The Hive Metastore service is down.</p> <p>The database used by the Hive Metastore is down.</p> <p>The Hive Metastore host is not reachable over the network.</p>	Using Ambari Web, stop the Hive service and then restart it.
WebHCat Server Status	This host-level alert is triggered if the WebHCat server cannot be determined to be up and responding to client requests.	<p>The WebHCat server is down.</p> <p>The WebHCat server is hung and not responding.</p>	Restart the WebHCat server using Ambari Web.

Alert	Description	Potential Causes	Possible Remedies
		The WebHCat server is not reachable over the network.	

8.5.8. Oozie Alerts

Alert	Description	Potential Causes	Possible Remedies
Oozie Server Web UI	This host-level alert is triggered if the Oozie server Web UI is unreachable.	The Oozie server is down. Oozie Server is not down but is not listening to the correct network port/address.	Check for dead Oozie Server in Ambari Web.
Oozie Server Status	This host-level alert is triggered if the Oozie server cannot be determined to be up and responding to client requests.	The Oozie server is down. The Oozie server is hung and not responding. The Oozie server is not reachable over the network.	Restart the Oozie service using Ambari Web.

8.5.9. ZooKeeper Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
Percent ZooKeeper Servers Available	AGGREGATE	This service-level alert is triggered if the configured percentage of ZooKeeper processes cannot be determined to be up and listening on the network for the configured critical threshold, given in seconds. It aggregates the results of ZooKeeper process checks.	The majority of your ZooKeeper servers are down and not responding.	Check the dependent services to make sure they are operating correctly. Check the ZooKeeper logs (/var/log/hadoop/zookeeper.log) for further information. If the failure was associated with a particular workload, try to understand the workload better. Restart the ZooKeeper servers from the Ambari UI.
ZooKeeper Server Process	PORT	This host-level alert is triggered if the ZooKeeper server process cannot be determined to be up and listening on the network for the configured critical threshold, given in seconds.	The ZooKeeper server process is down on the host. The ZooKeeper server process is up and running but not listening on the correct network port (default 2181).	Check for any errors in the ZooKeeper logs (/var/log/hbase/) and restart the ZooKeeper process using Ambari Web. Run the <code>netstat -tupl</code> command to check if the ZooKeeper server process is bound to the correct network port.

8.5.10. Ambari Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
Host Disk Usage	SCRIPT	This host-level alert is triggered if the amount of	The amount of free disk space left is low.	Check host for disk space to free or add more storage.

Alert	Alert Type	Description	Potential Causes	Possible Remedies
		disk space used on a host goes above specific thresholds (50% warn, 80% crit).		
Ambari Agent Heartbeat	SERVER	This alert is triggered if the server has lost contact with an agent.	Ambari Server host is unreachable from Agent host Ambari Agent is not running	Check connection from Agent host to Ambari Server Check Agent is running
Ambari Server Alerts	SERVER	This alert is triggered if the server detects that there are alerts which have not run in a timely manner	Agents are not reporting alert status Agents are not running	Check that all Agents are running and heartbeating
Ambari Server Performance	SERVER	This alert is triggered if the Ambari Server detects that there is a potential performance problem with Ambari.	This type of issue can arise for many reasons, but is typically attributed to slow database queries and host resource exhaustion.	Check your Ambari Server database connection and database activity. Check your Ambari Server host for resource exhaustion such as memory.

8.5.11. Ambari Metrics Alerts

Alert	Description	Potential Causes	Possible Remedies
Metrics Collector Process	This alert is triggered if the Metrics Collector cannot be confirmed to be up and listening on the configured port for number of seconds equal to threshold.	The Metrics Collector process is not running.	Check the Metrics Collector is running.
Metrics Collector – ZooKeeper Server Process	This host-level alert is triggered if the Metrics Collector ZooKeeper Server Process cannot be determined to be up and listening on the network.	The Metrics Collector process is not running.	Check the Metrics Collector is running.
Metrics Collector – HBase Master Process	This alert is triggered if the Metrics Collector HBase Master Processes cannot be confirmed to be up and listening on the network for the configured critical threshold, given in seconds.	The Metrics Collector process is not running.	Check the Metrics Collector is running.
Metrics Collector – HBase Master CPU Utilization	This host-level alert is triggered if CPU utilization of the Metrics Collector exceeds certain thresholds.	Unusually high CPU utilization generally the sign of an issue in the daemon configuration.	Tune the Ambari Metrics Collector.
Metrics Monitor Status	This host-level alert is triggered if the Metrics Monitor process cannot be confirmed to be up and running on the network.	The Metrics Monitor is down.	Check whether the Metrics Monitor is running on the given host.
Percent Metrics Monitors Available	This is an AGGREGATE alert of the Metrics Monitor Status.	Metrics Monitors are down.	Check the Metrics Monitors are running.
Metrics Collector - Auto-Restart Status	This alert is triggered if the Metrics Collector has been auto-started for number of times equal to start threshold in a 1 hour timeframe. By default if restarted 2 times in an hour, you will receive a Warning alert. If restarted 4 or more times in an hour, you will receive a Critical alert.	The Metrics Collector is running but is unstable and causing restarts. This could be due to improper tuning.	Tune the Ambari Metrics Collector.

Alert	Description	Potential Causes	Possible Remedies
Percent Metrics Monitors Available	This is an AGGREGATE alert of the Metrics Monitor Status.	Metrics Monitors are down.	Check the Metrics Monitors.
Grafana Web UI	This host-level alert is triggered if the AMS Grafana Web UI is unreachable.	Grafana process is not running.	Check whether the Grafana process is running. Restart if it has gone down.

More Information

[Tuning Ambari Metrics](#)

8.5.12. SmartSense Alerts

Alert	Description	Potential Causes	Possible Remedies
SmartSense Server Process	This alert is triggered if the HST server process cannot be confirmed to be up and listening on the network for the configured critical threshold, given in seconds.	HST server is not running.	Start HST server process. If startup fails, check the hst-server.log.
SmartSense Bundle Capture Failure	This alert is triggered if the last triggered SmartSense bundle is failed or timed out.	Some nodes are timed out during capture or fail during data capture. It could also be because upload to Hortonworks fails.	From the "Bundles" page check the status of bundle. Next, check which agents have failed or timed out, and review their logs. You can also initiate a new capture.
SmartSense Long Running Bundle	This alert is triggered if the SmartSense in-progress bundle has possibility of not completing successfully on time.	Service components that are getting collected may not be running. Or some agents may be timing out during data collection/upload.	Restart the services that are not running. Force-complete the bundle and start a new capture.
SmartSense Gateway Status	This alert is triggered if the SmartSense Gateway server process is enabled but is unable to reach.	SmartSense Gateway is not running.	Start the gateway. If gateway start fails, review hst-gateway.log

8.6. Managing Notifications

Using alert groups and notifications enables you to create groups of alerts and set up notification targets for each group in such a way that you can notify different parties interested in certain sets of alerts by using different methods. For example, you might want your Hadoop Operations team to receive all alerts by email, regardless of status, while at the same time you want your System Administration team to receive only RPC and CPU-related alerts that are in Critical state, and only by simple network management protocol (SNMP).

To achieve these different results, you can have one alert notification that manages email for all alert groups for all severity levels, and a different alert notification group that manages SNMP on critical-severity alerts for an alert group that contains the RPC and CPU alerts.

8.7. Creating and Editing Notifications

To create or edit alert notifications:

Steps

1. In Ambari Web, click **Alerts**.
2. On the **Alerts** page, click the **Actions** menu, then click **Manage Notifications**.
3. In **Manage Alert Notifications**, click + to create a new alert notification.

In **Create Alert Notification**,

- In **Name**, enter a name for the notification
 - In **Groups**, click **All** or **Custom** to assign the notification to every or set of groups that you specify
 - In **Description**, type a phrase that describes the notification
 - In **Method**, click **EMAIL**, **SNMP** (for MIB-based) or **Custom SNMP** as the method by which Ambari server handles delivery of this notification.
4. Complete the fields for the notification method you selected.
 - For email notification, provide information about your SMTP infrastructure, such as SMTP server, port, to and from addresses, and whether authentication is required to relay messages through the server.

You can add custom properties to the SMTP configuration based on Javamail SMTP options.

Email To	A comma-separated list of one or more email addresses to which to send the alert email
SMTP Server	The FQDN or IP address of the SMTP server to use to relay the alert email
SMTP Port	The SMTP port on the SMTP server
Email From	A single email address to be the originator of the alert email
Use Authentication	Determine whether your SMTP server requires authentication before it can relay messages. Be sure to also provide the username and password credentials

- For MIB-based SNMP notification, provide the version, community, host, and port to which the SNMP trap should be sent.:

Version	SNMPv1 or SNMPv2c, depending on the network environment
Hosts	A comma-separated list of one or more host FQDNs to which to send the trap
Port	The port on which a process is listening for SNMP traps

For SNMP notifications, Ambari uses a "MIB", a text file manifest of alert definitions, to transfer alert information from cluster operations to the alerting infrastructure. A MIB summarizes how object IDs map to objects or attributes.

For example, MIB file content looks like this:

```

apacheAmbariAlertEntry OBJECT-TYPE
    SYNTAX      AlertEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Each Alert Event"
    INDEX { alertDefinitionId }
    ::= { apacheAmbariAlertTable 1 }

alertDefinitionId      OBJECT-TYPE
    SYNTAX      Integer32 (-2147483648..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "ID of the Alert"
    ::= { apacheAmbariAlertEntry 1 }

alertDefinitionName    OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  accessible-for-notify
    STATUS      current
    DESCRIPTION
        "Alert Definition Name"
    ::= { apacheAmbariAlertEntry 2 }

```

You can find the MIB file for your cluster on the Ambari Server host, at:

```
/var/lib/ambari-server/resources/APACHE-AMBARI-MIB.txt
```

- For Custom SNMP notification, provide the version, community, host, and port to which the SNMP trap should be sent.

Also, the OID parameter must be configured properly for SNMP trap context. If no custom, enterprise-specific OID is used, you should use the following:

Version	SNMPv1 or SNMPv2c, depending on the network environment
OID	1.3.6.1.4.1.18060.16.1.1
Hosts	A comma-separated list of one or more host FQDNs to which to send the trap
Port	The port on which a process is listening for SNMP traps

5. Click **Save**.

More Information

[Managing Notifications \[116\]](#)

[Javaimail SMTP options](#)

8.8. Creating or Editing Alert Groups

To create or edit alert groups:

Steps

1. In Ambari Web, click **Alerts**.
2. On the **Alerts** page, click the **Actions** menu, then click **Manage Alert Groups**.
3. In **Manage Alert Groups**, click + to create a new alert notification.
4. In, **Create Alert Group**, enter a group name and click **Save**.
5. By clicking on the custom group in the list, you can add or delete alert definitions from this group, and change the notification targets for the group.
6. When you finish your assignments, click **Save**.

8.9. Dispatching Notifications

When an alert is enabled and the alert status changes (for example, from OK to CRITICAL or CRITICAL to OK), Ambari sends either an email or SNMP notification, depending on how notifications are configured.

For email notifications, Ambari sends an email digest that includes all alert status changes. For example, if two alerts become critical, Ambari sends one email message that `Alert A is CRITICAL` and `Ambari B alert is CRITICAL`. Ambari does not send another email notification until status changes again.

For SNMP notifications, Ambari sends one SNMP trap per alert status change. For example, if two alerts become critical, Ambari sends two SNMP traps, one for each alert, and then sends two more when the two alerts change.

8.10. Viewing the Alert Status Log

Whether or not Ambari is configured to send alert notifications, it writes alert status changes to a log on the Ambari Server host. To view this log:

Steps

1. On the Ambari server host, browse to the log directory:

```
cd /var/log/ambari-server/
```

2. View the `ambari-alerts.log` file.
3. Log entries include the time of the status change, the alert status, the alert definition name, and the response text:

```
2015-08-10 22:47:37,120 [OK] [HARD] [STORM] (Storm Server Process) TCP OK -
0.000s response on port 8744
2015-08-11 11:06:18,479 [CRITICAL] [HARD] [AMBARI]
[ambari_server_agent_heartbeat] (Ambari Agent Heartbeat) c6401.ambari.
apache.org is not sending heartbeats
2015-08-11 11:08:18,481 [OK] [HARD] [AMBARI] [ambari_server_agent_heartbeat]
(Ambari Agent Heartbeat) c6401.ambari.apache.org is healthy
```

8.10.1. Customizing Notification Templates

The notification template content produced by Ambari is tightly coupled to a notification type. Email and SNMP notifications both have customizable templates that you can use to generate content. This section describes the steps necessary to change the template used by Ambari when creating alert notifications.

Alert Templates XML Location

By default, an `alert-templates.xml` ships with Ambari. This file contains all of the templates for every known type of notification (for example, EMAIL and SNMP). This file is bundled in the Ambari server `.jar` file so that the template is not exposed on the disk; however, that file is used in the following text, as a reference example.

When you customize the alert template, you are effectively overriding the default alert template's XML, as follows:

1. On the Ambari server host, browse to `/etc/ambari-server/conf` directory.
2. Edit the `ambari.properties` file.
3. Add an entry for the location of your new template:

```
alerts.template.file=/foo/var/alert-templates-custom.xml
```

4. Save the file and restart Ambari Server.

After you restart Ambari, any notification types defined in the new template override those bundled with Ambari. If you choose to provide your own template file, you only need to define notification templates for the types that you wish to override. If a notification template type is not found in the customized template, Ambari will default to the templates that ship with the JAR.

Alert Templates XML Structure

The structure of the template file is defined as follows. Each `<alert-template>` element declares what type of alert notification it should be used for.

```
<alert-templates>
  <alert-template type="EMAIL">
    <subject>
      Subject Content
    </subject>
    <body>
      Body Content
    </body>
  </alert-template>
  <alert-template type="SNMP">
    <subject>
      Subject Content
    </subject>
    <body>
      Body Content
    </body>
  </alert-template>
</alert-templates>
```

Template Variables

The template uses Apache Velocity to render all tokenized content. The following variables are available for use in your template:

<code>\$alert.getAlertDefinition()</code>	The definition of which the alert is an instance.
<code>\$alert.getAlertText()</code>	The specific alert text.
<code>\$alert.getAlertName()</code>	The name of the alert.
<code>\$alert.getAlertState()</code>	The alert state (OK, WARNING, CRITICAL, or UNKNOWN)
<code>\$alert.getServiceName()</code>	The name of the service that the alert is defined for.
<code>\$alert.hasComponentName()</code>	True if the alert is for a specific service component.
<code>\$alert.getComponentName()</code>	The component, if any, that the alert is defined for.
<code>\$alert.hasHostName()</code>	True if the alert was triggered for a specific host.
<code>\$alert.getHostName()</code>	The hostname, if any, that the alert was triggered for.
<code>\$ambari.getServerUrl()</code>	The Ambari Server URL.
<code>\$ambari.getServerVersion()</code>	The Ambari Server version.
<code>\$ambari.getServerHostName()</code>	The Ambari Server hostname.
<code>\$dispatch.getTargetName()</code>	The notification target name.
<code>\$dispatch.getTargetDescription()</code>	The notification target description.
<code>\$summary.getAlerts(service,alertState)</code>	A list of all alerts for a given service or alert state (OK WARNING CRITICAL UNKNOWN)
<code>\$summary.getServicesByAlertState(alertState)</code>	A list of all services for a given alert state (OK WARNING CRITICAL UNKNOWN)
<code>\$summary.getServices()</code>	A list of all services that are reporting an alert in the notification.
<code>\$summary.getCriticalCount()</code>	The CRITICAL alert count.
<code>\$summary.getOkCount()</code>	The OK alert count.
<code>\$summary.getTotalCount()</code>	The total alert count.
<code>\$summary.getUnknownCount()</code>	The UNKNOWN alert count.
<code>\$summary.getWarningCount()</code>	The WARNING alert count.
<code>\$summary.getAlerts()</code>	A list of all of the alerts in the notification.

Example: Modify Alert EMAIL Subject

The following example illustrates how to change the subject line of all outbound email notifications to include a hard-coded identifier:

1. Download the `alert-templates.xml` code as your starting point.
2. On the Ambari Server, save the template to a location such as `/var/lib/ambari-server/resources/alert-templates-custom.xml`.
3. Edit the `alert-templates-custom.xml` file and modify the subject link for the `<alert-template type="EMAIL">` template:

```
<subject>  
<![CDATA[Petstore Ambari has $summary.getTotalCount() alerts!]]>  
</subject>
```

4. Save the file.
5. Browse to `/etc/ambari-server/conf` directory.
6. Edit the `ambari.properties` file.
7. Add an entry for the location of your new template file.

```
alerts.template.file=/var/lib/ambari-server/resources/alert-  
templates-custom.xml
```

8. Save the file and restart Ambari Server.

9. Using Ambari Core Services

The Ambari core services enable you to monitor, analyze, and search the operating status of hosts in your cluster. This chapter describes how to use and configure the following Ambari Core Services:

- [Understanding Ambari Metrics \[123\]](#)
- [Ambari Log Search \(Technical Preview\) \[179\]](#)
- [Ambari Infra \[183\]](#)

9.1. Understanding Ambari Metrics

Ambari Metrics System (AMS) collects, aggregates, and serves Hadoop and system metrics in Ambari-managed clusters.

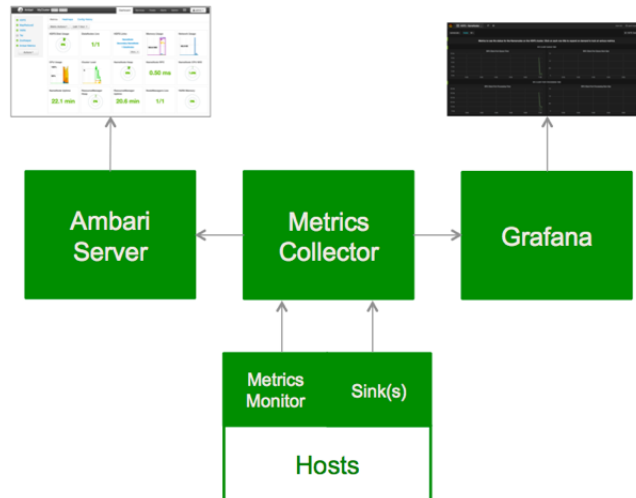
- [AMS Architecture \[123\]](#)
- [Using Grafana \[124\]](#)
- [Grafana Dashboards Reference \[129\]](#)
- [AMS Performance Tuning \[167\]](#)
- [AMS High Availability \[172\]](#)

9.1.1. AMS Architecture

AMS has four components: **Metrics Monitors**, **Hadoop Sinks**, **Metrics Collector**, and **Grafana**.

- **Metrics Monitors** on each host in the cluster collect system-level metrics and publish to the **Metrics Collector**.
- **Hadoop Sinks** plug in to Hadoop components to publish Hadoop metrics to the **Metrics Collector**.
- The **Metrics Collector** is a daemon that runs on a specific host in the cluster and receives data from the registered publishers, the **Monitors**, and the **Sinks**.
- **Grafana** is a daemon that runs on a specific host in the cluster and serves pre-built dashboards for visualizing metrics collected in the **Metrics Collector**.

The following high-level illustration shows how the components of AMS work together to collect metrics and make those metrics available to Ambari.



9.1.2. Using Grafana

Ambari Metrics System includes Grafana with pre-built dashboards for advanced visualization of cluster metrics.

- [Accessing Grafana \[124\]](#)
- [Viewing Grafana Dashboards \[125\]](#)
- [Viewing Selected Metrics on Grafana Dashboards \[127\]](#)
- [Viewing Metrics for Selected Hosts \[128\]](#)

More Information

<http://grafana.org/>

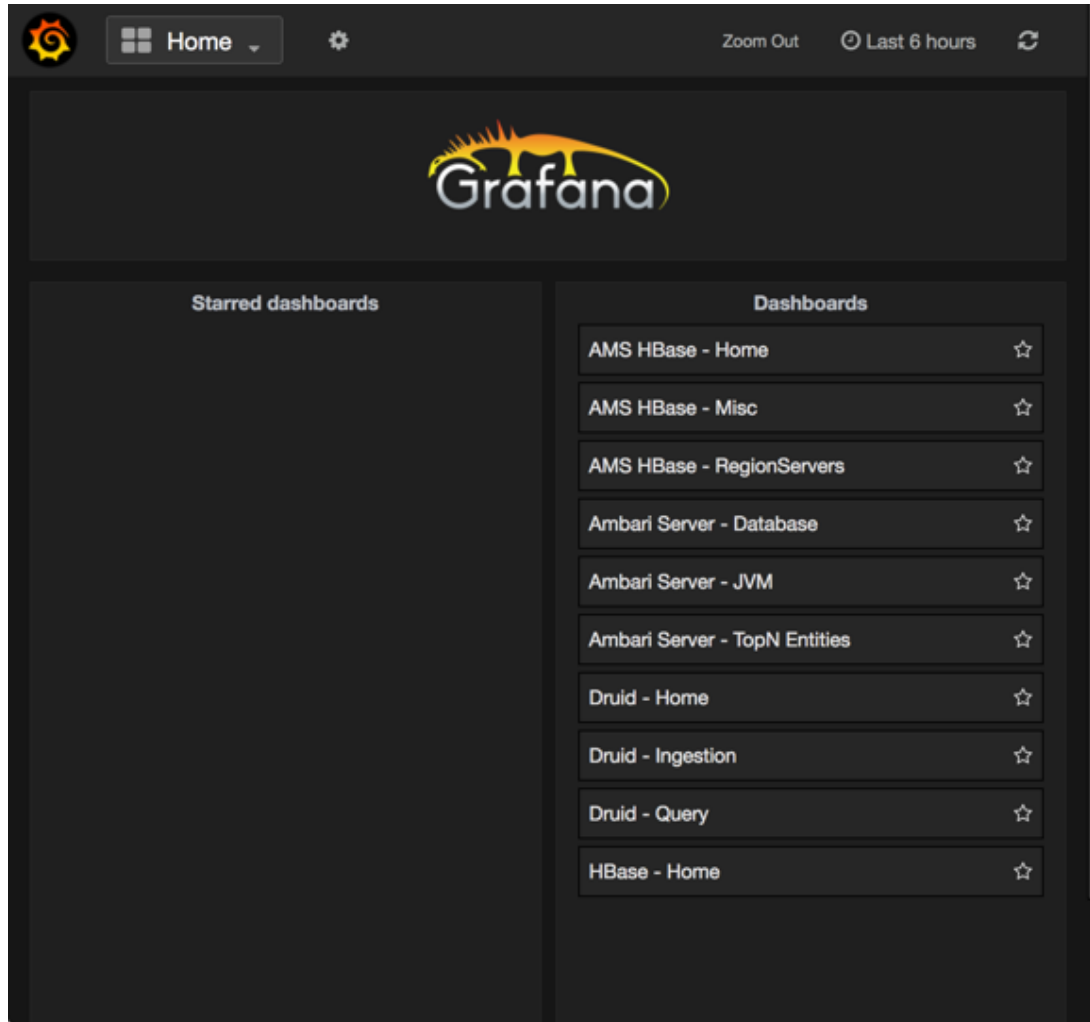
9.1.2.1. Accessing Grafana

To access the Grafana UI:

Steps

1. In Ambari Web, browse to **Services > Ambari Metrics > Summary**.
2. Select **Quick Links** and then choose **Grafana**.

