

Hortonworks Data Platform

Apache Ambari User Guide

(September 21, 2016)

Hortonworks Data Platform: Apache Ambari User Guide

Copyright © 2012-2016 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. Overview: Ambari User Guide	1
1. Architecture	1
1.1. Sessions	2
2. Accessing Ambari Web	2
2. Viewing the Ambari Dashboards	3
1. Viewing the Cluster Dashboard	3
1.1. Scanning Service Status	4
1.2. Widget Descriptions	4
1.3. Widget Details	5
1.4. Linking to Service UIs	5
1.5. Viewing Cluster-Wide Metrics	6
2. Modifying the Cluster Dashboard	7
2.1. Adding a Widget to the Dashboard	8
2.2. Resetting the Dashboard	8
2.3. Customizing Widget Display	9
3. Viewing Cluster Heatmaps	9
3. Managing Hosts	11
1. Working with Hosts	11
2. Filtering the Hosts List	11
3. Performing Host-Level Actions	12
4. Viewing Components on a Host	13
5. Decommissioning Masters and Slaves	15
5.1. How to Decommission a Component	16
6. How to Delete a Component	16
7. Deleting a Host from a Cluster	17
7.1. How to Delete a Host from a Cluster	17
8. Setting Maintenance Mode	18
8.1. Setting Maintenance Mode for Services, Components, and Hosts	18
8.2. How to Turn On Maintenance Mode for a Service	18
8.3. How to Turn On Maintenance Mode for a Host	18
8.4. How to Turn On Maintenance Mode for a Host (alternative using filtering for hosts)	19
8.5. Maintenance Mode Use Cases	19
9. Adding Hosts to a Cluster	21
10. Rack Awareness	21
4. Managing Services	23
1. Starting and Stopping All Services	24
2. Selecting a Service	24
3. Adding a Service	24
3.1. Adding a Service to your Hadoop cluster	24
4. Editing Service Config Properties	28
5. Viewing Service Summary and Alerts	29
5.1. Alerts and Health Checks	29
5.2. Modifying the Service Dashboard	29
6. Performing Service Actions	32
7. Removing A Service	32
8. Monitoring Background Operations	32
9. Operations Audit	34

10. Using Quick Links	35
11. Rolling Restarts	35
11.1. Setting Rolling Restart Parameters	35
11.2. Aborting a Rolling Restart	36
12. Managing YARN	36
12.1. Refreshing YARN Capacity Scheduler	36
13. Managing HDFS	37
13.1. Rebalancing HDFS	37
13.2. Tuning Garbage Collection	37
14. Managing Storm	38
15. Managing Apache Atlas	38
5. Managing Service High Availability	39
1. NameNode High Availability	39
1.1. How To Configure NameNode High Availability	39
1.2. How to Roll Back NameNode HA	44
2. ResourceManager High Availability	52
2.1. How to Configure ResourceManager High Availability	52
2.2. How to Disable ResourceManager High Availability	53
3. HBase High Availability	55
3.1. Adding an HBase Master Component	55
4. Hive High Availability	55
4.1. Adding a Hive Metastore Component	56
4.2. Adding a HiveServer2 Component	56
4.3. Adding a WebHCat Component"?">	56
5. Storm High Availability	57
5.1. Adding a Nimbus Component	57
6. Oozie High Availability	57
6.1. Adding an Oozie Server Component	57
7. Apache Atlas High Availability	58
6. Managing Configurations	60
1. Configuring Services	60
1.1. Updating Service Properties	60
1.2. Restarting Components	60
2. Using Host Config Groups	60
3. Customizing Log Settings	62
4. Downloading Client Configs	63
5. Service Configuration Versions	63
5.1. Basic Concepts	63
5.2. Terminology	64
5.3. Saving a Change	64
5.4. Viewing History	65
5.5. Comparing Versions	66
5.6. Reverting a Change	67
5.7. Versioning and Host Config Groups	67
7. Administering the Cluster	69
1. Stack and Versions	69
2. Service Accounts	69
3. Kerberos	70
3.1. How To Regenerate Keytabs	71
3.2. How To Disable Kerberos	71
8. Monitoring and Alerts	72