A read-only version of the Grafana interface opens in a new tab in your browser:



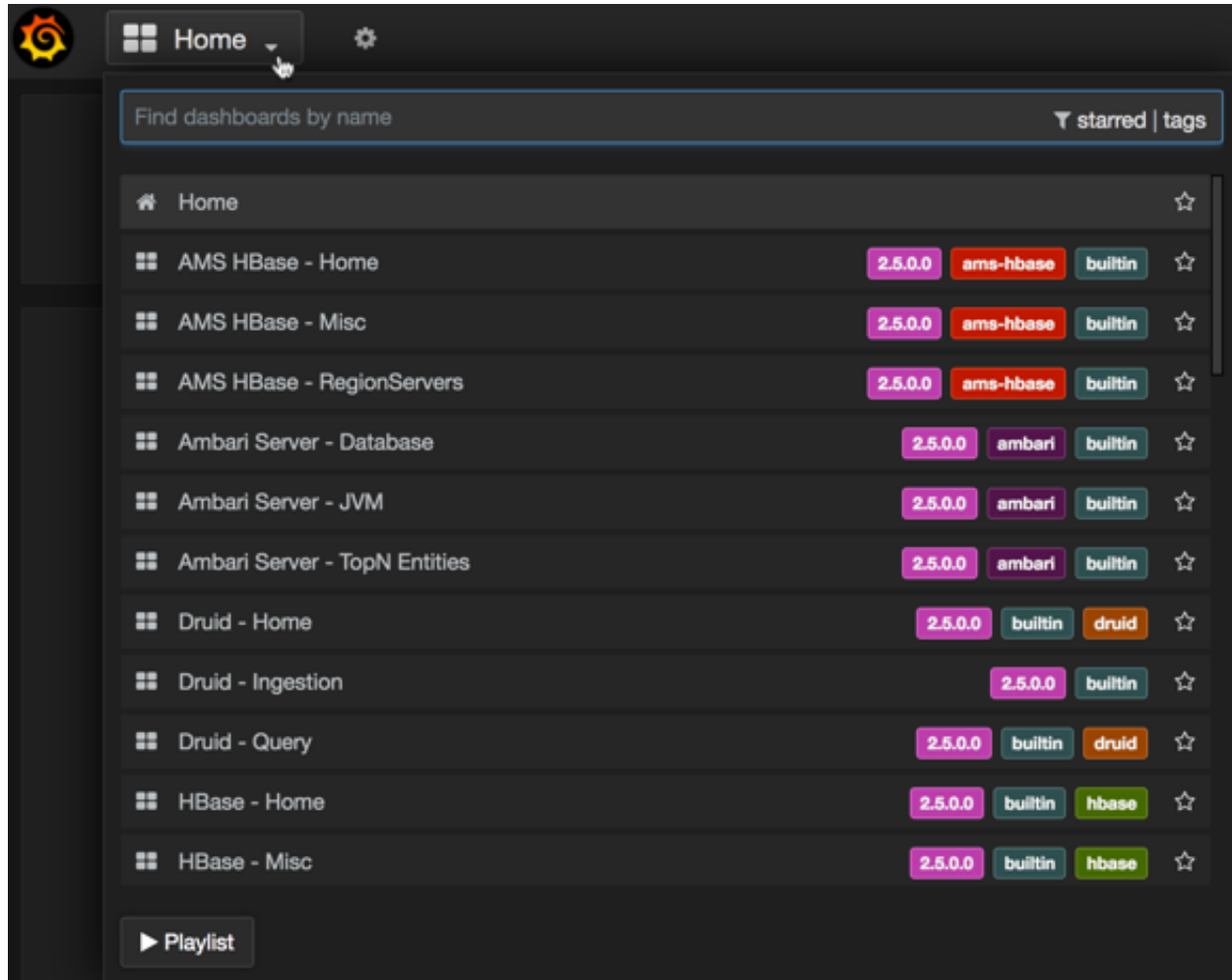
9.1.2.2. Viewing Grafana Dashboards

On the Grafana home page, **Dashboards** provides a short list of links to AMS, Ambari server, Druid and HBase metrics.

To view specific metrics included in the list:

Steps

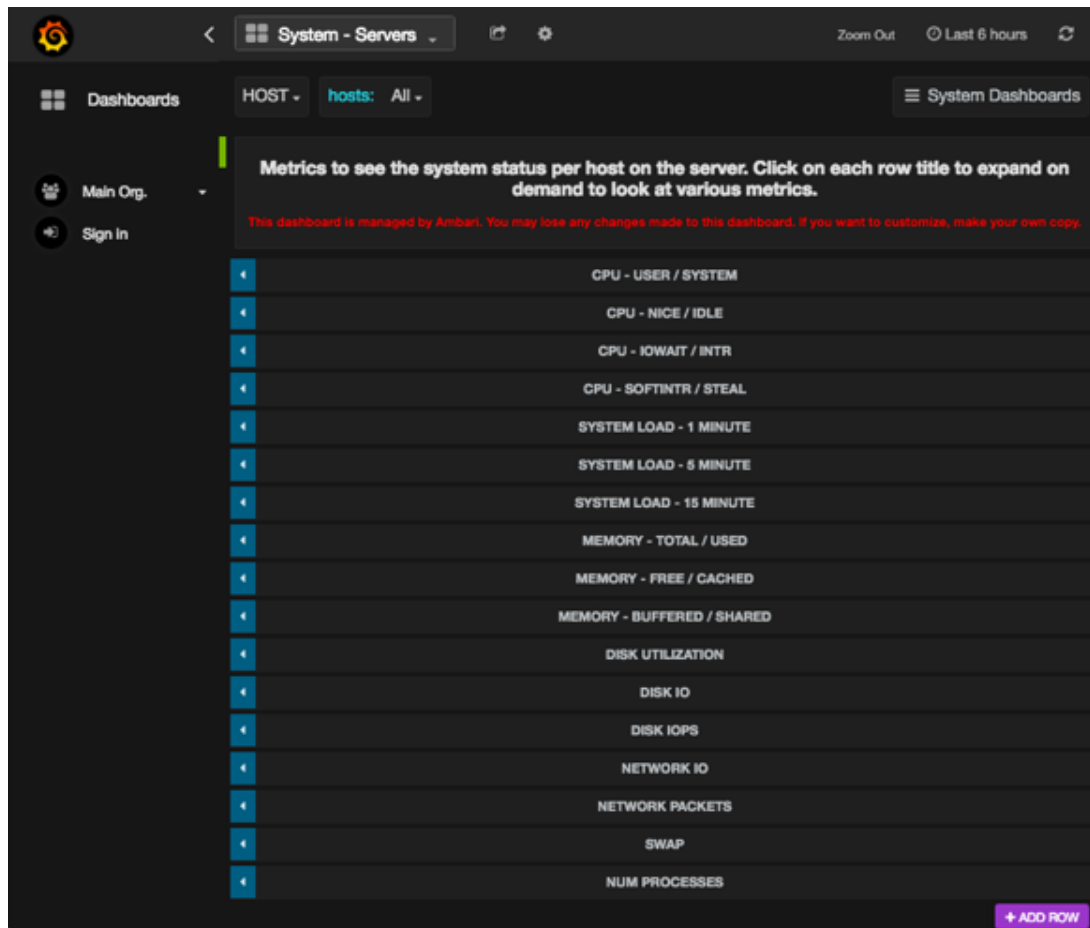
1. In Grafana, browse to **Dashboards**.
2. Click a dashboard name.
3. To see more available dashboards, click the **Home** list.



4. Scroll down to view the whole list.

1. Click a dashboard name, for example **System - Servers**.

The **System - Servers** dashboard opens:



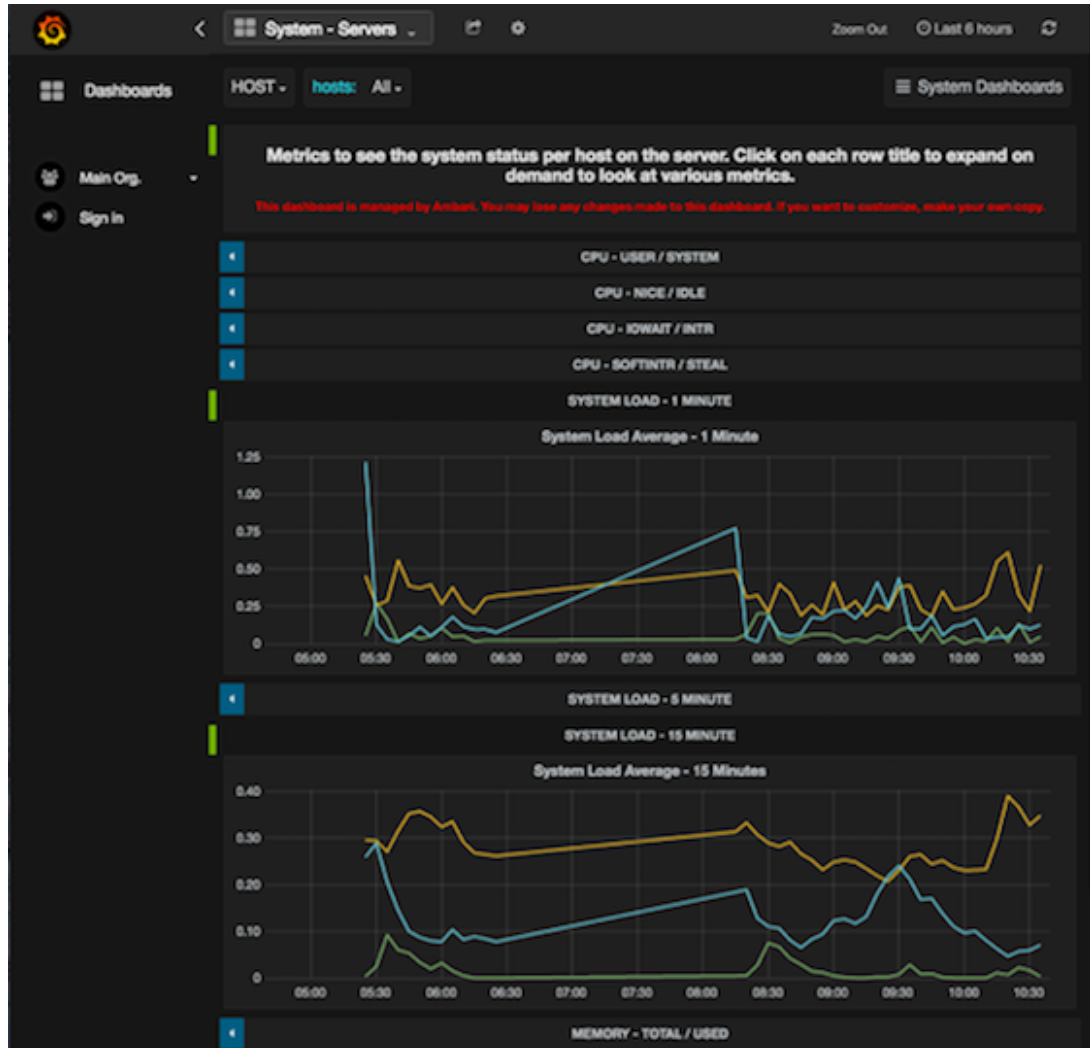
9.1.2.3. Viewing Selected Metrics on Grafana Dashboards

On a dashboard, expand one or more rows to view detailed metrics, continuing the previous example using the **System - Servers** dashboard:

1. In the **System - Servers** dashboard, click a row name. For example, click **System Load Average - 1 Minute**.

The row expands to display a chart that shows metrics information: in this example, the System Load Average - 1 Minute and the System Load Average - 15 Minute rows:

2.

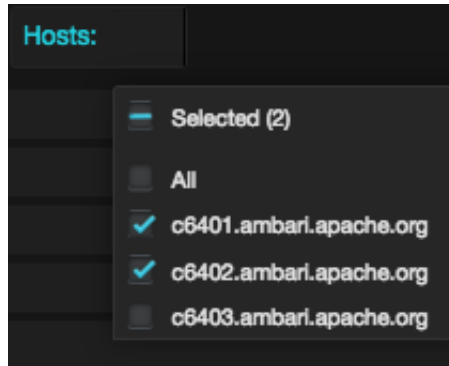


9.1.2.4. Viewing Metrics for Selected Hosts

By default, Grafana shows metrics for all hosts in your cluster. You can limit the displayed metrics to one or more hosts by selecting them from the **Hosts** menu.:

1. Expand **Hosts**.
2. Select one or more host names.

A check mark appears next to selected host names:



Note

Selections in the **Hosts** menu apply to all metrics in the current dashboard. Grafana refreshes the current dashboards when you select a new set of hosts.

9.1.3. Grafana Dashboards Reference

Ambari Metrics System includes Grafana with pre-built dashboards for advanced visualization of cluster metrics.

- [AMS HBase Dashboards \[129\]](#)
- [Ambari Dashboards \[137\]](#)
- [HDFS Dashboards \[139\]](#)
- [YARN Dashboards \[143\]](#)
- [Hive Dashboards \[146\]](#)
- [Hive LLAP Dashboards \[148\]](#)
- [HBase Dashboards \[152\]](#)
- [Kafka Dashboards \[161\]](#)
- [Storm Dashboards \[163\]](#)
- [System Dashboards \[164\]](#)
- [NiFi Dashboard \[166\]](#)

9.1.3.1. AMS HBase Dashboards

AMS HBase refers to the HBase instance managed by Ambari Metrics Service independently. It does not have any connection with the cluster HBase service. AMS HBase Grafana dashboards track the same metrics as the regular HBase dashboard, but for the AMS-owned instance.

The following Grafana dashboards are available for AMS HBase:

- [AMS HBase - Home \[130\]](#)
- [AMS HBase - RegionServers \[131\]](#)

- [AMS HBase - Misc \[136\]](#)

9.1.3.1.1. AMS HBase - Home

The AMS HBase - Home dashboards display basic statistics about an HBase cluster. These dashboards provide insight to the overall status for the HBase cluster.

Row	Metrics	Description
REGIONSERVERS / REGIONS	Num RegionServers	Total number of RegionServers in the cluster.
	Num Dead RegionServers	Total number of RegionServers that are dead in the cluster.
	Num Regions	Total number of regions in the cluster.
	Avg Num Regions per RegionServer	Average number of regions per RegionServer.
NUM REGIONS/STORES	Num Regions / Stores - Total	Total number of regions and stores (column families) in the cluster.
	Store File Size / Count - Total	Total data file size and number of store files.
NUM REQUESTS	Num Requests - Total	Total number of requests (read, write and RPCs) in the cluster.
	Num Request - Breakdown - Total	Total number of get,put,mutate,etc requests in the cluster.
REGIONSERVER MEMORY	RegionServer Memory - Average	Average used, max or committed on-heap and offheap memory for RegionServers.
	RegionServer Offheap Memory - Average	Average used, free or committed on-heap and offheap memory for RegionServers.
MEMORY - MEMSTORE BLOCKCACHE	Memstore - BlockCache - Average	Average blockcache and memstore sizes for RegionServers.
	Num Blocks in BlockCache - Total	Total number of (hfile) blocks in the blockcaches across all RegionServers.
BLOCKCACHE	BlockCache Hit/Miss/s Total	Total number of blockcache hits misses and evictions across all RegionServers.
	BlockCache Hit Percent - Average	Average blockcache hit percentage across all RegionServers.
OPERATION LATENCIES - GET/MUTATE	Get Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Get operation across all RegionServers.
	Mutate Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Mutate operation across all RegionServers.
OPERATION LATENCIES - DELETE/INCREMENT	Delete Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Delete operation across all RegionServers.
	Increment Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Increment operation across all RegionServers.
OPERATION LATENCIES - APPEND/REPLAY	Append Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Append operation across all RegionServers.
	Replay Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Replay operation across all RegionServers.
REGIONSERVER RPC	RegionServer RPC - Average	Average number of RPCs, active handler threads and open connections across all RegionServers.
	RegionServer RPC Queues - Average	Average number of calls in different RPC scheduling queues and the size of all requests in the RPC queue across all RegionServers.

Row	Metrics	Description
REGIONSERVER RPC	RegionServer RPC Throughput - Average	Average sent and received bytes from the RPC across all RegionServers.

9.1.3.1.2. AMS HBase - RegionServers

The AMS HBase - RegionServers dashboards display metrics for RegionServers in the monitored HBase cluster, including some performance-related data. These dashboards help you view basic I/O data and compare load among RegionServers.

Row	Metrics	Description
NUM REGIONS	Num Regions	Number of regions in the RegionServer.
STORE FILES	Store File Size	Total size of the store files (data files) in the RegionServer.
	Store File Count	Total number of store files in the RegionServer.
NUM REQUESTS	Num Total Requests /s	Total number of requests (both read and write) per second in the RegionServer.
	Num Write Requests /s	Total number of write requests per second in the RegionServer.
	Num Read Requests /s	Total number of read requests per second in the RegionServer.
NUM REQUESTS - GET / SCAN	Num Get Requests /s	Total number of Get requests per second in the RegionServer.
	Num Scan Next Requests /s	Total number of Scan requests per second in the RegionServer.
NUM REQUESTS - MUTATE / DELETE	Num Mutate Requests - /s	Total number of Mutate requests per second in the RegionServer.
	Num Delete Requests /s	Total number of Delete requests per second in the RegionServer.
NUM REQUESTS - APPEND / INCREMENT	Num Append Requests /s	Total number of Append requests per second in the RegionServer.
	Num Increment Requests /s	Total number of Increment requests per second in the RegionServer.
	Num Replay Requests /s	Total number of Replay requests per second in the RegionServer.
MEMORY	RegionServer Memory Used	Heap Memory used by the RegionServer.
	RegionServer Offheap Memory Used	Offheap Memory used by the RegionServer.
MEMSTORE	Memstore Size	Total Memstore memory size of the RegionServer.
BLOCKCACHE - OVERVIEW	BlockCache - Size	Total BlockCache size of the RegionServer.
	BlockCache - Free Size	Total free space in the BlockCache of the RegionServer.
	Num Blocks in Cache	Total number of hfile blocks in the BlockCache of the RegionServer.
BLOCKCACHE - HITS/MISSES	Num BlockCache Hits /s	Number of BlockCache hits per second in the RegionServer.
	Num BlockCache Misses /s	Number of BlockCache misses per second in the RegionServer.
	Num BlockCache Evictions /s	Number of BlockCache evictions per second in the RegionServer.
	BlockCache Caching Hit Percent	Percentage of BlockCache hits per second for requests that requested cache blocks in the RegionServer.

Row	Metrics	Description
	BlockCache Hit Percent	Percentage of BlockCache hits per second in the RegionServer.
OPERATION LATENCIES - GET	Get Latencies - Mean	Mean latency for Get operation in the RegionServer.
	Get Latencies - Median	Median latency for Get operation in the RegionServer.
	Get Latencies - 75th Percentile	75th percentile latency for Get operation in the RegionServer.
	Get Latencies - 95th Percentile	95th percentile latency for Get operation in the RegionServer.
	Get Latencies - 99th Percentile	99th percentile latency for Get operation in the RegionServer.
	Get Latencies - Max	Max latency for Get operation in the RegionServer.
OPERATION LATENCIES - SCAN NEXT	Scan Next Latencies - Mean	Mean latency for Scan operation in the RegionServer.
	Scan Next Latencies - Median	Median latency for Scan operation in the RegionServer.
	Scan Next Latencies - 75th Percentile	75th percentile latency for Scan operation in the RegionServer.
	Scan Next Latencies - 95th Percentile	95th percentile latency for Scan operation in the RegionServer.
	Scan Next Latencies - 99th Percentile	99th percentile latency for Scan operation in the RegionServer.
	Scan Next Latencies - Max	Max latency for Scan operation in the RegionServer.
OPERATION LATENCIES - MUTATE	Mutate Latencies - Mean	Mean latency for Mutate operation in the RegionServer.
	Mutate Latencies - Median	Median latency for Mutate operation in the RegionServer.
	Mutate Latencies - 75th Percentile	75th percentile latency for Mutate operation in the RegionServer.
	Mutate Latencies - 95th Percentile	95th percentile latency for Mutate operation in the RegionServer.
	Mutate Latencies - 99th Percentile	99th percentile latency for Mutate operation in the RegionServer.
	Mutate Latencies - Max	Max latency for Mutate operation in the RegionServer.
OPERATION LATENCIES - DELETE	Delete Latencies - Mean	Mean latency for Delete operation in the RegionServer.
	Delete Latencies - Median	Median latency for Delete operation in the RegionServer.
	Delete Latencies - 75th Percentile	75th percentile latency for Delete operation in the RegionServer.
	Delete Latencies - 95th Percentile	95th percentile latency for Delete operation in the RegionServer.
	Delete Latencies - 99th Percentile	99th percentile latency for Delete operation in the RegionServer.
	Delete Latencies - Max	Max latency for Delete operation in the RegionServer.