1. Managing Alerts	72
1.1. Alert Types	73
1.2. Alert Check Counts	74
2. Configuring Notifications	75
2.1. Customizing Notification Templates	77
3. List of Predefined Alerts	79
3.1. HDFS Service Alerts	80
3.2. HDFS HA Alerts	83
3.3. NameNode HA Alerts	84
3.4. YARN Alerts	84
3.5. MapReduce2 Alerts	85
3.6. HBase Service Alerts	86
3.7. Hive Alerts	87
3.8. Oozie Alerts	87
3.9. ZooKeeper Alerts	87
3.10. Ambari Alerts	88
3.11. Ambari Metrics Alerts	88
9. Using Ambari Core Services	90
1. Ambari Metrics	90
1.1. AMS Architecture	90
1.2. Using Grafana	91
1.3. Performance Tuning	123
1.4. Moving the Metrics Collector	127
1.5. (Optional) Enabling Individual Region, Table, and User Metrics for HBase	127
2. Ambari Log Search (Technical Preview)	128
2.1. Log Search Architecture	128
2.2. Installing Log Search	129
2.3. Using Log Search	130
3. Ambari Infra	133

1. Overview: Ambari User Guide

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 a non-trivial task. 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-read and use, centralized web interface, Ambari Web.

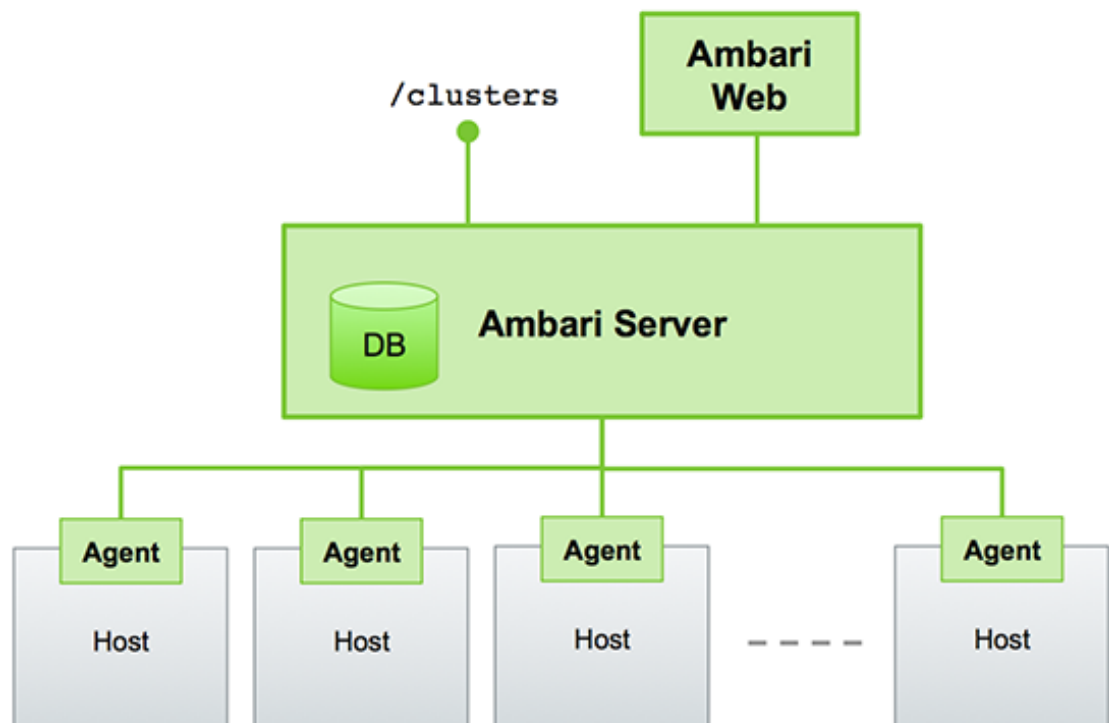
Ambari Web displays information such as service-specific summaries, graphs, and alerts. You use Ambari Web to create and manage your HDP cluster and to perform basic operational tasks such as starting and stopping services, adding hosts to your cluster, and updating service configurations. You also can use Ambari Web to perform administrative tasks for your cluster such as enabling Kerberos security and performing Stack upgrades.

For more information on administering Ambari users, groups and views, refer to [Hortonworks Data Platform Apache Ambari Administration](#).