Row	Metrics	Description
OPERATION LATENCIES - INCREMENT	Increment Latencies - Mean	Mean latency for Increment operation in the RegionServer.
	Increment Latencies - Median	Median latency for Increment operation in the RegionServer.
	Increment Latencies - 75th Percentile	75th percentile latency for Increment operation in the RegionServer.
	Increment Latencies - 95th Percentile	95th percentile latency for Increment operation in the RegionServer.
	Increment Latencies - 99th Percentile	99th percentile latency for Increment operation in the RegionServer.
	Increment Latencies - Max	Max latency for Increment operation in the RegionServer.
OPERATION LATENCIES - APPEND	Append Latencies - Mean	Mean latency for Append operation in the RegionServer.
	Append Latencies - Median	Median latency for Append operation in the RegionServer.
	Append Latencies - 75th Percentile	75th percentile latency for Append operation in the RegionServer.
	Append Latencies - 95th Percentile	95th percentile latency for Append operation in the RegionServer.
	Append Latencies - 99th Percentile	99th percentile latency for Append operation in the RegionServer.
	Append Latencies - Max	Max latency for Append operation in the RegionServer.
OPERATION LATENCIES - REPLAY	Replay Latencies - Mean	Mean latency for Replay operation in the RegionServer.
	Replay Latencies - Median	Median latency for Replay operation in the RegionServer.
	Replay Latencies - 75th Percentile	75th percentile latency for Replay operation in the RegionServer.
	Replay Latencies - 95th Percentile	95th percentile latency for Replay operation in the RegionServer.
	Replay Latencies - 99th Percentile	99th percentile latency for Replay operation in the RegionServer.
	Replay Latencies - Max	Max latency for Replay operation in the RegionServer.
RPC - OVERVIEW	Num RPC /s	Number of RPCs per second in the RegionServer.
	Num Active Handler Threads	Number of active RPC handler threads (to process requests) in the RegionServer.
	Num Connections	Number of connections to the RegionServer.
RPC - QUEUES	Num RPC Calls in General Queue	Number of RPC calls in the general processing queue in the RegionServer.
	Num RPC Calls in Priority Queue	Number of RPC calls in the high priority (for system tables) processing queue in the RegionServer.
	Num RPC Calls in Replication Queue	Number of RPC calls in the replication processing queue in the RegionServer.
	RPC - Total Call Queue Size	Total data size of all RPC calls in the RPC queues in the RegionServer.
RPC - CALL QUEUED TIMES	RPC - Call Queued Time - Mean	Mean latency for RPC calls to stay in the RPC queue in the RegionServer.

Row	Metrics	Description
	RPC - Call Queued Time - Median	Median latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - 75th Percentile	75th percentile latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - 95th Percentile	95th percentile latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - 99th Percentile	99th percentile latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - Max	Max latency for RPC calls to stay in the RPC queue in the RegionServer.
RPC - CALL PROCESS TIMES	RPC - Call Process Time - Mean	Mean latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - Median	Median latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - 75th Percentile	75th percentile latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - 95th Percentile	95th percentile latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - 99th Percentile	99th percentile latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - Max	Max latency for RPC calls to be processed in the RegionServer.
RPC - THROUGHPUT	RPC - Received bytes /s	Received bytes from the RPC in the RegionServer.
	RPC - Sent bytes /s	Sent bytes from the RPC in the RegionServer.
WAL - FILES	Num WAL - Files	Number of Write-Ahead-Log files in the RegionServer.
	Total WAL File Size	Total files sized of Write-Ahead-Logs in the RegionServer.
WAL - THROUGHPUT	WAL - Num Appends /s	Number of append operations per second to the filesystem in the RegionServer.
	WAL - Num Sync /s	Number of sync operations per second to the filesystem in the RegionServer.
WAL - SYNC LATENCIES	WAL - Sync Latencies - Mean	Mean latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - Median	Median latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - 75th Percentile	75th percentile latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - 95th Percentile	95th percentile latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - 99th Percentile	99th percentile latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - Max	Max latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
WAL - APPEND LATENCIES	WAL - Append Latencies - Mean	Mean latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.

Row	Metrics	Description
	WAL - Append Latencies - Median	Median latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - 75th Percentile	95th percentile latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - 95th Percentile	95th percentile latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - 99th Percentile	99th percentile latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - Max	Max latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
WAL - APPEND SIZES	WAL - Append Sizes - Mean	Mean data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - Median	Median data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - 75th Percentile	75th percentile data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - 95th Percentile	95th percentile data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - 99th Percentile	99th percentile data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - Max	Max data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
SLOW OPERATIONS	WAL Num Slow Append /s	Number of append operations per second to the filesystem that took more than 1 second in the RegionServer.
	Num Slow Gets /s	Number of Get requests per second that took more than 1 second in the RegionServer.
	Num Slow Puts /s	Number of Put requests per second that took more than 1 second in the RegionServer.
	Num Slow Deletes /s	Number of Delete requests per second that took more than 1 second in the RegionServer.
FLUSH/COMPACTION QUEUES	Flush Queue Length	Number of Flush operations waiting to be processed in the RegionServer. A higher number indicates flush operations being slow.
	Compaction Queue Length	Number of Compaction operations waiting to be processed in the RegionServer. A higher number indicates compaction operations being slow.
	Split Queue Length	Number of Region Split operations waiting to be processed in the RegionServer. A higher number indicates split operations being slow.
JVM - GC COUNTS	GC Count /s	Number of Java Garbage Collections per second.
	GC Count ParNew /s	Number of Java ParNew (YoungGen) Garbage Collections per second.
	GC Count CMS /s	Number of Java CMS Garbage Collections per second.
JVM - GC TIMES	GC Times /s	Total time spend in Java Garbage Collections per second.
	GC Times ParNew /s	Total time spend in Java ParNew(YoungGen) Garbage Collections per second.
	GC Times CMS /s	Total time spend in Java CMS Garbage Collections per second.

Row	Metrics	Description
LOCALITY	Percent Files Local	Percentage of files served from the local DataNode for the RegionServer.

9.1.3.1.3. AMS HBase - Misc

The AMS HBase - Misc dashboards display miscellaneous metrics related to the HBase cluster. You can use these metrics for tasks like debugging authentication and authorization issues and exceptions raised by RegionServers.

Row	Metrics	Description
REGIONS IN TRANSITION	Master - Regions in Transition	Number of regions in transition in the cluster.
	Master - Regions in Transition Longer Than Threshold Time	Number of regions in transition that are in transition state for longer than 1 minute in the cluster.
	Regions in Transition Oldest Age	Maximum time that a region stayed in transition state.
NUM THREADS - RUNNABLE	Master Num Threads - Runnable	Number of threads in the Master.
	RegionServer Num Threads - Runnable	Number of threads in the RegionServer.
NUM THREADS - BLOCKED	Master Num Threads - Blocked	Number of threads in the Blocked State in the Master.
	RegionServer Num Threads - Blocked	Number of threads in the Blocked State in the RegionServer.
NUM THREADS - WAITING	Master Num Threads - Waiting	Number of threads in the Waiting State in the Master.
	RegionServer Num Threads - Waiting	Number of threads in the Waiting State in the RegionServer.
NUM THREADS - TIMED WAITING	Master Num Threads - Timed Waiting	Number of threads in the Timed-Waiting State in the Master.
	RegionServer Num Threads - Timed Waiting	Number of threads in the Timed-Waiting State in the RegionServer.
NUM THREADS - NEW	Master Num Threads - New	Number of threads in the New State in the Master.
	RegionServer Num Threads - New	Number of threads in the New State in the RegionServer.
NUM THREADS - TERMINATED	Master Num Threads - Terminated	Number of threads in the Terminated State in the Master.
	RegionServer Num Threads - Terminated	Number of threads in the Terminated State in the RegionServer.
RPC AUTHENTICATION	RegionServer RPC Authentication Successes /s	Number of RPC successful authentications per second in the RegionServer.
	RegionServer RPC Authentication Failures /s	Number of RPC failed authentications per second in the RegionServer.

Row	Metrics	Description
RPC Authorization	RegionServer RPC Authorization Successes /s	Number of RPC successful autorizations per second in the RegionServer.
	RegionServer RPC Authorization Failures /s	Number of RPC failed autorizations per second in the RegionServer.
EXCEPTIONS	Master Exceptions /s	Number of exceptions in the Master.
	RegionServer Exceptions /s	Number of exceptions in the RegionServer.

9.1.3.2. Ambari Dashboards

The following Grafana dashboards are available for Ambari:

- [Ambari Server Database \[137\]](#)
- [Ambari Server JVM \[137\]](#)
- [Ambari Server Top N \[138\]](#)

9.1.3.2.1. Ambari Server Database

Metrics that show operating status for the Ambari server database.

Row	Metrics	Description
TOTAL READ ALL QUERY	Total Read All Query Counter (Rate)	Total ReadAllQuery operations performed.
	Total Read All Query Timer (Rate)	Total time spent on ReadAllQuery.
TOTAL CACHE HITS & MISSES	Total Cache Hits (Rate)	Total cache hits on Ambari Server with respect to EclipseLink cache.
	Total Cache Misses (Rate)	Total cache misses on Ambari Server with respect to EclipseLink cache.
QUERY	Query Stages Timings	Average time spent on every query sub stage by Ambari Server
	Query Types Avg. Timings	Average time spent on every query type by Ambari Server.
HOST ROLE COMMAND ENTITY	Counter.ReadAllQuery.HostRoleCommandEntity (Rate)	ReadAllQuery operations per second in which ReadAllQuery operation on HostRoleCommandEntity is performed.
	Timer.ReadAllQuery.HostRoleCommandEntity (Rate)	ReadAllQuery operation on HostRoleCommandEntity is performed.
	ReadAllQuery.HostRoleCommandEntity (Rate)	ReadAllQuery operation for a ReadAllQuery operation on HostRoleCommandEntity (Timer / Counter).

9.1.3.2.2. Ambari Server JVM

Metrics to see status for the Ambari Server Java virtual machine.

Row	Metrics	Description
JVM - MEMORY PRESSURE	Heap Usage	Used, max or committed on-heap memory for Ambari Server.

Row	Metrics	Description
	Off-Heap Usage	Used, max or committed off-heap memory for Ambari Server.
JVM GC COUNT	GC Count Par new /s	Number of Java ParNew (YoungGen) Garbage Collections per second.
	GC Time Par new /s	Total time spend in Java ParNew(YoungGen) Garbage Collections per second.
	GC Count CMS /s	Number of Java Garbage Collections per second.
	GC Time Par CMS /s	Total time spend in Java CMS Garbage Collections per second.
JVM THREAD COUNT	Thread Count	Number of active, daemon, deadlock, blocked and runnable threads.

9.1.3.2.3. Ambari Server Top N

Metrics to see top performing users and operations for Ambari.

Row	Metrics	Description
READ ALL QUERY	Top ReadAllQuery Counters	Top N Ambari Server entities by number of ReadAllQuery operations performed.
	Top ReadAllQuery Timers	Top N Ambari Server entities by time spent on ReadAllQuery operations.
Cache Misses	Cache Misses	Top N Ambari Server entities by number of Cache Misses.

9.1.3.3. Druid Dashboards

The following Grafana dashboards are available for Druid:

- [Druid - Home \[138\]](#)
- [Druid - Ingestion \[139\]](#)
- [Druid - Query \[139\]](#)

9.1.3.3.1. Druid - Home

Metrics that show operating status for Druid.

Row	Metrics	Description
DRUID BROKER	JVM Heap	JVM Heap used by the Druid Broker Node
	JVM GCM Time	Time spent by the Druid Broker Node in JVM Garbage collection
DRUID HISTORICAL	JVM Heap	JVM Heap used by the Druid Historical Node
	JVM GCM Time	Time spent by the Druid Historical Node in JVM Garbage collection
DRUID COORDINATER	JVM Heap	JVM Heap used by the Druid Coordinator Node
	JVM GCM Time	Time spent by the Druid Coordinator Node in JVM Garbage collection
DRUID OVERLORD	JVM Heap	JVM Heap used by the Druid Overlord Node
	JVM GCM Time	Time spent by the Druid Overlord Node in JVM Garbage collection
DRUID MIDDLEMANAGER	JVM Heap	JVM Heap used by the Druid Middlemanager Node

Row	Metrics	Description
	JVM GCM Time	Time spent by the Druid Middlemanager Node in JVM Garbage collection

9.1.3.3.2. Druid - Ingestion

Metrics to see status for Druid data ingestion rates.

Row	Metrics	Description
INGESTION METRICS	Ingested Events	Number of events ingested on real time nodes
	Events Thrown Away	Number of events rejected because they are outside the windowPeriod.
	Unparseable Events	Number of events rejected because they did not parse
INTERMEDIATE PERSISTS METRICS	Persisted Rows	Number of Druid rows persisted on disk
	Average Persist Time	Average time taken to persist intermediate segments to disk
	Intermediate Persist Count	Number of times that intermediate segments were persisted
SEGMENT SIZE METRICS	Ave Segment Size	Average size of added Druid segments
	Total Segment Size	Total size of added Druid segments

9.1.3.3.3. Druid - Query

Metrics to see status of Druid queries.

Row	Metrics	Description
Query Time Metrics	Broker Query Time	Average Time taken by Druid Broker node to process queries
	Historical Query Time	Average time taken by Druid historical nodes to process queries
	Realtime Query Time	Average time taken by Druid real time nodes to process queries
SEGMENT SCAN METRICS	Historical Segment Scan Time	Average time taken by Druid historical nodes to scan individual segments
	Realtime Segment Scan Time	Average time taken by Druid real time nodes to scan individual segments
	Historical Query Wait Time	Average time spent waiting for a segment to be scanned on historical node
	Realtime Query Wait Time	Average time spent waiting for a segment to be scanned on real time node
	Pending Historical Segment Scans	Average Number of pending segment scans on historical nodes
	Pending Realtime Segment Scans	Average Number of pending segment scans on real time nodes

9.1.3.4. HDFS Dashboards

The following Grafana dashboards are available for Hadoop Distributed File System (HDFS) components:

- [HDFS - Home \[140\]](#)
- [HDFS - NameNodes \[140\]](#)

- [HDFS - DataNodes \[141\]](#)
- [HDFS - Top-N \[142\]](#)
- [HDFS - Users \[143\]](#)

9.1.3.4.1. HDFS - Home

The HDFS - Home dashboard displays metrics that show operating status for HDFS.



Note

In a NameNode HA setup, metrics are collected from and displayed for both the active and the standby NameNode.

Row	Metrics	Description
NUMBER OF FILES UNDER CONSTRUCTION & RPC CLIENT CONNECTIONS	Number of Files Under Construction	Number of HDFS files that are still being written.
	RPC Client Connections	Number of open RPC connections from clients on NameNode(s).
TOTAL FILE OPERATIONS & CAPACITY USED	Total File Operations	Total number of operations on HDFS files, including file creation/deletion/rename/truncation, directory/file/block information retrieval, and snapshot related operations.
	Capacity Used	"CapacityTotalGB" shows total HDFS storage capacity, in GB. "CapacityUsedGB" indicates total used HDFS storage capacity, in GB.
RPC CLIENT PORT SLOW CALLS & HDFS TOTAL LOAD	RPC Client Port Slow Calls	Number of slow RPC requests on NameNode. A "slow" RPC request is one that takes more time to complete than 99.7% of other requests.
	HDFS Total Load	Total number of connections on all the DataNodes sending/receiving data.
ADD BLOCK STATUS	Add Block Time	The average time (in ms) serving addBlock RPC request on NameNode(s).
	Add Block Num Ops	The rate of addBlock RPC requests on NameNode(s).

9.1.3.4.2. HDFS - NameNodes

Metrics to see status for the NameNodes.

Row	Metrics	Description
RPC CLIENT QUEUE TIME	RPC Client Port Queue Time	Average time that a RPC request (on the RPC port facing to the HDFS clients) waits in the queue.
	RPC Client Port Queue Num Ops	Total number of RPC requests in the client port queue.
RPC CLIENT PORT PROCESSING TIME	RPC Client Port Processing Time	Average RPC request processing time in milliseconds, on the client port.
	RPC Client Port Processing Num Ops	Total number of RPC active requests through the client port.
GC COUNT & GC TIME	GC Count	Shows the JVM garbage collection rate on the NameNode.
	GC Time	Shows the garbage collection time in milliseconds.
GC PAR NEW	GC Count Par New	The number of times young generation garbage collection happened.
	GC Time Par New	Indicates the duration of young generation garbage collection.

Row	Metrics	Description
GC EXTRA SLEEP & WARNING THRESHOLD EXCEEDED	GC Extra Sleep Time	Indicates total garbage collection extra sleep time.
	GC Warning Threshold Exceeded Count	Indicates number of times that the garbage collection warning threshold is exceeded
RPC CLIENT PORT QUEUE & BACKOFF	RPC Client Port Queue Length	Indicates the current length of the RPC call queue.
	RPC Client Port Backoff	Indicates number of client backoff requests.
RPC SERVICE PORT QUEUE & NUM OPS	RPC Service Port Queue Time	Average time a RPC request waiting in the queue, in milliseconds. These requests are on the RPC port facing to the HDFS services, including DataNodes and the other NameNode.
	RPC Service Port Queue Num Ops	Total number of RPC requests waiting in the queue. These requests are on the RPC port facing to the HDFS services, including DataNodes and the other NameNode.
RPC SERVICE PORT PROCESSING TIME & NUM OPS	RPC Service Port Processing Time	Average RPC request processing time in milliseconds, for the service port.
	RPC Service Port Processing Num Ops	Number of RPC requests processed for the service port.
RPC SERVICE PORT CALL QUEUE LENGTH & SLOW CALLS	RPC Service Port Call Queue Length	The current length of the RPC call queue.
	RPC Service Port Slow Calls	The number of slow RPC requests, for the service port.
TRANSACTIONS SINCE LAST EDIT & CHECKPOINT	Transactions Since Last Edit Roll	Total number of transactions since the last editlog segment.
	Transactions Since Last Checkpoint	Total number of transactions since the last editlog segment checkpoint.
LOCK QUEUE LENGTH & EXPIRED HEARTBEATS	Lock Queue Length	Shows the length of the wait Queue for the FSNameSystemLock.
	Expired Heartbeats	Indicates the number of times expired heartbeats are detected on NameNode.
THREADS BLOCKED / WAITING	Threads Blocked	Indicates the number of threads in a BLOCKED state, which means they are waiting for a lock.
	Threads Waiting	Indicates the number of threads in a WAITING state, which means they are waiting for another thread to perform an action.

9.1.3.4.3. HDFS - DataNodes

Metrics to see status for the DataNodes.

Row	Metrics	Description
BLOCKS WRITTEN / READ	Blocks Written	The rate or number of blocks written to a DataNode.
	Blocks Read	The rate or number of blocks read from a DataNode.
FSYNCH TIME / NUM OPS	Fsynch Time	Average fsync time.
	Fsynch Num Ops	Total number of fsync operations.
DATA PACKETS BLOCKED / NUM OPS	Data Packet Blocked Time	Indicates the average waiting time of transferring a data packet on a DataNode.
	Data Packet Blocked Num Ops	Indicates the number of data packets transferred on a DataNode.
PACKET TRANSFER BLOCKED / NUM OPS	Packet Transfer Time	Average transfer time of sending data packets on a DataNode.

Row	Metrics	Description
	Packet Transfer Num Ops	Indicates the number of data packets blocked on a DataNode.
NETWORK ERRORS / GC COUNT	Network Errors	Rate of network errors on JVM.
	GC Count	Garbage collection DataNode hits.
GC TIME / GC TIME PARNEW	GC Time	JVM garbage collection time on a DataNode.
	GC Time ParNew	Young generation (ParNew) garbage collection time on a DataNode.

9.1.3.4.4. HDFS - Top-N

Metrics that show

- Which users perform most HDFS operations on the cluster
- Which HDFS operations run most often on the cluster.

Row	Metrics	Description
TOP N - Operations Count	Top N Total Operations Count	Represents the metrics that show the total operation count per operation for all users
	1 min sliding window	1 minute interval
	Top N Total Operations Count	Represents the metrics that show the total operation count per operation for all users
	5 min sliding window	5 minute interval
TOP N - Total Operations Count By User	Top N Total Operations Count by User	Represents the metrics that show the total operation count per user
	1 min sliding window	shown for 1-minute intervals
	Top N Total Operations Count by User	Represents the metrics that show the total operation count per user
	5 min sliding window	shown for 5-minute intervals
TOP N - Operations by User	Top N Total Operations Count by User	Represents the metrics that show the total operation count per user
	25 min sliding window	shown for 25-minute intervals
	TOP N - Operations by User	Represents the drilled down User x Op metrics against the TotalCount
	1 min sliding window	shown for 1-minute intervals
TOP N - Operations by User	TOP N - Operations by User	Represents the drilled down User x Op metrics against the TotalCount
		shown for 5-minute intervals

Row	Metrics	Description
	5 min sliding window	
	TOP N - Operations by User	Represents the drilled down User x Op metrics against the TotalCount
	25 min sliding window	shown for 25-minute intervals

9.1.3.4.5. HDFS - Users

Metrics to see status for Users.

Row	Metrics	Description
Namenode Rpc Caller Volume	Namenode Rpc Caller Volume	Number of RPC calls made by top(10) users.
Namenode Rpc Caller Priority	Namenode Rpc Caller Priority	Priority assignment for incoming calls from top(10) users.

9.1.3.5. YARN Dashboards

The following Grafana dashboards are available for YARN:

- [YARN - Home \[143\]](#)
- [YARN - Applications \[143\]](#)
- [YARN - MR JobHistory Server \[144\]](#)
- [YARN - MR JobHistory Server \[144\]](#)
- [YARN - NodeManagers \[144\]](#)
- [YARN - Queues \[145\]](#)
- [YARN - ResourceManager \[145\]](#)

9.1.3.5.1. YARN - Home

Metrics to see the overall status for the YARN cluster.

Metrics	Description
Nodes	The number of (active, unhealthy, lost) nodes in the cluster.
Apps	The number of (running, pending, completed, failed) apps in the cluster.
Cluster Memory Available	Total available memory in the cluster.

9.1.3.5.2. YARN - Applications

Metrics to see status of Applications on the YARN Cluster.

Metrics	Description
Applications By Running Time	Number of apps by running time in 4 categories by default (< 1 hour, 1 ~ 5 hours, 5 ~ 24 hours, > 24 hours).

Metrics	Description
Apps Running vs Pending	The number of running apps vs the number of pending apps in the cluster.
Apps Submitted vs Completed	The number of submitted apps vs the number of completed apps in the cluster.
Avg AM Launch Delay	The average time taken from allocating an AM container to launching an AM container.
Avg AM Register Delay	The average time taken from RM launches an AM container to AM registers back with RM.

9.1.3.5.3. YARN - MR JobHistory Server

Metrics to see status of the Job History Server.

Row	Metrics	Description
JVM METRICS	GC Count	Accumulated GC count over time.
	GC Time	Accumulated GC time over time.
	Heap Mem Usage	Current heap memory usage.
	NonHeap Mem Usage	Current non-heap memory usage.

9.1.3.5.4. YARN - NodeManagers

Metrics to see status of YARN NodeManagers on the YARN cluster.

Row	Metrics	Description
NUM CONTAINERS	Containers Running	Current number of running containers.
	Containers Failed	Accumulated number of failed containers.
	Containers Killed	Accumulated number of killed containers.
	Containers Completed	Accumulated number of completed containers.
MEMORY UTILIZATION	Memory Available	Available memory for allocating containers on this node.
	Used Memory	Used memory by containers on this node.
DISK UTILIZATION	Disk Utilization for Good Log Dirs	Disk utilization percentage across all good log directories.
	Disk Utilization for Good Local Dirs	Disk utilization percentage across all good local directories.
	Bad Log Dirs	Number of bad log directories.
	Bad Local Dirs	Number of bad local directories.
AVE CONTAINER LAUNCH DELAY	Ave Container Launch Delay	Average time taken for a NM to launch a container.
RPC METRICS	RPC Avg Processing Time	Average time for processing a RPC call.
	RPC Avg Queue Time	Average time for queuing a PRC call.
	RPC Call Queue Length	The length of the RPC call queue.
	RPC Slow Calls	Number of slow RPC calls.
JVM METRICS	Heap Mem Usage	Current heap memory usage.
	NonHeap Mem Usage	Current non-heap memory usage.
	GC Count	Accumulated GC count over time.
	GC Time	Accumulated GC time over time.

Row	Metrics	Description
LOG4J METRICS	LOG ERROR	Number of ERROR logs.
	LOG FATAL	Number of FATAL logs.

9.1.3.5.5. YARN - Queues

Metrics to see status of Queues on the YARN cluster.

Row	Metrics	Description
NUM APPS	Apps Running	Current number of running applications.
	Apps Pending	Current number of pending applications.
	Apps Completed	Accumulated number of completed applications over time.
	Apps Failed	Accumulated number of failed applications over time.
	Apps Killed	Accumulated number of killed applications over time.
	Apps Submitted	Accumulated number of submitted applications over time.
NUM CONTAINERS	Containers Running	Current number of running containers.
	Containers Pending	Current number of pending containers.
	Containers Reserved	Current number of Reserved containers.
	Total Containers Allocated	Accumulated number of containers allocated over time.
	Total Node Local Containers Allocated	Accumulated number of node-local containers allocated over time.
	Total Rack Local Containers Allocated	Accumulated number of rack-local containers allocated over time.
	Total OffSwitch Containers Allocated	Accumulated number of off-switch containers allocated over time.
MEMORY UTILIZATION	Allocated Memory	Current amount of memory allocated for containers.
	Pending Memory	Current amount of memory asked by applications for allocating containers.
	Available Memory	Current amount of memory available for allocating containers.
	Reserved Memory	Current amount of memory reserved for containers.
	Memory Used by AM	Current amount of memory used by AM containers.
CONTAINER ALLOCATION DELAY	Ave AM Container Allocation Delay	Average time taken to allocate an AM container since the AM container is requested.

9.1.3.5.6. YARN - ResourceManager

Metrics to see status of ResourceManagers on the YARN cluster.

Row	Metrics	Description
RPC STATS	RPC Avg Processing / Queue Time	Average time for processing/queuing a RPC call.
	RPC Call Queue Length	The length of the RPC call queue.
	RPC Slow calls	Number of slow RPC calls.

Row	Metrics	Description
MEMORY USAGE	Heap Mem Usage	Current heap memory usage.
	NonHeap Mem Usage	Current non-heap memory usage.
GC STATS	GC count	Accumulated GC count over time.
	GcTime	Accumulated GC time over time.
LOG ERRORS	Log Error / Fatal	Number of ERROR/FATAL logs.
AUTHORIZATION & AUTHENTICATION FAILURES	RPC Authorization Failures	Number of authorization failures.
	RPC Authentication Failures	Number of authentication failures.

9.1.3.5.7. YARN - TimelineServer

Metrics to see the overall status for TimelineServer.

Row	Metrics	Description
DATA READS	Timeline Entity Data Reads	Accumulated number of read operations.
	Timeline Entity Data Read time	Average time for reading a timeline entity.
DATA WRITES	Timeline Entity Data Write	Accumulated number of write operations.
	Timeline Entity Data Write Time	Average time for writing a timeline entity.
JVM METRICS	GC Count	Accumulated GC count over time.
	GC Time	Accumulated GC time over time.
	Heap Usage	Current heap memory usage.
	NonHeap Usage	Current non-heap memory usage.

9.1.3.6. Hive Dashboards

The following Grafana dashboards are available for Hive:

- [Hive - Home \[146\]](#)
- [Hive - HiveMetaStore \[147\]](#)
- [Hive - HiveServer2 \[147\]](#)

9.1.3.6.1. Hive - Home

Metrics that show the overall status for Hive service.

Row	Metrics	Description
WAREHOUSE SIZE - AT A GLANCE	DB count at startup	Number of databases present at the last warehouse service startup time.
	Table count at startup	Number of tables present at the last warehouse service startup time.
	Partition count at startup	Number of partitions present at the last warehouse service startup time.

Row	Metrics	Description
WAREHOUSE SIZE - REALTIME GROWTH	#tables created (ongoing)	Number of tables created since the last warehouse service startup.
	#partitions created (ongoing)	Number of partitions created since the last warehouse service startup.
MEMORY PRESSURE	HiveMetaStore Memory - Max	Heap memory usage by Hive MetaStores. If applicable, indicates max usage across multiple instances.
	HiveServer2 Memory - Max	Heap memory usage by HiveServer2. If applicable, indicates max usage across multiple instances.
	HiveMetaStore Offheap Memory - Max	Non-heap memory usage by Hive MetaStores. If applicable, indicates max usage across multiple instances.
	HiveServer2 Offheap Memory - Max	Non-heap memory usage by HiveServer2. If applicable, indicates max across multiple instances.
	HiveMetaStore app stop times (due to GC stops)	Total time spent in application pauses caused by garbage collection across Hive MetaStores.
	HiveServer2 app stop times (due to GC stops)	Total time spent in application pauses caused by garbage collection across HiveServer2.
METASTORE - CALL TIMES	API call times - Health Check roundtrip (get_all_databases)	Time taken to process a low-cost call made by health checks to all metastores.
	API call times - Moderate size call (get_partitions_by_name)	Time taken to process a moderate-cost call made by queries/exports/etc to all metastores. Data for this metric may not be available in a less active warehouse.

9.1.3.6.2. Hive - HiveMetaStore

Metrics that show operating status for HiveMetaStore hosts. Select a HiveMetaStore and a host to view relevant metrics.

Row	Metrics	Description
API TIMES	API call times - Health Check roundtrip (get_all_databases)	Time taken to process a low-cost call made by health checks to this metastore.
	API call times - Moderate size call (get_partitions_by_name)	Time taken to process a moderate-cost call made by queries/exports/etc to this metastore. Data for this metric may not be available in a less active warehouse.
MEMORY PRESSURE	App Stop times (due to GC)	Time spent in application pauses caused by garbage collection.
	Heap Usage	Current heap memory usage.
	Off-Heap Usage	Current non-heap memory usage.

9.1.3.6.3. Hive - HiveServer2

Metrics that show operating status for HiveServer2 hosts. Select a HiveServer2 and a host to view relevant metrics.

Row	Metrics	Description
API TIMES	API call times - Health Check	Time taken to process a low-cost call made by health checks to the metastore embedded in this HiveServer2. Data for this

Row	Metrics	Description
	roundtrip (get_all_databases)	metric may not be available if HiverServer2 is not running in an embedded-metastore mode.
	API call times - Moderate size call (get_partitions_by_name)	Time taken to process a moderate-cost call made by queries/exports/etc to the metastore embedded in this HiveServer2. Note: for this metric may not be available in a less active warehouse, or if HiveServer2 is not running in an embedded-metastore mode.
MEMORY PRESSURE	App Stop times (due to GC)	Time spent in application pauses caused by garbage collection.
	Heap Usage	Current heap memory usage.
	Off-Heap Usage	Current non-heap memory usage.
THREAD STATES	Active operation count	Current number of active operations in HiveServer2 and their running states.
	Completed operation states	Number of completed operations on HiveServer2 since the last restart. Indicates whether they completed as expected or encountered errors.

9.1.3.7. Hive LLAP Dashboards

The following Grafana dashboards are available for Apache Hive LLAP. The LLAP Heat map dashboard and the LLAP Overview dashboard enable you to quickly see the hotspots among the LLAP daemons. If you find an issue and want to navigate to more specific information for a specific system, use the LLAP Daemon dashboard.

Note that all Hive LLAP dashboards show the state of the cluster and are useful for looking at cluster information from the previous hour or day. The dashboards do not show real-time results.

- [Hive LLAP - Heatmap \[148\]](#)
- [Hive LLAP - Overview \[149\]](#)
- [Hive LLAP - Daemon \[151\]](#)

9.1.3.7.1. Hive LLAP - Heatmap

The heat map dashboard shows all the nodes that are running LLAP daemons and includes a percentage summary for available executors and cache. This dashboard enables you to identify the hotspots in the cluster in terms of executors and cache.

The values in the table are color coded based on threshold: if the threshold is more than 50%, the color is green; between 20% and 50%, the color is yellow; and less than 20%, the color is red.

Row	Metrics	Description
Heat maps	Remaining Cache Capacity	Shows the percentage of cache capacity remaining across the nodes. For example, if the grid is green, the cache is being under utilized. If the grid is red, there is high utilization of cache.
	Remaining Cache Capacity	Same as above (Remaining Cache Capacity), but shows the cache hit ratio.
	Executor Free Slots	Shows the percentage of executor free slots that are available on each nodes.

9.1.3.7.2. Hive LLAP - Overview

The overview dashboard shows the aggregated information across all of the clusters: for example, the total cache memory from all the nodes. This dashboard enables you to see that your cluster is configured and running correctly. For example, you might have configured 10 nodes but you see only 8 nodes running.

If you find an issue by viewing this dashboard, you can open the LLAP Daemon dashboard to see which node is having the problem.

Row	Metrics	Description
Overview	Total Executor Threads	Shows the total number of executors across all nodes.
	Total Executor Memory	Shows the total amount of memory for executors across all nodes.
	Total Cache Memory	Shows the total amount of memory for cache across all nodes.
	Total JVM Memory	Shows the total amount of max Java Virtual Machine (JVM) memory across all nodes.
Cache Metrics Across all nodes	Total Cache Usage	Shows the total amount of cache usage (Total, Remaining, and Used) across all nodes.
	Average Cache Hit Rate	As the data is released from the cache, the curve should increase. For example, the first query should run at 0, the second at 80-90 seconds, and then the third 10% faster. If, instead, it decreases, there might be a problem in the cluster.
	Average Cache Read Requests	Shows how many requests are being made for the cache and how many queries you are able to run that make use of the cache. If it says 0, for example, your cache might not be working properly and this grid might reveal a configuration issue.
Cache Metrics Across all nodes	Total Cache Usage	Shows the total amount of cache usage (Total, Remaining, and Used) across all nodes.
	Average Cache Hit Rate	As the data is released from the cache, the curve should increase. For example, the first query should run at 0, the second at 80-90 seconds, and then the third 10% faster. If, instead, it decreases, there might be a problem in the cluster.
	Average Cache Read Requests	Shows how many requests are being made for the cache and how many queries you are able to run that make use of the cache. If it says 0, for example, your cache might not be working properly and this grid might reveal a configuration issue.
Executor Metrics Across All nodes	Total Executor Requests	Shows the total number of task requests that were handled, succeeded, failed, killed, evicted and rejected across all nodes. Handled: Total requests across all sub-groups Succeed: Total requests that were processed. For example, if you have 8 core machines, the number of total executor requests would be 8 Failed: Did not complete successfully because, for example, you ran out of memory Rejected: If all task priorities are the same, but there are still not enough slots to fulfill the request, the system will reject some tasks Evicted: Lower priority requests are evicted if the slots are filled by higher priority requests

Row	Metrics	Description
	Total Execution Slots	Shows the total execution slots, the number of free or available slots, and number of slots occupied in the wait queue across all nodes. Ideally, the threads available (blue) result should be the same as the threads that are occupied in the queue result.
	Time to Kill Pre-empted Task (300s interval)	Shows the time that it took to kill a query due to pre-emption in percentile (50th, 90th, 99th) latencies in 300 second intervals.
	Max Time To Kill Task (due to preemption)	Shows the maximum time taken to kill a task due to pre-emption. This grid and the one above show you if you are wasting a lot of time killing queries. Time lost while a task is waiting to be killed is time lost in the cluster. If your max time to kill is high, you might want to disable this feature.
	Pre-emption Time Lost (300s interval)	Shows the time lost due to pre-emption in percentile (50th, 90th, 99th) latencies in 300 second intervals.
	Max Time Lost In Cluster (due to pre-emption)	Shows the maximum time lost due to pre-emption. If your max time to kill is high, you might want to disable this feature.
IO Elevator Metrics Across All Nodes	Column Decoding Time (30s interval)	Shows the percentile (50th, 90th, 99th) latencies for time it takes to decode the column chunk (convert encoded column chunk to column vector batches for processing) in 30 second intervals. The cache comes from IO Elevator. It loads data from HDFS to the cache, and then from the cache to the executor. This metric shows how well the threads are performing and is useful to see that the threads are running.
	Max Column Decoding Time	Shows the maximum time taken to decode column chunk (convert encoded column chunk to column vector batches for processing).
JVM Metrics across all nodes	Average JVM Heap Usage	Shows the average amount of Java Virtual Machine (JVM) heap memory used across all nodes. If the heap usage keeps increasing, you might run out of memory and the task failure count would also increase.
	Average JVM Non-Heap Usage	Shows the average amount of JVM non-heap memory used across all nodes.
	Max GcTotalExtraSleepTime	Shows the maximum garbage collection extra sleep time in milliseconds across all nodes. Garbage collection extra sleep time measures when the garbage collection monitoring is delayed (for example, the thread does not wake up after 500 milliseconds).
	Max GcTimeMillis	Shows the total maximum GC time in milliseconds across all nodes.
	Total JVM Threads	Shows the total number of JVM threads that are in a NEW, RUNNABLE, WAITING, TIMED_WAITING, and TERMINATED state across all nodes.
JVM Metrics	Total JVM Heap Used	Shows the total amount of Java Virtual Machine (JVM) heap memory used in the daemon. If the heap usage keeps increasing, you might run out of memory and the task failure count would also increase.
	Total JVM Non-Heap Used	Shows the total amount of JVM non-heap memory used in the LLAP daemon.

Row	Metrics	Description
		If the non-heap memory is over-allocated, you might run out of memory and the task failure count would also increase.
	Max GcTotalExtraSleepTime	Shows the maximum garbage collection extra sleep time in milliseconds in the LLAP daemon. Garbage collection extra sleep time measures when the garbage collection monitoring is delayed (for example, the thread does not wake up after 500 milliseconds).
	Max GcTimeMillis	Shows the total maximum GC time in milliseconds in the LLAP daemon.
	Max JVM Threads Runnable	Shows the maximum number of Java Virtual Machine (JVM) threads that are in RUNNABLE state.
	Max JVM Threads Blocked	Shows the maximum number of JVM threads that are in BLOCKED state. If you are seeing spikes in the threads blocked, you might have a problem with your LLAP daemon.
	Max JVM Threads Waiting	Shows the maximum number of JVM threads that are in WAITING state.
	Max JVM Threads Timed Waiting	Shows the maximum number of JVM threads that are in TIMED_WAITING state.

9.1.3.7.3. Hive LLAP - Daemon

Metrics that show operating status for Hive LLAP Daemons.

Row	Metrics	Description
	Total Requests Submitted	Shows the total number of task requests handled by the daemon.
	Total Requests Succeeded	Shows the total number of successful task requests handled by the daemon.
	Total Requests Failed	Shows the total number of failed task requests handled by the daemon.
	Total Requests Killed	Shows the total number of killed task requests handled by the daemon.
	Total Requests Evicted From Wait Queue	Shows the total number of task requests handled by the daemon that were evicted from the wait queue. Tasks are evicted if all of the executor threads are in use by higher priority tasks.
	Total Requests Rejected	Shows the total number of task requests handled by the daemon that were rejected by the task executor service. Task are rejected if all of the executor threads are in use and the wait queue is full of tasks that are not eligible for eviction.
Executor Metrics	Available Execution Slots	Shows the total number of free slots that are available for execution including free executor threads and free slots in the wait queue.
	95th Percentile Pre-emption Time Lost (300s interval)	Shows the 95th percentile latencies for time lost due to pre-emption in 300 second intervals.
	Max Pre-emption Time Lost	Shows the maximum time lost due to pre-emption.
	95th Percentile Time to Kill Pre-empted Task (300s interval)	Shows the 95th percentile latencies for time taken to kill tasks due to pre-emption in 300 second intervals.
	Max Time To Kill Task Pre-empted Task	Shows the maximum time taken to kill a task due to pre-emption.

Row	Metrics	Description
Cache Metrics	Total Cache Used	Shows the total amount of cache usage (Total, Remaining, and Used) in LLAP daemon cache.
	Heap Usage	Shows the amount of memory remaining in LLAP daemon cache.
	Average Cache Hit Rate	As the data is released from the cache, the curve should increase. For example, the first query should run at 0, the second at 80-90 seconds, and then the third 10% faster. If, instead, it decreases, there might be a problem in the LLAP daemon.
	Total Cache Read Requests	Shows the total number of read requests received by LLAP daemon cache.
THREAD STATES	95th Percentile Column Decoding Time (30s interval)	Shows the 95th percentile latencies for time it takes to decode the column chunk (convert encoded column chunk to column vector batches for processing) in 30 second intervals. The cache comes from IO Elevator. It loads data from HDFS to the cache, and then from the cache to the executor. This metric shows how well the threads are performing and is useful to see that the threads are running.
	Max Column Decoding Time	Shows the maximum time taken to decode column chunk (convert encoded column chunk to column vector batches for processing).

9.1.3.8. HBase Dashboards

Monitoring an HBase cluster is essential for maintaining a high-performance and stable system. The following Grafana dashboards are available for HBase:

- [HBase - Home \[152\]](#)
- [HBase - RegionServers \[153\]](#)
- [HBase - Misc \[158\]](#)
- [HBase - Tables \[159\]](#)
- [HBase - Users \[161\]](#)



Important

Ambari disables per-region, per-table, and per-user metrics for HBase by default. See [Enabling Individual Region, Table, and User Metrics for HBase](#) if you want the Ambari Metrics System to display the more granular metrics of HBase system performance on the individual region, table, or user level.

9.1.3.8.1. HBase - Home

The HBase - Home dashboards display basic statistics about an HBase cluster. These dashboards provide insight to the overall status for the HBase cluster.

Row	Metrics	Description
REGIONSERVERS / REGIONS	Num RegionServers	Total number of RegionServers in the cluster.
	Num Dead RegionServers	Total number of RegionServers that are dead in the cluster.
	Num Regions	Total number of regions in the cluster.

Row	Metrics	Description
	Avg Num Regions per RegionServer	Average number of regions per RegionServer.
NUM REGIONS/STORES	Num Regions / Stores - Total	Total number of regions and stores (column families) in the cluster.
	Store File Size / Count - Total	Total data file size and number of store files.
NUM REQUESTS	Num Requests - Total	Total number of requests (read, write and RPCs) in the cluster.
	Num Request - Breakdown - Total	Total number of get,put,mutate,etc requests in the cluster.
REGIONSERVER MEMORY	RegionServer Memory - Average	Average used, max or committed on-heap and offheap memory for RegionServers.
	RegionServer Offheap Memory - Average	Average used, free or committed on-heap and offheap memory for RegionServers.
MEMORY - MEMSTORE BLOCKCACHE	Memstore - BlockCache - Average	Average blockcache and memstore sizes for RegionServers.
	Num Blocks in BlockCache - Total	Total number of (hfile) blocks in the blockcaches across all RegionServers.
BLOCKCACHE	BlockCache Hit/Miss/s Total	Total number of blockcache hits misses and evictions across all RegionServers.
	BlockCache Hit Percent - Average	Average blockcache hit percentage across all RegionServers.
OPERATION LATENCIES - GET/MUTATE	Get Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Get operation across all RegionServers.
	Mutate Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Mutate operation across all RegionServers.
OPERATION LATENCIES - DELETE/INCREMENT	Delete Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Delete operation across all RegionServers.
	Increment Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Increment operation across all RegionServers.
OPERATION LATENCIES - APPEND/REPLAY	Append Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Append operation across all RegionServers.
	Replay Latencies - Average	Average min, median, max, 75th, 95th, 99th percentile latencies for Replay operation across all RegionServers.
REGIONSERVER RPC	RegionServer RPC - Average	Average number of RPCs, active handler threads and open connections across all RegionServers.
	RegionServer RPC Queues - Average	Average number of calls in different RPC scheduling queues and the size of all requests in the RPC queue across all RegionServers.
REGIONSERVER RPC	RegionServer RPC Throughput - Average	Average sent and received bytes from the RPC across all RegionServers.