1. Architecture

The Ambari Server serves as the collection point for data from across your cluster. Each host has a copy of the Ambari Agent - either installed automatically by the Install wizard or manually - which allows the Ambari Server to control each host.

Figure - Ambari Server Architecture



1.1. Sessions

Ambari Web is a client-side JavaScript application, which 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 via 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. Refer to [Ambari Web Inactivity Timeout](#) in Hortonworks Data Platform Apache Ambari Security for more information.

2. Accessing Ambari Web

Typically, you start the Ambari Server and Ambari Web as part of the installation process. If Ambari Server is stopped, you can start it using a command line editor on the Ambari Server host machine. Enter the following command:

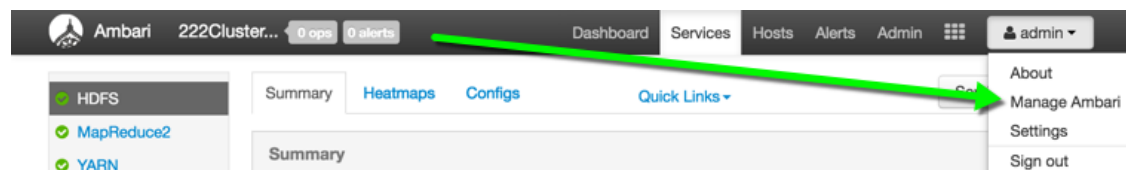
```
ambari-server start
```

To access Ambari Web, open a supported browser and enter the Ambari Web URL:

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

Enter your user name and password. If this is the first time Ambari Web is accessed, use the default values, `admin/admin`.

These values can be changed, and new users provisioned, using the Manage Ambari option.



For more information about managing users and other administrative tasks, see [Administering Ambari](#).

2. Viewing the Ambari Dashboards

This topic describes how to use Ambari Dashboards to monitor your HDP cluster. To see the Ambari Dashboards, select Dashboard, located at the top of the Ambari main window.



- [Viewing the Cluster Dashboard \[3\]](#)
- [Modifying the Cluster Dashboard \[7\]](#)
- [Viewing Cluster Heatmaps \[9\]](#)

1. Viewing the Cluster Dashboard

Ambari Web displays the **Dashboard** page as the home page. Use the **Dashboard** to view the operating status of your cluster. Each metrics widget displays status information for a single service in your HDP cluster. The Dashboard displays all metrics for the HDFS, YARN, HBase, and Storm services, and cluster-wide metrics by default. You can add and remove individual widgets, and rearrange the Dashboard by dragging and dropping each widget to a new location in the dashboard. Status information appears as simple pie and bar charts, more complex charts showing usage and load, sets of links to additional data sources, and values for operating parameters such as uptime and average RPC queue wait times. Most widgets display a single fact by default. For example, HDFS Disk Usage displays a load chart and a percentage figure.



Note

Each Service installed in your cluster also has a Service-specific dashboard. Refer to [Modifying the Service Dashboard](#) for more information.

1.1. Scanning Service Status

Notice the color of the dot appearing next to each component name in a list of components, services or hosts. The dot color and blinking action indicates operating status of each component, service, or host. For example, in the [Summary View](#), notice green dot next to each service name. The following colors and actions indicate service status:

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 the service name to open the **Services** screen, where you can see more detailed information on each service.

1.2. Widget Descriptions

The Dashboard includes metrics for the following services:

View **Metrics** that indicate the operating status of your cluster on the Ambari Dashboard. Each metrics widget displays status information for a single service in your HDP cluster. The Ambari Dashboard displays all metrics for the HDFS, YARN, HBase, and Storm services, and cluster-wide metrics by default.



Note

Metrics data for Storm is buffered and sent as a batch to Ambari every five minutes. After adding the Storm service, anticipate a five-minute delay for Storm metrics to appear.

You can add and remove individual widgets, and rearrange the dashboard by dragging and dropping each widget to a new location in the dashboard.

Status information appears as simple pie and bar charts, more complex charts showing usage and load, sets of links to additional data sources, and values for operating parameters such as uptime and average RPC queue wait times. Most widgets display a single fact by default. For example, HDFS Disk Usage displays a load chart and a percentage figure. The Ambari Dashboard includes metrics for the following services:

Ambari Service Metrics and Descriptions

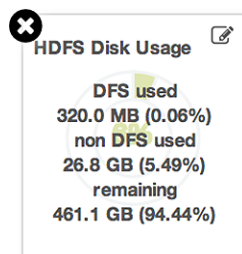
Metric:	Description:
HDFS	
HDFS Disk Usage	The Percentage of DFS used, which is a combination of DFS and non-DFS used.