9.1.3.8.2. HBase - RegionServers

The HBase - RegionServers dashboards display metrics for RegionServers in the monitored HBase cluster, including some performance-related data. These dashboards help you view basic I/O data and compare load among RegionServers.

Row	Metrics	Description
NUM REGIONS	Num Regions	Number of regions in the RegionServer.
STORE FILES	Store File Size	Total size of the store files (data files) in the RegionServer.

Row	Metrics	Description
	Store File Count	Total number of store files in the RegionServer.
NUM REQUESTS	Num Total Requests /s	Total number of requests (both read and write) per second in the RegionServer.
	Num Write Requests /s	Total number of write requests per second in the RegionServer.
	Num Read Requests /s	Total number of read requests per second in the RegionServer.
NUM REQUESTS - GET / SCAN	Num Get Requests /s	Total number of Get requests per second in the RegionServer.
	Num Scan Next Requests /s	Total number of Scan requests per second in the RegionServer.
NUM REQUESTS - MUTATE / DELETE	Num Mutate Requests - /s	Total number of Mutate requests per second in the RegionServer.
	Num Delete Requests /s	Total number of Delete requests per second in the RegionServer.
NUM REQUESTS - APPEND / INCREMENT	Num Append Requests /s	Total number of Append requests per second in the RegionServer.
	Num Increment Requests /s	Total number of Increment requests per second in the RegionServer.
	Num Replay Requests /s	Total number of Replay requests per second in the RegionServer.
MEMORY	RegionServer Memory Used	Heap Memory used by the RegionServer.
	RegionServer Offheap Memory Used	Offheap Memory used by the RegionServer.
MEMSTORE	Memstore Size	Total Memstore memory size of the RegionServer.
BLOCKCACHE - OVERVIEW	BlockCache - Size	Total BlockCache size of the RegionServer.
	BlockCache - Free Size	Total free space in the BlockCache of the RegionServer.
	Num Blocks in Cache	Total number of hfile blocks in the BlockCache of the RegionServer.
BLOCKCACHE - HITS/MISSES	Num BlockCache Hits /s	Number of BlockCache hits per second in the RegionServer.
	Num BlockCache Misses /s	Number of BlockCache misses per second in the RegionServer.
	Num BlockCache Evictions /s	Number of BlockCache evictions per second in the RegionServer.
	BlockCache Caching Hit Percent	Percentage of BlockCache hits per second for requests that requested cache blocks in the RegionServer.
	BlockCache Hit Percent	Percentage of BlockCache hits per second in the RegionServer.
OPERATION LATENCIES - GET	Get Latencies - Mean	Mean latency for Get operation in the RegionServer.
	Get Latencies - Median	Median latency for Get operation in the RegionServer.
	Get Latencies - 75th Percentile	75th percentile latency for Get operation in the RegionServer.
	Get Latencies - 95th Percentile	95th percentile latency for Get operation in the RegionServer.
	Get Latencies - 99th Percentile	99th percentile latency for Get operation in the RegionServer.

Row	Metrics	Description
	Get Latencies - Max	Max latency for Get operation in the RegionServer.
OPERATION LATENCIES - SCAN NEXT	Scan Next Latencies - Mean	Mean latency for Scan operation in the RegionServer.
	Scan Next Latencies - Median	Median latency for Scan operation in the RegionServer.
	Scan Next Latencies - 75th Percentile	75th percentile latency for Scan operation in the RegionServer.
	Scan Next Latencies - 95th Percentile	95th percentile latency for Scan operation in the RegionServer.
	Scan Next Latencies - 99th Percentile	99th percentile latency for Scan operation in the RegionServer.
	Scan Next Latencies - Max	Max latency for Scan operation in the RegionServer.
OPERATION LATENCIES - MUTATE	Mutate Latencies - Mean	Mean latency for Mutate operation in the RegionServer.
	Mutate Latencies - Median	Median latency for Mutate operation in the RegionServer.
	Mutate Latencies - 75th Percentile	75th percentile latency for Mutate operation in the RegionServer.
	Mutate Latencies - 95th Percentile	95th percentile latency for Mutate operation in the RegionServer.
	Mutate Latencies - 99th Percentile	99th percentile latency for Mutate operation in the RegionServer.
	Mutate Latencies - Max	Max latency for Mutate operation in the RegionServer.
OPERATION LATENCIES - DELETE	Delete Latencies - Mean	Mean latency for Delete operation in the RegionServer.
	Delete Latencies - Median	Median latency for Delete operation in the RegionServer.
	Delete Latencies - 75th Percentile	75th percentile latency for Delete operation in the RegionServer.
	Delete Latencies - 95th Percentile	95th percentile latency for Delete operation in the RegionServer.
	Delete Latencies - 99th Percentile	99th percentile latency for Delete operation in the RegionServer.
	Delete Latencies - Max	Max latency for Delete operation in the RegionServer.
OPERATION LATENCIES - INCREMENT	Increment Latencies - Mean	Mean latency for Increment operation in the RegionServer.
	Increment Latencies - Median	Median latency for Increment operation in the RegionServer.
	Increment Latencies - 75th Percentile	75th percentile latency for Increment operation in the RegionServer.
	Increment Latencies - 95th Percentile	95th percentile latency for Increment operation in the RegionServer.
	Increment Latencies - 99th Percentile	99th percentile latency for Increment operation in the RegionServer.

Row	Metrics	Description
	Increment Latencies - Max	Max latency for Increment operation in the RegionServer.
OPERATION LATENCIES - APPEND	Append Latencies - Mean	Mean latency for Append operation in the RegionServer.
	Append Latencies - Median	Median latency for Append operation in the RegionServer.
	Append Latencies - 75th Percentile	75th percentile latency for Append operation in the RegionServer.
	Append Latencies - 95th Percentile	95th percentile latency for Append operation in the RegionServer.
	Append Latencies - 99th Percentile	99th percentile latency for Append operation in the RegionServer.
	Append Latencies - Max	Max latency for Append operation in the RegionServer.
OPERATION LATENCIES - REPLAY	Replay Latencies - Mean	Mean latency for Replay operation in the RegionServer.
	Replay Latencies - Median	Median latency for Replay operation in the RegionServer.
	Replay Latencies - 75th Percentile	75th percentile latency for Replay operation in the RegionServer.
	Replay Latencies - 95th Percentile	95th percentile latency for Replay operation in the RegionServer.
	Replay Latencies - 99th Percentile	99th percentile latency for Replay operation in the RegionServer.
	Replay Latencies - Max	Max latency for Replay operation in the RegionServer.
RPC - OVERVIEW	Num RPC /s	Number of RPCs per second in the RegionServer.
	Num Active Handler Threads	Number of active RPC handler threads (to process requests) in the RegionServer.
	Num Connections	Number of connections to the RegionServer.
RPC - QUEUES	Num RPC Calls in General Queue	Number of RPC calls in the general processing queue in the RegionServer.
	Num RPC Calls in Priority Queue	Number of RPC calls in the high priority (for system tables) processing queue in the RegionServer.
	Num RPC Calls in Replication Queue	Number of RPC calls in the replication processing queue in the RegionServer.
	RPC - Total Call Queue Size	Total data size of all RPC calls in the RPC queues in the RegionServer.
RPC - CALL QUEUED TIMES	RPC - Call Queued Time - Mean	Mean latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - Median	Median latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - 75th Percentile	75th percentile latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - 95th Percentile	95th percentile latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - 99th Percentile	99th percentile latency for RPC calls to stay in the RPC queue in the RegionServer.
	RPC - Call Queued Time - Max	Max latency for RPC calls to stay in the RPC queue in the RegionServer.

Row	Metrics	Description
RPC - CALL PROCESS TIMES	RPC - Call Process Time - Mean	Mean latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - Median	Median latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - 75th Percentile	75th percentile latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - 95th Percentile	95th percentile latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - 99th Percentile	99th percentile latency for RPC calls to be processed in the RegionServer.
	RPC - Call Process Time - Max	Max latency for RPC calls to be processed in the RegionServer.
RPC - THROUGHPUT	RPC - Received bytes /s	Received bytes from the RPC in the RegionServer.
	RPC - Sent bytes /s	Sent bytes from the RPC in the RegionServer.
WAL - FILES	Num WAL - Files	Number of Write-Ahead-Log files in the RegionServer.
	Total WAL File Size	Total files sized of Write-Ahead-Logs in the RegionServer.
WAL - THROUGHPUT	WAL - Num Appends /s	Number of append operations per second to the filesystem in the RegionServer.
	WAL - Num Sync /s	Number of sync operations per second to the filesystem in the RegionServer.
WAL - SYNC LATENCIES	WAL - Sync Latencies - Mean	Mean latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - Median	Median latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - 75th Percentile	75th percentile latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - 95th Percentile	95th percentile latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - 99th Percentile	99th percentile latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
	WAL - Sync Latencies - Max	Max latency for Write-Ahead-Log sync operation to the filesystem in the RegionServer.
WAL - APPEND LATENCIES	WAL - Append Latencies - Mean	Mean latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - Median	Median latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - 75th Percentile	95th percentile latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - 95th Percentile	95th percentile latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - 99th Percentile	99th percentile latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - Max	Max latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.

Row	Metrics	Description
WAL - APPEND SIZES	WAL - Append Sizes - Mean	Mean data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - Median	Median data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - 75th Percentile	75th percentile data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - 95th Percentile	95th percentile data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - 99th Percentile	99th percentile data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Sizes - Max	Max data size for Write-Ahead-Log append operation to the filesystem in the RegionServer.
SLOW OPERATIONS	WAL Num Slow Append /s	Number of append operations per second to the filesystem that took more than 1 second in the RegionServer.
	Num Slow Gets /s	Number of Get requests per second that took more than 1 second in the RegionServer.
	Num Slow Puts /s	Number of Put requests per second that took more than 1 second in the RegionServer.
	Num Slow Deletes /s	Number of Delete requests per second that took more than 1 second in the RegionServer.
FLUSH/COMPACTION QUEUES	Flush Queue Length	Number of Flush operations waiting to be processed in the RegionServer. A higher number indicates flush operations being slow.
	Compaction Queue Length	Number of Compaction operations waiting to be processed in the RegionServer. A higher number indicates compaction operations being slow.
	Split Queue Length	Number of Region Split operations waiting to be processed in the RegionServer. A higher number indicates split operations being slow.
JVM - GC COUNTS	GC Count /s	Number of Java Garbage Collections per second.
	GC Count ParNew /s	Number of Java ParNew (YoungGen) Garbage Collections per second.
	GC Count CMS /s	Number of Java CMS Garbage Collections per second.
JVM - GC TIMES	GC Times /s	Total time spend in Java Garbage Collections per second.
	GC Times ParNew /s	Total time spend in Java ParNew(YoungGen) Garbage Collections per second.
	GC Times CMS /s	Total time spend in Java CMS Garbage Collections per second.
LOCALITY	Percent Files Local	Percentage of files served from the local DataNode for the RegionServer.

9.1.3.8.3. HBase - Misc

The HBase - Misc dashboards display miscellaneous metrics related to the HBase cluster. You can use these metrics for tasks like debugging authentication and authorization issues and exceptions raised by RegionServers.

Row	Metrics	Description
REGIONS IN TRANSITION	Master - Regions in Transition	Number of regions in transition in the cluster.
	Master - Regions in Transition Longer	Number of regions in transition that are in transition state for longer than 1 minute in the cluster.

Row	Metrics	Description
	Than Threshold Time	
	Regions in Transition Oldest Age	Maximum time that a region stayed in transition state.
NUM THREADS - RUNNABLE	Master Num Threads - Runnable	Number of threads in the Master.
	RegionServer Num Threads - Runnable	Number of threads in the RegionServer.
NUM THREADS - BLOCKED	Master Num Threads - Blocked	Number of threads in the Blocked State in the Master.
	RegionServer Num Threads - Blocked	Number of threads in the Blocked State in the RegionServer.
NUM THREADS - WAITING	Master Num Threads - Waiting	Number of threads in the Waiting State in the Master.
	RegionServer Num Threads - Waiting	Number of threads in the Waiting State in the RegionServer.
NUM THREADS - TIMED WAITING	Master Num Threads - Timed Waiting	Number of threads in the Timed-Waiting State in the Master.
	RegionServer Num Threads - Timed Waiting	Number of threads in the Timed-Waiting State in the RegionServer.
NUM THREADS - NEW	Master Num Threads - New	Number of threads in the New State in the Master.
	RegionServer Num Threads - New	Number of threads in the New State in the RegionServer.
NUM THREADS - TERMINATED	Master Num Threads - Terminated	Number of threads in the Terminated State in the Master.
	RegionServer Num Threads - Terminated	Number of threads in the Terminated State in the RegionServer.
RPC AUTHENTICATION	RegionServer RPC Authentication Successes /s	Number of RPC successful authentications per second in the RegionServer.
	RegionServer RPC Authentication Failures /s	Number of RPC failed authentications per second in the RegionServer.
RPC Authorization	RegionServer RPC Authorization Successes /s	Number of RPC successful autorizations per second in the RegionServer.
	RegionServer RPC Authorization Failures /s	Number of RPC failed autorizations per second in the RegionServer.
EXCEPTIONS	Master Exceptions /s	Number of exceptions in the Master.
	RegionServer Exceptions /s	Number of exceptions in the RegionServer.

9.1.3.8.4. HBase - Tables

HBase - Tables metrics reflect data on the table level. The dashboards and data help you compare load distribution and resource use among tables in a cluster at different times.

Row	Metrics	Description
NUM REGIONS/STORES	Num Regions	Number of regions for the table(s).
	Num Stores	Number of stores for the table(s).
TABLE SIZE	Table Size	Total size of the data (store files and MemStore) for the table(s).
	Average Region Size	Average size of the region for the table(s). Average Region Size is calculated from average of average region sizes reported by each RegionServer (may not be the true average).
MEMSTORE SIZE	MemStore Size	Total MemStore size of the table(s).
STORE FILES	Store File Size	Total size of the store files (data files) for the table(s).
	Num Store Files	Total number of store files for the table(s).
STORE FILE AGE	Max Store File Age	Maximum age of store files for the table(s). As compactions rewrite data, store files are also rewritten. Max Store File Age is calculated from the maximum of all maximum store file ages reported by each RegionServer.
	Min Store File Age	Minimum age of store files for the table(s). As compactions rewrite data, store files are also rewritten. Min Store File Age is calculated from the minimum of all minimum store file ages reported by each RegionServer.
	Average Store File Age	Average age of store files for the table(s). As compactions rewrite data, store files are also rewritten. Average Store File Age is calculated from the average of average store file ages reported by each RegionServer.
	Num Reference Files - Total on All	Total number of reference files for the table(s).
NUM TOTAL REQUESTS	Num Total Requests /s on Tables	Total number of requests (both read and write) per second for the table(s).
NUM READ REQUESTS	Num Read Requests /s	Total number of read requests per second for the table(s).
NUM WRITE REQUESTS	Num Write Requests /s	Total number of write requests per second for the table(s).
NUM FLUSHES	Num Flushes /s	Total number of flushes per second for the table(s).
FLUSHED BYTES	Flushed MemStore Bytes	Total number of flushed MemStore bytes for the table(s).
	Flushed Output Bytes	Total number of flushed output bytes for the table(s).
FLUSH TIME HISTOGRAM	Flush Time Mean	Mean latency for Flush operation for the table(s).
	Flush Time Median	Median latency for Flush operation for the table(s).
	Flush Time 95th Percentile	95th percentile latency for Flush operation for the table(s).
	Flush Time Max	Maximum latency for Flush operation for the table(s).
FLUSH MEMSTORE SIZE HISTOGRAM	Flush MemStore Size Mean	Mean size of the MemStore for Flush operation for the table(s).
	Flush MemStore Size Median	Median size of the MemStore for Flush operation for the table(s).
	Flush Output Size 95th Percentile	95th percentile size of the MemStore for Flush operation for the table(s).
	Flush MemStore Size Max	Max size of the MemStore for Flush operation for the table(s).
FLUSH OUTPUT SIZE HISTOGRAM	Flush Output Size Mean	Mean size of the output file for Flush operation for the table(s).
	Flush Output Size Median	Median size of the output file for Flush operation for the table(s).

Row	Metrics	Description
	Flush Output Size 95th Percentile	95th percentile size of the output file for Flush operation for the table(s).
	Flush Output Size Max	Max size of the output file for Flush operation for the table(s).

9.1.3.8.5. HBase - Users

The HBase - Users dashboards display metrics and detailed data on a per-user basis across the cluster. You can click the second drop-down arrow in the upper-left corner to select a single user, a group of users, or all users, and you can change your user selection at any time.

Row	Metrics	Description
NUM REQUESTS - GET/SCAN	Num Get Requests /s	Total number of Get requests per second for the user(s).
	Num Scan Next Requests /s	Total number of Scan requests per second for the user(s).
NUM REQUESTS - MUTATE/DELETE	Num Mutate Requests /s	Total number of Mutate requests per second for the user(s).
	Num Delete Requests /s	Total number of Delete requests per second for the user(s).
NUM REQUESTS - APPEND/INCREMENT	Num Append Requests /s	Total number of Append requests per second for the user(s).
	Num Increment Requests /s	Total number of Increment requests per second for the user(s).

9.1.3.9. Kafka Dashboards

The following Grafana dashboards are available for Kafka:

- [Kafka - Home \[161\]](#)
- [Kafka - Hosts \[162\]](#)
- [Kafka - Topics \[162\]](#)

9.1.3.9.1. Kafka - Home

Metrics that show overall status for the Kafka cluster.

Row	Metrics	Description
BYTES IN & OUT / MESSAGES IN	Bytes In & Bytes Out /sec	Rate at which bytes are produced into the Kafka cluster and the rate at which bytes are being consumed from the Kafka cluster.
	Messages In /sec	Number of messages produced into the Kafka cluster.
CONTROLLER/LEADER COUNT & REPLICA MAXLAG	Active Controller Count	Number of active controllers in the Kafka cluster. This should always equal one.
	Replica MaxLag	Shows the lag of each replica from the leader.
	Leader Count	Number of partitions for which a particular host is the leader.
UNDER REPLICATED PARTITIONS & OFFLINE PARTITIONS COUNT	Under Replicated Partitions	Indicates if any partitions in the cluster are under-replicated.
	Offline Partitions Count	Indicates if any partitions are offline (which means that no leaders or replicas are available for producing or consuming).
PRODUCER & CONSUMER REQUESTS	Producer Req /sec	Rate at which producer requests are made to the Kafka cluster.

Row	Metrics	Description
	Consumer Req /sec	Rate at which consumer requests are made from the Kafka cluster.
LEADER ELECTION AND UNCLEAN LEADER ELECTIONS	Leader Election Rate	Rate at which leader election is happening in the Kafka cluster.
	Unclean Leader Elections	Indicates if there are any unclean leader elections. Unclean leader election indicates that a replica which is not part of ISR is elected as a leader.
ISR SHRINKS / ISR EXPANDED	IsrShrinksPerSec	If the broker goes down, ISR shrinks. In such case, this metric indicates if any of the partitions are not part of ISR.
	IsrExpandsPerSec	Once the broker comes back up and catches up with the leader, this metric indicates if any partitions rejoined ISR.
REPLICA FETCHER MANAGER	ReplicaFetcherManagerMaxLag	The maximum lag in messages between the follower and leader replicas.

9.1.3.9.2. Kafka - Hosts

Metrics that show operating status for Kafka cluster on a per broker level.

Use the drop-down menus to customize your results:

- Kafka broker
- Host
- Whether to view the largest (top) or the smallest (bottom) values
- Number of values that you want to view
- Aggregator to use: average, max value, or the sum of values

Row	Metrics	Description
BYTES IN & OUT / MESSAGES IN / UNDER REPLICATED PARTITIONS	Bytes In & Bytes Out /sec	Rate at which bytes produced into the Kafka broker and rate at which bytes are being consumed from the Kafka broker.
	Messages In /sec	Number of messages produced into the Kafka broker.
	Under Replicated Partitions	Number of under-replicated partitions in the Kafka broker.
PRODUCER & CONSUMER REQUESTS	Producer Req /sec	Rate at which producer requests are made to the Kafka broker.
	Consumer Req /sec	Rate at which consumer requests are made from the Kafka broker.
REPLICA MANAGER PARTITION/ LEADER/FETCHER MANAGER MAX LAG	Replica Manager Partition Count	Number of topic partitions being replicated for the Kafka broker.
	Replica Manager Leader Count	Number of topic partitions for which the Kafka broker is the leader.
	Replica Fetcher Manager MaxLag clientId Replica	Shows the lag in replicating topic partitions.
ISR SHRINKS / ISR EXPANDS	IsrShrinks /sec	Indicates if any replicas failed to be in ISR for the host.
	IsrExpands /sec	Indicates if any replica has caught up with leader and re-joined the ISR for the host.

9.1.3.9.3. Kafka - Topics

Metrics related to Kafka cluster on a per topic level. Select a topic (by default, all topics are selected) to view the metrics for that topic.

Row	Metrics	Description
MESSAGES IN/OUT & BYTES IN/OUT	MessagesInPerSec	Rate at which messages are being produced into the topic.
	MessagesOutPerSec	Rate at which messages are being consumed from the topic.
TOTAL FETCH REQUESTS	TotalFetchRequestsPerSec	Number of consumer requests coming for the topic.
TOTAL PRODUCE REQUESTS /SEC	TotalProduceRequestsPerSec	Number of producer requests being sent to the topic.
FETCHER LAG METRICS CONSUMER LAG	FetcherLagMetrics ConsumerLag	Shows the replica fetcher lag for the topic.

9.1.3.10. Storm Dashboards

The following Grafana dashboards are available for Storm:

- [Storm - Home \[163\]](#)
- [Storm - Topology \[163\]](#)
- [Storm - Components \[164\]](#)

9.1.3.10.1. Storm - Home

Metrics that show the operating status for Storm.

Row	Metrics	Description
Unnamed	Topologies	Number of topologies in the cluster.
	Supervisors	Number of supervisors in the cluster.
	Total Executors	Total number of executors running for all topologies in the cluster.
	Total Tasks	Total number of tasks for all topologies in the cluster.
Unnamed	Free Slots	Number of free slots for all supervisors in the cluster.
	Used Slots	Number of used slots for all supervisors in the cluster.
	Total Slots	Total number of slots for all supervisors in the cluster. Should be more than 0.

9.1.3.10.2. Storm - Topology

Metrics that show the overall operating status for Storm topologies. Select a topology (by default, all topologies are selected) to view metrics for that topology.

Row	Metrics	Description
RECORDS	All Tasks Input/Output	Input Records is the number of input messages executed on all tasks, and Output Records is the number of messages emitted on all tasks.
	All Tasks Acked Tuples	Number of messages acked (completed) on all tasks.
	All Tasks Failed Tuples	Number of messages failed on all tasks.
LATENCY / QUEUE	All Spouts Latency	Average latency on all spout tasks.
	All Tasks Queue	Receive Queue Population is the total number of tuples waiting in the receive queue, and Send Queue Population is the total number of tuples waiting in the send queue.
MEMORY USAGE	All workers memory usage on heap	Used bytes on heap for all workers in topology.

Row	Metrics	Description
	All workers memory usage on non-heap	Used bytes on non-heap for all workers in topology.
GC	All workers GC count	PSScavenge count is the number of occurrences for parallel scavenge collector and PSMarkSweep count is the number of occurrences for parallel scavenge mark and sweep collector.
	All workers GC time	PSScavenge timeMs is the sum of the time parallel scavenge collector takes (in milliseconds), and PSMarkSweep timeMs is the sum of the time parallel scavenge mark and sweep collector takes (in milliseconds). Note that GC metrics are provided based on worker GC setting, so these metrics are only available for default GC option for worker.childopts. If you use another GC option for worker, you need to copy the dashboard and update the metric name manually.

9.1.3.10.3. Storm - Components

Metrics that show operating status for Storm topologies on a per component level. Select a topology and a component to view related metrics.

Row	Metrics	Description
RECORDS	Input/Output	Input Records is the number of messages executed on the selected component, and Output Records is the number of messages emitted on the selected component.
	Acked Tuples	Number of messages acked (completed) on the selected component.
	Failed Tuples	Number of messages failed on the selected component.
LATENCY / QUEUE	Latency	Complete Latency is the average complete latency on the select component (for Spout), and Process Latency is the average process latency on the selected component (for Bolt).
	Queue	Receive Queue Population is the total number of tuples waiting in receive queues on the selected component, and Send Queue Population is the total number of tuples waiting in send queues on the selected component.

9.1.3.11. System Dashboards

The following Grafana dashboards are available for System:

- [System - Home \[164\]](#)
- [System - Servers \[165\]](#)

9.1.3.11.1. System - Home

Metrics to see the overall status of the cluster.

Row	Metrics	Description
OVERVIEW AVERAGES	Logical CPU Count Per Server	Average number of CPUs (including hyperthreading) aggregated for selected hosts.
	Total Memory Per Server	Total system memory available per server aggregated for selected hosts.
	Total Disk Space Per Server	Total disk space per server aggregated for selected hosts.
OVERVIEW - TOTALS	Logical CPU Count Total	Total Number of CPUs (including hyperthreading) aggregated for selected hosts.

Row	Metrics	Description
	Total Memory	Total system memory available per server aggregated for selected hosts.
	Total Disk Space	Total disk space per server aggregated for selected hosts.
CPU	CPU Utilization - Average	CPU utilization aggregated for selected hosts.
SYSTEM LOAD	System Load - Average	Load average (1 min, 5 min and 15 min) aggregated for selected hosts.
MEMORY	Memory - Average	Average system memory utilization aggregated for selected hosts.
	Memory - Total	Total system memory available aggregated for selected hosts.
DISK UTILITIZATION	Disk Utilitization - Average	Average disk usage aggregated for selected hosts.
	Disk Utilitization - Total	Total disk available for selected hosts.
DISK IO	Disk IO - Average (upper chart)	Disk read/write counts (iops) co-related with bytes aggregated for selected hosts.
	Disk IO - Average (lower chart)	Average Individual read/write statistics as MBps aggregated for selected hosts.
	Disk IO - Total	Sum of read/write bytes/sec aggregated for selected hosts.
NETWORK IO	Network IO - Average	Average Network statistics as MBps aggregated for selected hosts.
NETWORK PACKETS	Network IO - Total	Sum of Network packets as MBps aggregated for selected hosts.
	Network Packets - Average	Average of Network packets as KBps aggregated for selected hosts.
SWAP/NUM PROCESSES	Swap Space - Average	Average swap space statistics aggregated for selected hosts.
	Num Processes - Average	Average number of processes aggregated for selected hosts.



Note

- Average implies sum/count for values reported by all hosts in the cluster. Example: In a 30 second window, if 98 out of 100 hosts reported 1 or more value, it is the $SUM(Avg \text{ value from each host} + \text{Interpolated value for } 2 \text{ missing hosts})/100$.
- Sum/Total implies the sum of all values in a timeslice (30 seconds) from all hosts in the cluster. The same interpolation rule applies.

9.1.3.11.2. System - Servers

Metrics to see the system status per host on the server.

Row	Metrics	Description
CPU - USER/SYSTEM	CPU Utilization - User	CPU utilization per user for selected hosts.
	CPU Utilization - System	CPU utilization per system for selected hosts.
CPU - NICE/IDLE	CPU Utilization - Nice	CPU nice (Unix) time spent for selected hosts.

Row	Metrics	Description
	CPU Utilization - Idle	CPU idle time spent for selected hosts.
CPU - IOWAIT/INTR	CPU Utilization - iowait	CPU IO wait time for selected hosts.
	CPU Utilization - Hardware Interrupt	CPU IO interrupt execute time for selected hosts.
CPU - SOFTINTR/STEAL	CPU Utilization - Software Interrupt	CPU time spent processing soft irq's for selected hosts.
	CPU Utilization - Steal (VM)	CPU time spent processing steal time (virtual cpu wait) for selected hosts.
SYSTEM LOAD - 1 MINUTE	System Load Average - 1 Minute	1 minute load average for selected hosts.
SYSTEM LOAD - 5 MINUTE	System Load Average - 5 Minute	5 minute load average for selected hosts.
SYSTEM LOAD - 15 MINUTE	System Load Average - 15 Minute	15 minute load average for selected hosts.
MEMORY - TOTAL/USED	Memory - Total	Total memory in GB for selected hosts.
	Memory - Used	Used memory in GB for selected hosts.
MEMORY - FREE/CACHED	Memory - Free	Total free memory in GB for selected hosts.
	Memory - Cached	Total cached memory in GB for selected hosts.
MEMORY - BUFFERED/SHARED	Memory - Buffered	Total buffered memory in GB for selected hosts.
	Memory - Shared	Total shared memory in GB for selected hosts.
DISK UTILITZATION	Disk Used	Disk space used in GB for selected hosts.
	Disk Free	Disk space available in GB for selected hosts.
DISK IO	Read Bytes	IOPS as read MBps for selected hosts.
	Write Bytes	IOPS as write MBps for selected hosts.
DISK IOPS	Read Count	IOPS as read count for selected hosts.
	Write Count	IOPS as write count for selected hosts.
NETWORK IO	Network Bytes Received	Network utilization as byte/sec received for selected hosts.
	Network Bytes Sent	Network utilization as byte/sec sent for selected hosts.
NETWORK PACKETS	Network Packets Received	Network utilization as packets received for selected hosts.
	Network Packets Sent	Network utilization as packets sent for selected hosts.
SWAP	Swap Space - Total	Total swap space available for selected hosts.
	Swap Space - Free	Total free swap space for selected hosts.
NUM PROCESSES	Num Processes - Total	Count of processes and total running processes for selected hosts.
	Num Processes - Runnable	Count of processes and total running processes for selected hosts.

9.1.3.12. NiFi Dashboard

The following Grafana dashboard is available for NiFi:

- [NiFi-Home \[167\]](#)

9.1.3.12.1. NiFi-Home

You can use the following metrics to assess the general health of your NiFi cluster.

For all metrics available in the NiFi-Home dashboard, the single value you see is the average of the information submitted by each node in your NiFi cluster.

Row	Metrics	Description
JVM Info	JVM Heap Usage	Displays the amount of memory being used by the JVM process. For NiFi, the default configuration is 512 MB.
	JVM File Descriptor Usage	Shows the number of connections to the operating system. You can monitor this metric to ensure that your JVM file descriptors, or connections, are opening and closing as tasks complete.
	JVM Uptime	Displays how long a Java process has been running. You can use this metric to monitor Java process longevity, and any unexpected restarts.
Thread Info	Active Threads	NiFi has two user configurable thread pools: <ul style="list-style-type: none"> • Maximum timer driven thread count (default 10) • Maximum event driven thread count (default 5) This metrics displays the number of active threads from these two pools.
	Thread Count	Displays the total number of threads for the JVM process that is running NiFi. This value is larger than the two pools above, because NiFi uses more than just the timer and event driven threads.
	Daemon Thread Count	Displays the number of daemon threads that are running. A daemon thread is a thread that does not prevent the JVM from exiting when the program finishes, even if the thread is still running.
FlowFile Info	FlowFiles Received	Displays the number of FlowFiles received into NiFi from an external system in the last 5 minutes.
	FlowFiles Sent	Displays the number of FlowFiles sent from NiFi to an external system in the last 5 minutes.
	FlowFiles Queued	Displays the number of FlowFiles queued in a NiFi processor connection.
Byte Info	Bytes Received	Displays the number of bytes of FlowFile data received into NiFi from an external system, in the last 5 minutes.
	Bytes Sent	Displays the number of bytes of FlowFile data sent from NiFi to an external system, in the last 5 minutes.
	Bytes Queued	Displays the number of bytes of FlowFile data queued in a NiFi processor connection.

9.1.4. AMS Performance Tuning

To set up Ambari Metrics System, in your environment, review and customize the following Metrics Collector configuration options.

- [Customizing the Metrics Collector Mode \[168\]](#)
- [Customizing TTL Settings \[169\]](#)
- [Customizing Memory Settings \[170\]](#)

- [Customizing Cluster-Environment-Specific Settings \[170\]](#)
- [Moving the Metrics Collector \[171\]](#)
- [\(Optional\) Enabling Individual Region, Table, and User Metrics for HBase \[172\]](#)

9.1.4.1. Customizing the Metrics Collector Mode

Metrics Collector is built using Hadoop technologies such as Apache HBase, Apache Phoenix, and Apache Traffic Server (ATS). The Collector can store metrics data on the local file system, referred to as *embedded mode*, or use an external HDFS, referred to as *distributed mode*. By default, the Collector runs in embedded mode. In embedded mode, the Collector captures and writes metrics to the local file system on the host where the Collector is running.



Important

When running in embedded mode, you should confirm that `hbase.rootdir` and `hbase.tmp.dir` have adequately sized and lightly used partitions. Directory configurations in **Ambari Metrics > Configs > Advanced > ams-hbase-site** are using a sufficiently sized and not heavily utilized partition, such as:

```
file:///grid/0/var/lib/ambari-metrics-collector/hbase.
```

You should also confirm that the TTL settings are appropriate.

When the Collector is configured for distributed mode, it writes metrics to HDFS, and the components run in distributed processes, which helps to manage CPU and memory consumption.

To switch the Metrics Collector from embedded mode to distributed mode,

Steps

1. In **Ambari Web**, browse to **Services > Ambari Metrics > Configs**.
2. Change the values of listed properties to the values shown in the following table:

Configuration Section	Property	Description	Value
General	Metrics Service operation mode (timeline.metrics.service.operation.mode)	Designates whether to run in distributed or embedded mode.	distributed
Advanced ams-hbase-site	hbase.cluster.distributed	Indicates AMS will run in distributed mode.	true
Advanced ams-hbase-site	hbase.rootdir 1	The HDFS directory location where metrics will be stored.	hdfs://\$NAMENODE_FQDN:8020/apps/ams/metrics

3. Using **Ambari Web > Hosts > Components** restart the Metrics Collector.

If your cluster is configured for a highly available NameNode, set the `hbase.rootdir` value to use the HDFS name service instead of the NameNode host name:

```
hdfs://hdfsnameservice/apps/ams/metrics
```

Optionally, you can migrate existing data from the local store to HDFS prior to switching to distributed mode:

Steps

1. Create an HDFS directory for the `ams` user:

```
su - hdfs -c 'hdfs dfs -mkdir -p /apps/ams/metrics'
```

2. Stop Metrics Collector.

3. Copy the metric data from the AMS local directory to an HDFS directory. This is the value of `hbase.rootdir` in `Advanced ams-hbase-site` used when running in embedded mode. For example:

```
su - hdfs -c 'hdfs dfs -copyFromLocal  
/var/lib/ambari-metrics-collector/hbase/* /apps/ams/metrics'
```

```
su - hdfs -c 'hdfs dfs -chown -R ams:hadoop  
/apps/ams/metrics'
```

4. Switch to distributed mode.
5. Restart the Metrics Collector.

If you are working with Apache HBase cluster metrics and want to display the more granular metrics of HBase cluster performance on the individual region, table, or user level, see .

More Information

[Customizing Cluster-Environment-Specific Settings \[170\]](#)

[Customizing TTL Settings \[169\]](#)

[Enabling Individual Region, Table, and User Metrics for HBase](#)

9.1.4.2. Customizing TTL Settings

AMS enables you to configure Time To Live (TTL) for aggregated metrics by navigating to `Ambari Metrics > Configs > Advanced ams-site`. Each property name is self explanatory and controls the amount of time to keep metrics (in seconds) before they are purged. The values for these TTL's are set in seconds.

For example, assume that you are running a single-node sandbox and want to ensure that no values are stored for more than seven days, to reduce local disk space consumption. In this case, you can set to 604800 (seven days, in seconds) any property ending in `.ttl` that has a value greater than 604800.

You likely want to do this for properties such as `timeline.metrics.cluster.aggregator.daily.ttl`, which controls the daily aggregation TTL and is set by default to two years. Two other properties that consume a lot of disk space are

- `timeline.metrics.cluster.aggregator.minute.ttl`, which controls minute -level aggregated metrics TTL, and
- `timeline.metrics.host.aggregator.ttl`, which controls host-based precision metrics TTL.

If you are working in an environment prior to Apache Ambari 2.1.2, you should make these settings during installation; otherwise, you must use the HBase shell by running the following command from the Collector host:

```
/usr/lib/ams-hbase/bin/hbase --config /etc/ams-hbase/conf shell
```

After you are connected, you must update each of the following tables with the TTL value `hbase(main):000:0> alter 'METRIC_RECORD_DAILY', { NAME => '0', TTL => 604800}`:

Map This TTL Property...	To This HBase Table...
<code>timeline.metrics.cluster.aggregator.daily.ttl</code>	<code>METRIC_AGGREGATE_DAILY</code>
<code>timeline.metrics.cluster.aggregator.hourly.ttl</code>	<code>METRIC_AGGREGATE_HOURLY</code>
<code>timeline.metrics.cluster.aggregator.minute.ttl</code>	<code>METRIC_AGGREGATE</code>
<code>timeline.metrics.host.aggregator.daily.ttl</code>	<code>METRIC_RECORD_DAILY</code>
<code>timeline.metrics.host.aggregator.hourly.ttl</code>	<code>METRIC_RECORD_HOURLY</code>
<code>timeline.metrics.host.aggregator.minute.ttl</code>	<code>METRIC_RECORD_MINUTE</code>
<code>timeline.metrics.host.aggregator.ttl</code>	<code>METRIC_RECORD</code>

9.1.4.3. Customizing Memory Settings

Because AMS uses multiple components (such as Apache HBase and Apache Phoenix) for metrics storage and query, multiple tunable properties are available to you for tuning memory use:

Configuration	Property	Description
Advanced <code>ams-env</code>	<code>metrics_collector_heapsize</code>	Heap size configuration for the Collector.
Advanced <code>ams-hbase-env</code>	<code>hbase_regionserver_heapsize</code>	Heap size configuration for the single AMS HBase Region Server.
Advanced <code>ams-hbase-env</code>	<code>hbase_master_heapsize</code>	Heap size configuration for the single AMS HBase Master.
Advanced <code>ams-hbase-env</code>	<code>regionserver_xmn_size</code>	Maximum value for the young generation heap size for the single AMS HBase RegionServer.
Advanced <code>ams-hbase-env</code>	<code>hbase_master_xmn_size</code>	Maximum value for the young generation heap size for the single AMS HBase Master.

9.1.4.4. Customizing Cluster-Environment-Specific Settings

The Metrics Collector mode, TTL settings, memory settings, and disk space requirements for AMS depend on the number of nodes in the cluster. The following table lists specific recommendations and tuning guidelines for each.

Cluster Environment	Host Count	Disk Space	Collector Mode	TTL	Memory Settings
Single-Node Sandbox	1	2GB	embedded	Reduce TTLs to 7 Days	<code>metrics_collector_heap_size=1024</code> <code>hbase_regionserver_heapsize=512</code> <code>hbase_master_heapsize=512</code> <code>hbase_master_xmn_size=128</code>
PoC	1-5	5GB	embedded	Reduce TTLs to 30 Days	<code>metrics_collector_heap_size=1024</code> <code>hbase_regionserver_heapsize=512</code> <code>hbase_master_heapsize=512</code>

Cluster Environment	Host Count	Disk Space	Collector Mode	TTL	Memory Settings
					hbase_master_xmn_size=128
Pre-Production	5-20	20GB	embedded	Reduce TTLs to 3 Months	metrics_collector_heap_size=1024 hbase_regionserver_heapsize=1024 hbase_master_heapsize=512 hbase_master_xmn_size=128
Production	20-50	50GB	embedded	n.a.	metrics_collector_heap_size=1024 hbase_regionserver_heapsize=1024 hbase_master_heapsize=512 hbase_master_xmn_size=128
Production	50-200	100GB	embedded	n.a.	metrics_collector_heap_size=2048 hbase_regionserver_heapsize=2048 hbase_master_heapsize=2048 hbase_master_xmn_size=256
Production	200-400	200GB	embedded	n.a.	metrics_collector_heap_size=2048 hbase_regionserver_heapsize=2048 hbase_master_heapsize=2048 hbase_master_xmn_size=512
Production	400-800	200GB	distributed	n.a.	metrics_collector_heap_size=8192 hbase_regionserver_heapsize=122288 hbase_master_heapsize=1024 hbase_master_xmn_size=1024 regionserver_xmn_size=1024
Production	800+	500GB	distributed	n.a.	metrics_collector_heap_size=12288 hbase_regionserver_heapsize=16384 hbase_master_heapsize=16384 hbase_master_xmn_size=2048 regionserver_xmn_size=1024

9.1.4.5. Moving the Metrics Collector

Use this procedure to move the Ambari Metrics Collector to a new host:

1. In **Ambari Web** , stop the **Ambari Metrics** service.
2. Execute the following API call to delete the current Metric Collector component:

```
curl -u admin:admin -H "X-Requested-By:ambari" -i -X
DELETE http://ambari.server:8080/api/v1/clusters/cluster.name/
hosts/metrics.collector.hostname/host_components/METRICS_COLLECTOR
```

3. Execute the following API call to add Metrics Collector to a new host:


```
curl -u admin:admin -H "X-Requested-By:ambari" -i -X
POST http://ambari.server:8080/api/v1/clusters/cluster.name/
hosts/metrics.collector.hostname/host_components/METRICS_COLLECTOR
```

4. In Ambari Web, go the page of the host on which you installed the new Metrics Collector and click **Install the Metrics Collector**.
5. In **Ambari Web**, start the **Ambari Metrics** service.



Note

Restarting all services is not required after moving the Ambari Metrics Collector, using Ambari 2.5 and later.

9.1.4.6. (Optional) Enabling Individual Region, Table, and User Metrics for HBase

Other than HBase RegionServer metrics, Ambari disables per region, per table, and per user metrics by default, because these metrics can be numerous and therefore cause performance issues.

If you want Ambari to collect these metrics, you can re-enable them; however, you should first test this option and confirm that your AMS performance is acceptable.

1. On the Ambari Server, browse to the following location:

```
/var/lib/ambari-server/resources/common-services/
HBASE/0.96.0.2.0/package/templates
```

2. Edit the following template files:

```
hadoop-metrics2-hbase.properties-GANGLIA-MASTER.j2
hadoop-metrics2-hbase.properties-GANGLIA-RS.j2
```

3. Either comment out or remove the following lines:

```
*.source.filter.class=org.apache.hadoop.metrics2.filter.RegexFilter
hbase.*.source.filter.exclude=.*(Regions|Users|Tables).*
```

4. Save the template files and restart Ambari Server for the changes to take effect.



Important

If you upgrade Ambari to a newer version, you must re-apply this change to the template file.

9.1.5. AMS High Availability

Ambari installs the Ambari Metrics System (AMS), into the cluster with a single **Metrics Collector** component by default. The **Collector** is a daemon that runs on a specific host in the cluster and receives data from the registered publishers, the **Monitors** and **Sinks**.

Depending on your needs, you might require AMS to have two Collectors to cover a High Availability scenario. This section describes the steps to enable AMS High Availability.

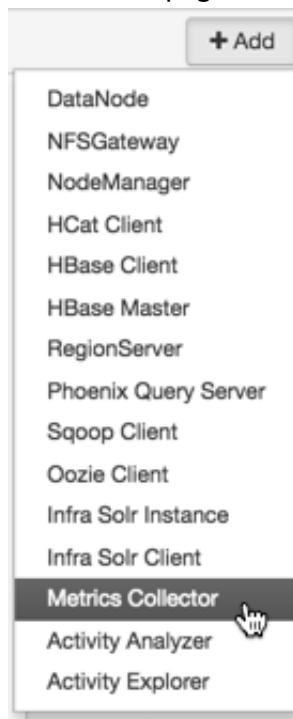
Prerequisite

You **must** deploy AMS in distributed (not embedded) mode.