Metric:	Description:
Data Nodes Live	The number of DataNodes live, as reported from the NameNode.
NameNode Heap	The percentage of NameNode JVM Heap 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 (HDP 2.1 or later Stacks)	
ResourceManager Heap	The percentage of ResourceManager JVM Heap used.
ResourceManager Uptime	The ResourceManager uptime calculation.
NodeManagers Live	The number of DataNodes live, as reported from the ResourceManager.
YARN Memory	The percentage of available YARN memory (used vs. total available).
HBase	
HBase Master Heap	The percentage of NameNode JVM Heap 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 (HDP 2.1 or later Stacks)	
Supervisors Live	The number of Supervisors live, as reported from the Nimbus server.

1.3. Widget Details

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

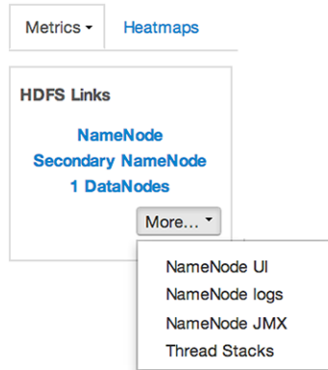
More detailed information about the service displays, as shown in the following example:



- To remove a widget from the mashup, click the white X.
- To edit the display of information in a widget, click the pencil icon. For more information about editing a widget, see [Customizing Metrics Display](#).

1.4. 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, using the links shown in the following example:



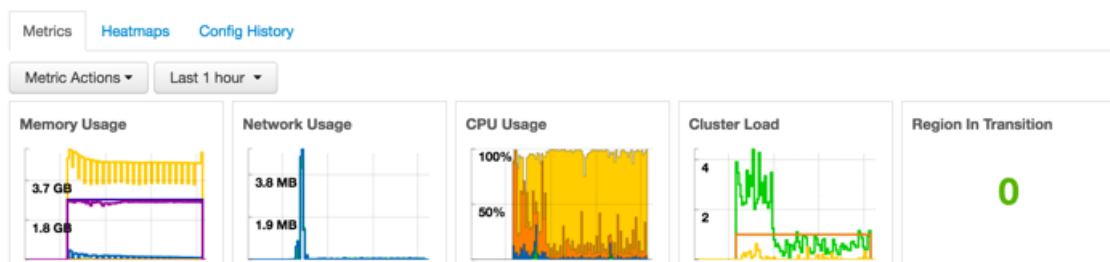
Choose the `More` drop-down to select from the list of links available for each service. The Ambari Dashboard includes additional links to metrics for the following services:

Links to More Metrics for HDP Services

Service:	Metric:	Description:
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.

1.5. Viewing Cluster-Wide Metrics

Cluster-wide metrics display information that represents your whole cluster. The Ambari Dashboard shows the following cluster-wide metrics:



Ambari Cluster-Wide Metrics and Descriptions

Metric:	Description:
Memory Usage	The cluster-wide memory utilization, including memory cached, swapped, used, shared.

Metric:	Description:
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.

- To remove a widget from the dashboard, click the white X.
- Hover your cursor over each cluster-wide metric to magnify the chart or itemize the widget display.
- To remove or add metric items from each cluster-wide metric widget, 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 pop-out window, as shown in the following example:



Use the pop-up window in the same ways that you use cluster-wide metric widgets on the dashboard.

To close the widget pop-up window, choose OK.