To provide AMS High Availability:

Steps

1. In **Ambari Web**, browse to the host where you would like to install another collector.
2. On the **Host** page, choose **+Add**.

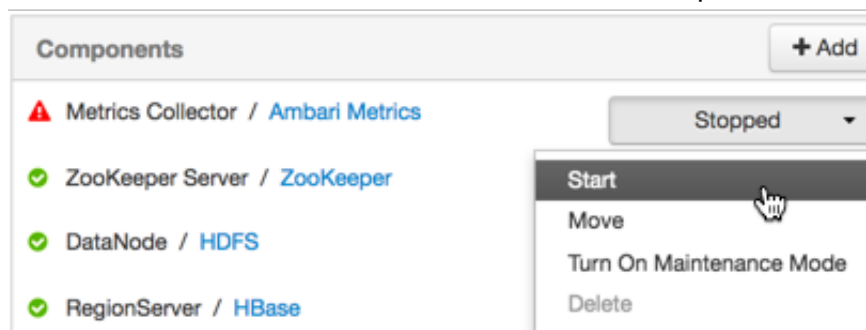


3. Select **Metrics Collector** from the list.

Ambari installs the new Metrics Collector and configures Ambari Metrics for HA.

The new Collector will be installed in a "stopped" state.

4. In **Ambari Web**, will have to start the new Collector component from Ambari Web.





Note

If you attempt to add a second Collector to the cluster without first switching AMS to distributed mode, the collector will install but will not be able to be started.

```
Traceback (most recent call last):
File
  "/var/lib/ambari-agent/cache/common-services/AMBARI_METRICS/0.1.0/
package/scripts/metrics_collector.py", line 150, in <module>
  AmsCollector().execute()
File
  "/usr/lib/python2.6/site-packages/resource_management/libraries/
script/script.py", line 313, in execute
  method(env)
File
  "/var/lib/ambari-agent/cache/common-services/AMBARI_METRICS/0.1.
0/package/scripts/metrics_collector.py", line 48, in start self.
configure(env, action = 'start') # for security
File
  "/usr/lib/python2.6/site-packages/resource_management/
libraries/script/script.py", line 116, in locking_configure
  original_configure(obj, *args, **kw)
File
  "/var/lib/ambari-agent/cache/common-services/AMBARI_METRICS/0.1.0/
package/scripts/metrics_collector.py", line 42, in configure raise
Fail("AMS in embedded mode cannot have more than 1 instance.
Delete all but 1 instances or switch to Distributed mode ")
  resource_management.core.exceptions.
Fail: AMS in embedded mode cannot have more than 1 instance.
Delete all but 1 instances or switch to Distributed mode
```

Workaround: Delete the newly added Collector, enable distributed mode, then re-add the Collector.

More Information

[AMS Architecture \[123\]](#)

[Customizing the Metrics Collector Mode \[168\]](#)

9.1.6. AMS Security

The following sections describe tasks to be performed when setting up security for the Ambari Metrics System.

- [Changing the Grafana Admin Password \[174\]](#)
- [Set Up HTTPS for AMS \[175\]](#)
- [Set Up HTTPS for Grafana \[178\]](#)

9.1.6.1. Changing the Grafana Admin Password

If you need to change the Grafana Admin password after you initially install Ambari, you have to change the password directly in Grafana, and then make the same change in the Ambari Metrics configuration:

Steps

1. In **Ambari Web**, browse to **Services > Ambari Metrics** select **Quick Links**, and then choose **Grafana**.

The Grafana UI opens in read-only mode.

2. Click **Sign In**, in the left column.
3. Log in as admin, using the unchanged password.
4. Click the admin label in the left column to view the admin profile, and then click **Change password**.
5. Enter the unchanged password, enter and confirm the new password, and click **Change Password**.
6. Return to **Ambari Web > Services > Ambari Metrics** and browse to the **Configs** tab.
7. In the General section, update and confirm the Grafana Admin Password with the new password.
8. Save the configuration and restart the services, as prompted.

9.1.6.2. Set Up HTTPS for AMS

If you want to limit access to AMS to HTTPS connections, you must provide a certificate. While it is possible to use a self-signed certificate for initial trials, it is not suitable for production environments. After you get your certificate, you must run a special setup command.

Steps

1. Create your own CA certificate.

```
openssl req -new -x509 -keyout ca.key -out ca.crt -days 365
```

2. Import CA certificate into the truststore.

```
# keytool -keystore /<path>/truststore.jks -alias CARoot -import -file ca.crt -storepass bigdata
```

3. Check truststore.

```
# keytool -keystore /<path>/truststore.jks -list
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

caroot, Feb 22, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
AD:EE:A5:BC:A8:FA:61:2F:4D:B3:53:3D:29:23:58:AB:2E:B1:82:AF
```

You should see `trustedCertEntry` for CA.

4. Generate certificate for AMS Collector and store private key in keystore.

```
# keytool -genkey -alias c6401.ambari.apache.org -keyalg RSA -keysize 1024
-dname "CN=c6401.ambari.apache.org,OU=IT,O=Apache,L=US,ST=US,C=US" -keypass
bigdata -keystore /<path>/keystore.jks -storepass bigdata
```



Note

If you use an alias different than the default hostname (`c6401.ambari.apache.org`), then, in step 12, set the `ssl.client.truststore.alias` config to use that alias.

5. Create certificate request for AMS collector certificate.

```
keytool -keystore /<path>/keystore.jks -alias c6401.ambari.apache.org -
certreq -file c6401.ambari.apache.org.csr -storepass bigdata
```

6. Sign the certificate request with the CA certificate.

```
openssl x509 -req -CA ca.crt -CAkey ca.key -in c6401.ambari.apache.org.csr
-out c6401.ambari.apache.org_signed.crt -days 365 -CAcreateserial -passin
pass:bigdata
```

7. Import CA certificate into the keystore.

```
keytool -keystore /<path>/keystore.jks -alias CARoot -import -file ca.crt -
storepass bigdata
```

8. Import signed certificate into the keystore.

```
keytool -keystore /<path>/keystore.jks -alias c6401.ambari.apache.org -
import -file c6401.ambari.apache.org_signed.crt -storepass bigdata
```

9. Check keystore.

```
caroot2, Feb 22, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
 7C:B7:0C:27:8E:0D:31:E7:BE:F8:BE:A1:A4:1E:81:22:FC:E5:37:D7
[root@c6401 tmp]# keytool -keystore /tmp/keystore.jks -list
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

caroot, Feb 22, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
 AD:EE:A5:BC:A8:FA:61:2F:4D:B3:53:3D:29:23:58:AB:2E:B1:82:AF
c6401.ambari.apache.org, Feb 22, 2016, PrivateKeyEntry,
Certificate fingerprint (SHA1):
 A2:F9:BE:56:7A:7A:8B:4C:5E:A6:63:60:B7:70:50:43:34:14:EE:AF
```

You should see `PrivateKeyEntry` for the ams collector hostname entry and `trustedCertEntry` for CA.

10. Copy /<path>/truststore.jks to all nodes to /<path>/truststore.jks and set appropriate access permissions.

11. Copy /<path>/keystore.jks to AMS collector node ONLY to /<path>/keystore.jks and set appropriate access permissions. Recommended: set owner to ams user and access permissions to 400.

12. In Ambari Web, update the following AMS configs, in Advanced:

- ams-site/timeline.metrics.service.http.policy=HTTPS_ONLY
- ams-ssl-server/ssl.server.keystore.keypassword=bigdata
- ams-ssl-server/ssl.server.keystore.location=/<path>/keystore.jks
- ams-ssl-server/ssl.server.keystore.password=bigdata
- ams-ssl-server/ssl.server.keystore.type=jks
- ams-ssl-server/ssl.server.truststore.location=/<path>/truststore.jks
- ams-ssl-server/ssl.server.truststore.password=bigdata
- ams-ssl-server/ssl.server.truststore.reload.interval=10000
- ams-ssl-server/ssl.server.truststore.type=jks
- ams-ssl-client/ssl.client.truststore.location=/<path>/truststore.jks
- ams-ssl-client/ssl.client.truststore.password=bigdata
- ams-ssl-client/ssl.client.truststore.type=jks

13. In Ambari Web, Add the following AMS config property, using Custom ams-ssl-client # Add Property:

```
[metrics_collector_hostname_fqdn].ssl.client.truststore.alias=<Alias used to create certificate for AMS on the host with the specified FQDN. (Default is hostname fqdn)>
```

14. Restart services with stale configs.

15. Configure Ambari server to use truststore.

```
# ambari-server setup-security
Using python /usr/bin/python
Security setup options...
=====
Choose one of the following options:
 [1] Enable HTTPS for Ambari server.
 [2] Encrypt passwords stored in ambari.properties file.
 [3] Setup Ambari kerberos JAAS configuration.
 [4] Setup truststore.
 [5] Import certificate to truststore.
=====
Enter choice, (1-5): 4
Do you want to configure a truststore [y/n] (y)?
TrustStore type [jks/jceks/pkcs12] (jks): jks
```

```
Path to TrustStore file :/<path>/keystore.jks
Password for TrustStore:
Re-enter password:
Ambari Server 'setup-security' completed successfully.
```

16. Configure ambari server to use https instead of http in requests to AMS Collector by adding "server.timeline.metrics.https.enabled=true" to ambari.properties file.

```
# echo "server.timeline.metrics.https.enabled=true" >> /etc/ambari-server/
conf/ambari.properties
```

17. Restart ambari server.

9.1.6.3. Set Up HTTPS for Grafana

If you want to limit access to the Grafana to HTTPS connections, you must provide a certificate. While it is possible to use a self-signed certificate for initial trials, it is not suitable for production environments. After you get your certificate, you must run a special setup command.

Steps

1. Log on to the host with Grafana.
2. Browse to the Grafana configuration directory:

```
cd /etc/ambari-metrics-grafana/conf/
```

3. Locate your certificate.

If you want to create a temporary self-signed certificate, you can use this as an example:

```
openssl genrsa -out ams-grafana.key 2048
openssl req -new -key ams-grafana.key -out ams-grafana.csr
openssl x509 -req -days 365 -in ams-grafana.csr -signkey ams-grafana.key -
out ams-grafana.crt
```

4. Set the certificate and key file ownership and permissions so that they are accessible to Grafana:

```
chown ams:hadoop ams-grafana.crt
chown ams:hadoop ams-grafana.key
chmod 400 ams-grafana.crt
chmod 400 ams-grafana.key
```

For a non-root Ambari user, use

```
chmod 444 ams-grafana.crt
```

to enable the agent user to read the file.

5. In **Ambari Web**, browse to **> Services > Ambari Metrics > Configs**.
6. Update the following properties in the **Advanced ams-grafana-ini** section:

```
protocol    https
```

```
cert_file    /etc/ambari-metrics-grafana/conf/ams-grafana.crt
cert-Key     /etc/ambari-metrics-grafana/conf/ams-grafana.key
```

7. Save the configuration and restart the services as prompted.

9.2. Ambari Log Search (Technical Preview)

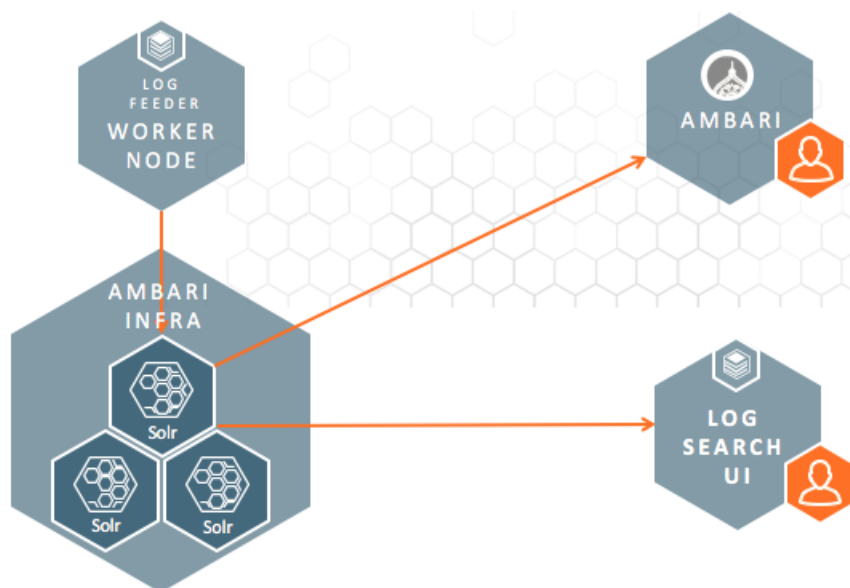
The following sections describe the Technical Preview release of Ambari Log Search, which you should use only in non-production clusters with fewer than 150 nodes. .

- [Log Search Architecture \[179\]](#)
- [Installing Log Search \[180\]](#)
- [Using Log Search \[180\]](#)

9.2.1. Log Search Architecture

Ambari Log Search enables you to search for logs generated by Ambari-managed HDP components. Ambari Log Search relies on the Ambari Infra service to provide Apache Solr indexing services. Two components compose the Log Search solution:

- [Log Feeder \[179\]](#)
- [Log Search Server \[180\]](#)



9.2.1.1. Log Feeder

The Log Feeder component parses component logs. A Log Feeder is deployed to every node in the cluster and interacts with all component logs on that host. When started, the

Log Feeder begins to parse all known component logs and sends them to the Apache Solr instances (managed by the Ambari Infra service) to be indexed.

By default, only FATAL, ERROR, and WARN logs are captured by the Log Feeder. You can temporarily or permanently add other log levels using the Log Search UI filter settings



(for temporary log level capture) or through the Log Search configuration control in Ambari.

9.2.1.2. Log Search Server

The Log Search Server hosts the Log Search UI web application, providing the API that is used by Ambari and the Log Search UI to access the indexed component logs. After logging in as a local or LDAP user, you can use the Log Search UI to visualize, explore, and search indexed component logs.

9.2.2. Installing Log Search

Log Search is a built-in service in Ambari 2.4 and later. You can add it during a new installation by using the [+Add Service](#) menu. The Log Feeders are automatically installed on all nodes in the cluster; you manually place the Log Search Server, optionally on the same server as the Ambari Server.

9.2.3. Using Log Search

Using **Ambari Log Search** includes the following activities:

- [Accessing Log Search \[180\]](#)
- [Using Log Search to Troubleshoot \[182\]](#)
- [Viewing Service Logs \[182\]](#)
- [Viewing Access Logs \[183\]](#)

9.2.3.1. Accessing Log Search

After Log Search is installed, you can use any of three ways to search the indexed logs:

- [Ambari Background Ops Log Search Link \[181\]](#)
- [Host Detail Logs Tab \[181\]](#)
- [Log Search UI \[181\]](#)



Note

Single Sign On (SSO) between Ambari and Log Search is not currently available.

9.2.3.1.1. Ambari Background Ops Log Search Link

When you perform lifecycle operations such as starting or stopping services, it is critical that you have access to logs that can help you recover from potential failures. These logs are now available in **Background Ops**. **Background Ops** also links to the Host Detail Logs tab, which lists all the log files that have been indexed and can be viewed for a specific host:

```

rev05.hortonworks.local
Tasks DataNode Start Copy Open Host Logs
Ambari stdout/stderr hadoop-hdfs-datanode-revo5.hortonworks.local.log
stderr: /var/lib/ambari-agent/data/errors-105.txt
None
stdout: /var/lib/ambari-agent/data/output-105.txt
2016-06-02 10:40:53,887 - The hadoop conf dir /usr/hdp/current/hadoop-client/conf exists, will call conf-se
lect on it for version 2.4.0.0-169
2016-06-02 10:40:53,887 - Checking if need to create versioned conf dir /etc/hadoop/2.4.0.0-169/0
2016-06-02 10:40:53,887 - call[[ambari-python-wrap, /usr/bin/conf-select, 'create-conf-dir', '--package
', 'hadoop', '--stack-version', '2.4.0.0-169', '--conf-version', '0']] {'logoutput': False, 'sudo': True, '
quiet': False, 'stderr': -1}
2016-06-02 10:40:53,910 - call returned (1, '/etc/hadoop/2.4.0.0-169/0 exist already', '')
2016-06-02 10:40:53,910 - checked_call[[ambari-python-wrap, /usr/bin/conf-select, 'set-conf-dir', '--pa
ckage', 'hadoop', '--stack-version', '2.4.0.0-169', '--conf-version', '0']] {'logoutput': False, 'sudo': Tr
ue, 'quiet': False}
2016-06-02 10:40:53,933 - checked_call returned (0, '/usr/hdp/2.4.0.0-169/hadoop/conf -> /etc/hadoop/2.4.0.
0-169/0')
2016-06-02 10:40:53,934 - Ensuring that hadoop has the correct symlink structure
2016-06-02 10:40:53,934 - Using hadoop conf dir: /usr/hdp/current/hadoop-client/conf
2016-06-02 10:40:54,050 - The hadoop conf dir /usr/hdp/current/hadoop-client/conf exists, will call conf-se
lect on it for version 2.4.0.0-169
2016-06-02 10:40:54,051 - Checking if need to create versioned conf dir /etc/hadoop/2.4.0.0-169/0
2016-06-02 10:40:54,051 - call[[ambari-python-wrap, /usr/bin/conf-select, 'create-conf-dir', '--package
', 'hadoop', '--stack-version', '2.4.0.0-169', '--conf-version', '0']] {'logoutput': False, 'sudo': True, '
quiet': False, 'stderr': -1}
 Do not show this dialog again when starting a background operation 

```

More Information

Background Ops

9.2.3.1.2. Host Detail Logs Tab

A **Logs** tab is added to each host detail page, containing a list of indexed, viewable log files, organized by service, component, and type. You can open and search each of these files by using a link to the Log Search UI:

Service	Component	Extension	
All	All	All	
HDFS	DataNode	hadoop-hdfs-datanode-revo5.hortonworks.local.log	Open in Log Search
MapReduce2	History Server	mapred-mapred-historyserver-revo5.hortonworks.local.log	Open in Log Search
Log Search	Log Feeder	logsearch-logfeeder.json	Open in Log Search
Log Search	Log Search Server	logsearch.json	Open in Log Search
Ambari Metrics	Metrics Collector	ambari-metrics-collector.log	Open in Log Search
HDFS	NameNode	hadoop-hdfs-namenode-revo5.hortonworks.local.log	Open in Log Search
YARN	NodeManager	yarn-yarn-nodemanager-revo5.hortonworks.local.log	Open in Log Search
Oozie	Oozie Server	oozie.log	Open in Log Search
HDFS	ShameNode	hadoop-hdfs-secondarynamenode-revo5.hortonworks.local.log	Open in Log Search
ZooKeeper	ZooKeeper Server	zookeeper-zookeeper-server-revo5.hortonworks.local.out	Open in Log Search

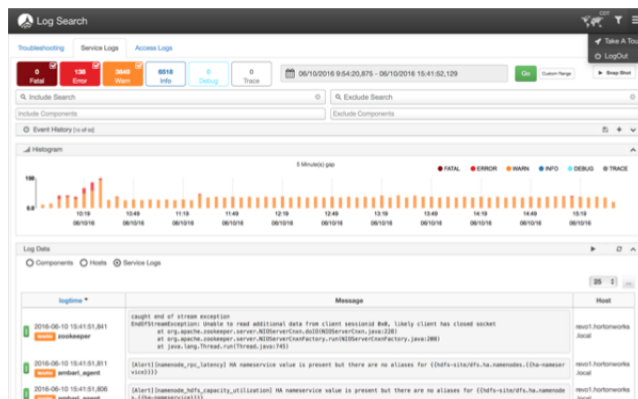
9.2.3.1.3. Log Search UI

The Log Search UI is a purpose-built web application used to search HDP component logs. The UI is focussed on helping operators quickly access and search logs from a single location. Logs can be filtered by log level, time, component, and can be searched by keyword. Helpful tools such as histograms to show number of logs by level for a time

period are available, as well as controls to help rewind and fast forward search sessions, contextual click to include/exclude terms in log viewing, and multi-tab displays for troubleshooting multi-component and host issues.

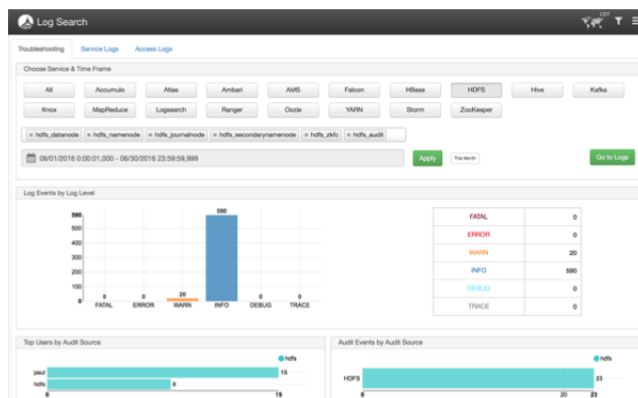
The Log Search UI is available from the Quick Links menu of the Log Search Service within Ambari Web.

To see a guided tour of Log Search UI features, choose **Take a Tour** from the Log Search UI main menu. Click **Next** to view each topic in the guided tour series.



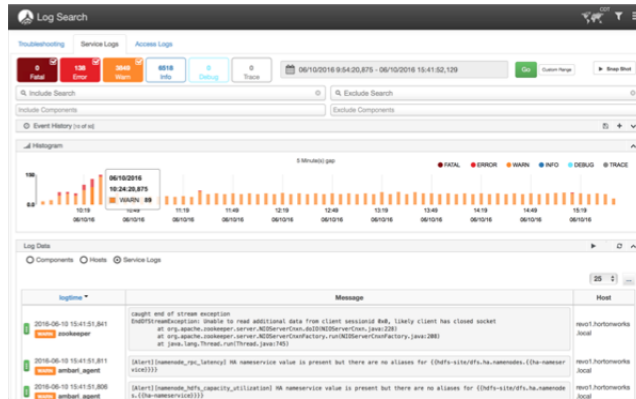
9.2.3.2. Using Log Search to Troubleshoot

To find logs related to a specific problem, use the **Troubleshooting** tab in the UI to select the service, components, and time frame related to the problem you are troubleshooting. For example, if you select HDFS, the UI automatically searches for HDFS-related components. You can select a time frame of yesterday or last week, or you can specify a custom value. Each of these specifications filters the results to match your interests. When you are ready to view the matching logs, you can click **Go to Logs**:



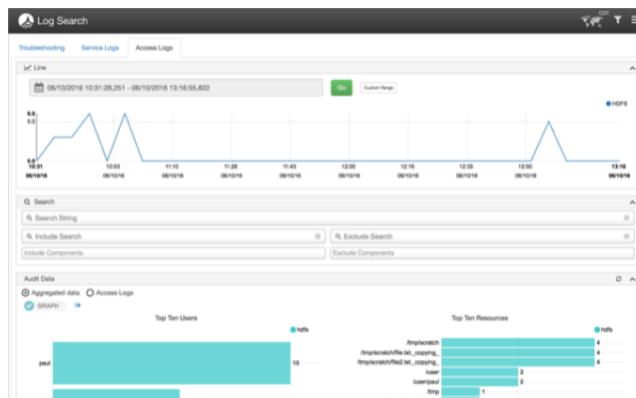
9.2.3.3. Viewing Service Logs

The **Service Logs** tab enables you to search across all component logs for specific keywords and to filter for specific log levels, components, and time ranges. The UI is organized so that you can quickly see how many logs were captured for each log level across the entire cluster, search for keywords, include and exclude components, and match logs to your search query:



9.2.3.4. Viewing Access Logs

When troubleshooting HDFS-related issues, you might find it helpful to search for and spot trends in HDFS access by users. The Access Logs tab enables you to view HDFS Audit log entries for a specific time frame, to see aggregated usage data showing the top ten HDFS users by file system resources accessed, as well as the top ten file system resources accessed across all users. This can help you find anomalies or hot and cold data sets.



9.3. Ambari Infra

Many services in HDP depend on core services to index data. For example, Apache Atlas uses indexing services for tagging lineage-free text search, and Apache Ranger uses indexing for audit data. The role of Ambari Infra is to provide these common shared services for stack components.

Currently, the Ambari Infra Service has only one component: the Infra Solr Instance. The Infra Solr Instance is a fully managed Apache Solr installation. By default, a single-node SolrCloud installation is deployed when the Ambari Infra Service is chosen for installation; however, you should install multiple Infra Solr Instances so that you have distributed indexing and search for Atlas, Ranger, and LogSearch (Technical Preview).

To install multiple Infra Solr Instances, you simply add them to existing cluster hosts through Ambari's **+Add Service** capability. The number of Infra Solr Instances you deploy depends on the number of nodes in the cluster and the services deployed.

Because one Ambari Infra Solr Instance is used by multiple HDP components, you should be careful when restarting the service, to avoid disrupting those dependent services. In HDP 2.5 and later, Atlas, Ranger, and Log Search (Technical Preview) dependent on Ambari Infra Service.



Note

Infra Solr Instance is intended for use only by HDP components; use by third-party components or applications is not supported.

9.3.1. Archiving & Purging Data

Large clusters produce many log entries, and Ambari Infra provides a convenient utility for archiving and purging logs that are no longer required.

This utility is called the Solr Data Manager. The Solr Data Manager is a python program available in `/usr/bin/infra-solr-data-manager`. This program allows users to quickly archive, delete, or save data from a Solr collection, with the following usage options.

9.3.1.1. Command Line Options

Operation Modes

-m MODE, --mode=MODE archive | delete | save

The mode to use depends on the intent. Archive will store data into the desired storage medium and then remove the data after it has been stored, Delete is self explanatory, and Save is just like Archive except that data is not deleted after it has been stored.

–

Connecting to Solr

-s SOLR_URL, --solr-url=SOLR_URL>

The URL to use to connect to the specific Solr Cloud instance.

For example:

`'http://c6401.ambari.apache.org:8886/solr'`.

-c COLLECTION, --collection=COLLECTION

The name of the Solr collection. For example: `'hadoop_logs'`

-k SOLR_KEYTAB, --solr-keytab=SOLR_KEYTAB

The keytab file to use when operating against a kerberized Solr instance.

-n SOLR_PRINCIPAL, --solr-principal=SOLR_PRINCIPAL

The principal name to use when operating against a kerberized Solr instance.

–

Record Schema

-i ID_FIELD, --id-field=ID_FIELD

The name of the field in the solr schema to use as the unique identifier for each record.

-f FILTER_FIELD, --filter-field=FILTER_FIELD

The name of the field in the solr schema to filter off of. For example: 'logtime'

-o DATE_FORMAT, --date-format=DATE_FORMAT

The custom date format to use with the -d DAYS field to match log entries that are older than a certain number of days.

-e END

Based on the filter field and date format, this argument configures the date that should be used as the end of the date range. If you use '2018-08-29T12:00:00.000Z', then any records with a filter field that is after that date will be saved, deleted, or archived depending on the mode.

-d DAYS, --days=DAYS

Based on the filter field and date format, this argument configures the number days before today should be used as the end of the range. If you use '30', then any records with a filter field that is older than 30 days will be saved, deleted, or archived depending on the mode.

-q ADDITIONAL_FILTER, --additional-filter=ADDITIONAL_FILTER

Any additional filter criteria to use to match records in the collection.

–

Extracting Records

-r READ_BLOCK_SIZE, --read-block-size=READ_BLOCK_SIZE

The number of records to read at a time from Solr. For example: '10' to read 10 records at a time.

-w WRITE_BLOCK_SIZE, --write-block-size=WRITE_BLOCK_SIZE

The number of records to write per output file. For example: '100' to write 100 records per file.

-j NAME, --name=NAME name included in result files

Additional name to add to the final filename created in save or archive mode.

-json-file

Default output format is one valid json document per record delimited by a newline. This option will write out a single valid JSON document containing all of the records.

-z COMPRESSION, --compression=COMPRESSION none | tar.gz | tar.bz2 | zip | gz

Depending on how output files will be analyzed, you have the choice to choose the optimal compression and file format to use for output files. Gzip compression is used by default.

–

Writing Data to HDFS

-a HDFS_KEYTAB, --hdfs-keytab=HDFS_KEYTAB

The keytab file to use when writing data to a kerberized HDFS instance.

-l HDFS_PRINCIPAL, --hdfs-principal=HDFS_PRINCIPAL

The principal name to use when writing data to a kerberized HDFS instance.

-u HDFS_USER, --hdfs-user=HDFS_USER

The user to connect to HDFS as.

-p HDFS_PATH, --hdfs-path=HDFS_PATH

The path in HDFS to write data to in save or archive mode.

–

Writing Data to S3

-t KEY_FILE_PATH, --key-file-path=KEY_FILE_PATH

The path to the file on the local file system that contains the AWS Access and Secret Keys. The file should contain the keys in this format: <accessKey>,<secretKey>

-b BUCKET, --bucket=BUCKET

The name of the bucket that data should be uploaded to in save or archive mode.

-y KEY_PREFIX, --key-prefix=KEY_PREFIX

The key prefix allows you to create a logical grouping of the objects in an S3 bucket. The prefix value is similar to a directory name enabling you to store data in the same directory in a bucket. For example, if your Amazon S3 bucket name is logs, and you set prefix to hadoop/, and the file on your storage device is hadoop_logs_-_2017-10-28T01_25_40.693Z.json.gz, then the file would be identified by this URL: http://s3.amazonaws.com/logs/hadoop/hadoop_logs_-_2017-10-28T01_25_40.693Z.json.gz

-g, --ignore-unfinished-uploading

To deal with connectivity issues, uploading extracted data can be retried. If you do not wish to resume uploads, use the -g flag to disable this behaviour.

–

Writing Data Locally

-x LOCAL_PATH, --local-path=LOCAL_PATH

The path on the local file system that should be used to write data to in save or archive mode

–

Examples

Deleting Indexed Data

In delete mode (-m delete), the program deletes data from the Solr collection. This mode uses the filter field (-f FILTER_FIELD) option to control which data should be removed from the index.

The command below will delete log entries from the hadoop_logs collection, which have been created before August 29, 2017, we'll use the -f option to specify the field in the Solr collection to use as a filter field, and the -e option to denote the end of the range of values to remove.

```
infra-solr-data-manager -m delete -s
://c6401.ambari.apache.org:8886/solr -c hadoop_logs -f logtime -e
2017-08-29T12:00:00.000Z
```

Archiving Indexed Data

In archive mode, the program fetches data from the Solr collection and writes it out to HDFS or S3, then deletes the data.

The program will fetch records from Solr and creates a file once the write block size is reached, or if there are no more matching records found in Solr. The program keeps track of its progress by fetching the records ordered by the filter field, and the id field, and always saves their last values. Once the file is written, it's is compressed using the configured compression type.

After the compressed file is created the program creates a command file containing instructions with next steps. In case of any interruptions or error during the next run for the same collection the program will start executing the saved command file, so all the data would be consistent. If the error is due to invalid configuration, and failures persist, the -g option can be used to ignore the saved command file. The program supports writing data to HDFS, S3, or Local Disk.

The command below will archive data from the solr collection hadoop_logs accessible at <http://c6401.ambari.apache.org:8886/solr> based on the field logtime, and will extract everything older than 1 day, read 10 documents at once, write 100 documents into a file, and copy the zip files into the local directory /tmp.

```
infra-solr-data-manager -m archive -s
http://c6401.ambari.apache.org:8886/solr -c hadoop_logs -f logtime -d
1 -r 10 -w 100 -x /tmp -v
```


Saving Indexed Data

Saving is similar to Archiving data except that the data is not deleted from Solr after the files are created and uploaded. The Save mode is recommended for testing that the data is written as expected before running the program in Archive mode with the same parameters.

The below example will save the last 3 days of hdfs audit logs into HDFS path "/" with the user hdfs, fetching data from a kerberized Solr.

```
infra-solr-data-manager -m save -s
http://c6401.ambari.apache.org:8886/solr -c audit_logs -f logtime -d 3
-r 10 -w 100 -q type:"hdfs_audit" -j hdfs_audit -k
/etc/security/keytabs/ambari-infra-solr.service.keytab -n
infra-solr/c6401.ambari.apache.org@AMBARI.APACHE.ORG -u hdfs -p /
```

Analyzing Archived Data With Hive

Once data has been archived or saved to HDFS, Hive tables can be used to quickly access and analyzed stored data. Only line delimited JSON files can be analyzed with Hive. Line delimited JSON files are created by default unless the `--json-file` argument is passed. Data saved or archived using `--json-file` cannot be analyzed with Hive. In the following examples, the `hive-json-serde.jar` is used to process the stored JSON data. Prior to creating the included tables, the jar must be added in the Hive shell:

```
ADD JAR <path-to-jar>/hive-json-serde.jar
```

Here are some examples for table schemes for various log types. Using external tables is recommended, as it has the advantage of keeping the archives in HDFS. First ensure a directory is created to store archived or stored line delimited logs:

```
hadoop fs -mkdir <some directory path>
```

Hadoop Logs

```
CREATE EXTERNAL TABLE hadoop_logs
(
  logtime string,
  level string,
  thread_name string,
  logger_name string,
  file string,
  line_number int,
  method string,
  log_message string,
  cluster string,
  type string,
  path string,
  logfile_line_number int,
  host string,
  ip string,
  id string,
  event_md5 string,
  message_md5 string,
  seq_num int
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
```

```
LOCATION '<some directory path>';
```

Audit Logs

As audit logs have a slightly different field set, we suggest to archive them separately using `-additional-filter`, and we offer separate schemas for HDFS, Ambari, and Ranger audit logs.

HDFS Audit Logs

```
CREATE EXTERNAL TABLE audit_logs_hdfs
(
  evtTime string,
  level string,
  logger_name string,
  log_message string,
  resource string,
  result int,
  action string,
  cliType string,
  req_caller_id string,
  ugi string,
  reqUser string,
  proxyUsers array<string>,
  authType string,
  proxyAuthType string,
  dst string,
  perm string,
  cluster string,
  type string,
  path string,
  logfile_line_number int,
  host string,
  ip string,
  cliIP string,
  id string,
  event_md5 string,
  message_md5 string,
  seq_num int
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION '<some directory path>';
```

Ambari Audit Logs

```
CREATE EXTERNAL TABLE audit_logs_ambari
(
  evtTime string,
  log_message string,
  resource string,
  result int,
  action string,
  reason string,
  ws_base_url string,
  ws_command string,
  ws_component string,
  ws_details string,
  ws_display_name string,
  ws_os string,
  ws_repo_id string,
  ws_repo_version string,
```

```
ws_repositories string,  
ws_request_id string,  
ws_roles string,  
ws_stack string,  
ws_stack_version string,  
ws_version_note string,  
ws_version_number string,  
ws_status string,  
ws_result_status string,  
cliType string,  
reqUser string,  
task_id int,  
cluster string,  
type string,  
path string,  
logfile_line_number int,  
host string,  
cliIP string,  
id string,  
event_md5 string,  
message_md5 string,  
seq_num int  
)  
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  
LOCATION '<some directory path>';
```

Ranger Audit Logs

```
CREATE EXTERNAL TABLE audit_logs_ranger  
(  
  evtTime string,  
  access string,  
  enforcer string,  
  resource string,  
  result int,  
  action string,  
  reason string,  
  resType string,  
  reqUser string,  
  cluster string,  
  cliIP string,  
  id string,  
  seq_num int  
)  
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  
LOCATION '<some directory path>';
```

9.3.2. Performance Tuning for Ambari Infra

When using Ambari Infra to index and store Ranger audit logs, you should properly tune Solr to handle the number of audit records stored per day. The following sections describe recommendations for tuning your operating system and Solr, based on how you use Ambari Infra and Ranger in your environment.

9.3.2.1. Operating System Tuning

Solr clients use many network connections when indexing and searching, and to avoid many open network connections, the following sysctl parameters are recommended:

```
net.ipv4.tcp_max_tw_buckets = 1440000
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
```

These settings can be made permanent by placing them in `/etc/sysctl.d/net.conf`, or they can be set at runtime using the following `sysctl` command example:

```
sysctl -w net.ipv4.tcp_max_tw_buckets=1440000
sysctl -w net.ipv4.tcp_tw_recycle=1
sysctl -w net.ipv4.tcp_tw_reuse=1
```

Additionally, the number of user processes for solr should be increased to avoid exceptions related to creating new native threads. This can be done by creating a new file named `/etc/security/limits.d/infra-solr.conf` with the following contents:

```
infra-solr - nproc 6000
```

9.3.2.2. JVM - GC Settings

The heap sizing and garbage collection settings are very important for production Solr instances when indexing many Ranger audit logs. For production deployments, we suggest setting the "Infra Solr Minimum Heap Size," and "Infra Solr Maximum Heap Size" to 12 GB. These settings can be found and applied by following the steps below:

Steps

1. In Ambari Web, browse to **Services > Ambari Infra > Configs**.
2. In the **Settings** tab you will see two sliders controlling the Infra Solr Heap Size.
3. Set the Infra Solr Minimum Heap Size to 12GB or 12,288MB.
4. Set the Infra Solr Maximum Heap Size to 12GB or 12,288MB.
5. Click **Save** to save the configuration and then restart the affected services as prompted by Ambari.

Using the G1 Garbage Collector is also recommended for production deployments. To use the G1 Garbage Collector with the Ambari Infra Solr Instance, follow the steps below:

Steps

1. In Ambari Web, browse to **Services > Ambari Infra > Configs**.
2. In the **Advanced** tab expand the section for **Advanced infra-solr-env**
3. In the **infra-solr-env template** locate the multi-line `GC_TUNE` environmental variable definition, and replace it with the following content:

```
GC_TUNE="-XX:+UseG1GC
-XX:+PerfDisableSharedMem
-XX:+ParallelRefProcEnabled
-XX:G1HeapRegionSize=4m
-XX:MaxGCPauseMillis=250
-XX:InitiatingHeapOccupancyPercent=75
-XX:+UseLargePages
```

```
-XX:+AggressiveOpts"
```

The value used for the `-XX:G1HeapRegionSize` is based on the 12GB Solr Maximum Heap recommended. If you choose to use a different heap size for the Solr server, please consult the following table for recommendations:

Heap Size	G1HeapRegionSize
< 4GB	1MB
4-8GB	2MB
8-16GB	4MB
16-32GB	8MB
32-64GB	16MB
>64GB	32MB

9.3.2.3. Environment-Specific Tuning Parameters

Each of the recommendations below are dependent on the number of audit records that are indexed per day. To quickly determine how many audit records are indexed per day, use the command examples below:

Using a HTTP client such as curl, execute the following command:

```
curl -g "http://<ambari infra hostname>:8886/solr/ranger_audits/select?q=(evtTime:[NOW-7DAYS+TO+*])&wt=json&indent=true&rows=0"
```

You should receive a message similar to the following:

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": "evtTime:[NOW-7DAYS TO *]",
      "indent": "true",
      "rows": "0",
      "wt": "json"
    }
  },
  "response": { "numFound": 306, "start": 0, "docs": [] }
}
```

Take the `numFound` element of the response and divide it by 7 to get the average number of audit records being indexed per day. You can also replace the `'7DAYS'` in the curl request with a broader time range, if necessary, using the following key words:

- 1MONTHS
- 7DAYS

Just ensure you divide by the appropriate number if you change the event time query. The average number of records per day will be used to identify which recommendations below apply to your environment.

Less Than 50 Million Audit Records Per Day

Based on the Solr REST API call if your average number of documents per day is less than 50 million records per day, the following recommendations apply. In each recommendation, the time to live, or TTL, which

controls how long a document should be kept in the index until it is removed is taken into consideration. The default TTL is 90 days, but some customers choose to be more aggressive, and remove documents from the index after 30 days. Due to this, recommendations for both common TTL settings are specified.

These recommendations assume that you are using our recommendation of 12GB heap per Solr server instance. In each situation we have recommendations for co-locating Solr with other master services, and for using dedicated Solr servers. Testing has shown that Solr performance requires different server counts depending on whether Solr is co-located or on dedicated servers. Based on our testing with Ranger, Solr shard sizes should be around 25GB for best overall performance. However, Solr shard sizes can go up to 50GB without a significant performance impact.

This configuration is our best recommendation for just getting started with Ranger and Ambari Infra so the only recommendation is using the default TTL of 90 days.

Default Time To Live (TTL) 90 days:

- Estimated total index size: ~ 150 GB to 450 GB
- Total number of primary/leader shards: 6
- Total number of shards including 1 replica each: 12
- Total number of co-located Solr nodes: ~ 3 nodes, up to 2 shards per node

(does not include replicas)

- Total number of dedicated Solr nodes: ~ 1 node, up to 12 shards per node

(does not include replicas)

50 - 100 Million Audit Records
Per Day

50 to 100 million records ~ 5 - 10 GB data per day.

Default Time To Live (TTL) 90 days:

- Estimated total index size: ~ 450 - 900 GB for 90 days
- Total number of primary/leader shards: 18-36
- Total number of shards including 1 replica each: 36-72
- Total number of co-located Solr nodes: ~ 9-18 nodes, up to 2 shards per node

(does not include replicas)

- Total number of dedicated Solr nodes: ~3-6 nodes, up to 12 shards per node

(does not include replicas)

Custom Time To Live (TTL) 30 days:

- Estimated total index size: 150 - 300 GB for 30 days
- Total number of primary/leader shards: 6-12
- Total number of shards including 1 replica each: 12-24
- Total number of co-located Solr nodes: ~3-6 nodes, up to 2 shards per node

(does not include replicas)

- Total number of dedicated Solr nodes: ~1-2 nodes, up to 12 shards per node

(does not include replicas)

100 - 200 Million Audit Records
Per Day

100 to 200 million records ~ 10 - 20 GB data per day.

Default Time To Live (TTL) 90 days:

- Estimated total index size: ~ 900 - 1800 GB for 90 days
- Total number of primary/leader shards: 36-72
- Total number of shards including 1 replica each: 72-144
- Total number of co-located Solr nodes: ~18-36 nodes, up to 2 shards per node

(does not include replicas)

- Total number of dedicated Solr nodes: ~3-6 nodes, up to 12 shards per node

(does not include replicas)

Custom Time To Live (TTL) 30 days:

- Estimated total index size: 300 - 600 GB for 30 days
- Total number of primary/leader shards: 12-24

- Total number of shards including 1 replica each: 24-48

- Total number of co-located Solr nodes: ~6-12 nodes, up to 2 shards per node
(does not include replicas)
- Total number of dedicated Solr nodes: ~1-3 nodes, up to 12 shards per node
(does not include replicas)

If you choose to use at least 1 replica for high availability, then increase the number of nodes accordingly. If high availability is a requirement, then consider using no less than 3 Solr nodes in any configuration.

As illustrated in these examples, a lower TTL requires less resources. If your compliance objectives call for longer data retention, you can use the SolrDataManager to archive data into long term storage (HDFS, or S3) and provides Hive tables allowing you to easily query that data. With this strategy, hot data can be stored in Solr for rapid access through the Ranger UI, and cold data can be archived to HDFS, or S3 with access provided through Ranger.

More Information

[Archiving and Purging Data](#)

9.3.2.4. Adding New Shards

If after reviewing the recommendations above, you need to add additional shards to your existing deployment, the following Solr documentation will help you understand how to accomplish that task: <https://archive.apache.org/dist/lucene/solr/ref-guide/apache-solr-ref-guide-5.5.pdf>

9.3.2.5. Out of Memory Exceptions

When using Ambari Infra with Ranger Audit, if you are seeing many instances of Solr exiting with Java “Out Of Memory” exceptions, a solution exists to update the Ranger Audit schema to use less heap memory by enabling DocValues. This change requires a re-index of data and is disruptive, but helps tremendously with heap memory consumption. Refer to this HCC article for the instructions on making this change: <https://community.hortonworks.com/articles/156933/restore-backup-ranger-audits-to-newly-collection.html>