2. Modifying the Cluster Dashboard

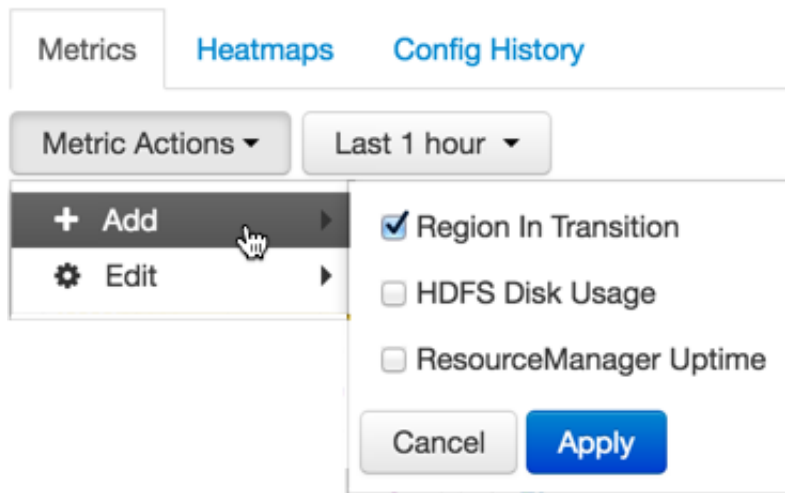
You can customize the Ambari Dashboard in the following ways:

- [Adding a Widget to the Dashboard \[8\]](#)
- [Resetting the Dashboard \[8\]](#)
- [Customizing Widget Display \[9\]](#)

2.1. Adding a Widget to the Dashboard

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

1. Select the Metrics drop-down, as shown in the following example:

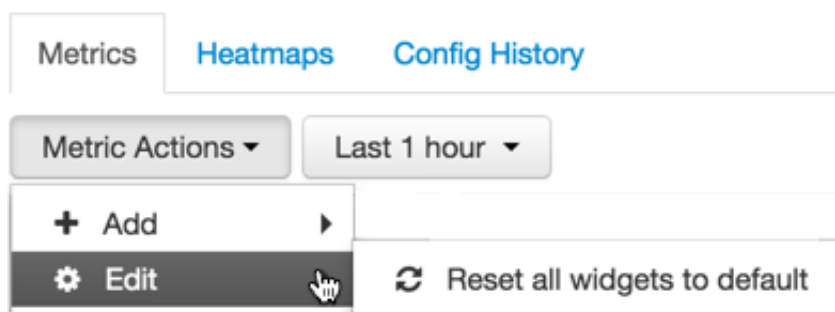


2. Choose Add.
3. Select a metric, such as Region in Transition.
4. Choose Apply.

2.2. Resetting the Dashboard

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

1. Select the Metrics drop-down, as shown in the following example:



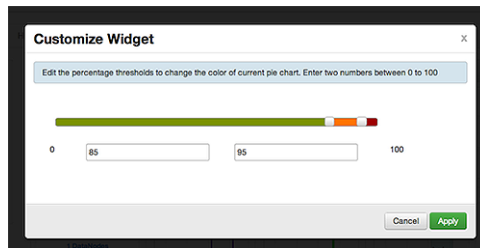
2. Choose Edit.
3. Choose Reset all widgets to default.

2.3. Customizing Widget Display

To customize the way a service widget displays metrics information:

1. Hover your cursor over a service widget.
2. Select the pencil-shaped, edit icon that appears in the upper-right corner.

The Customize Widget pop-up window displays properties that you can edit, as shown in the following example.



3. Follow the instructions in the Customize Widget pop-up 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, choose `Apply`.
5. To close the editor without saving any changes, choose `Cancel`.

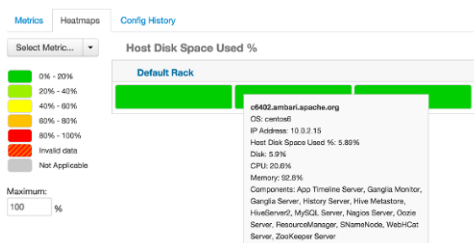


Note

Not all widgets support editing.

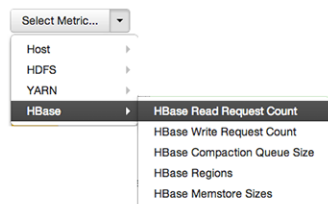
3. Viewing Cluster Heatmaps

Heatmaps provides a graphical representation of your overall cluster utilization using simple color coding.



A colored block represents each host in your cluster. To see more information about a specific host, hover over the block representing the host in which you are interested. A pop-up window displays 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 state is not available, the block displays "Invalid Data". Changing

the default maximum values for the heatmap lets you fine tune the representation. Use the Select Metric drop-down to select the metric type.



Heatmaps supports the following metrics:

Metric	Uses
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

Use Ambari Hosts to manage multiple 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 supports filtering your selection of host components, based on operating status, host health, and defined host groupings.

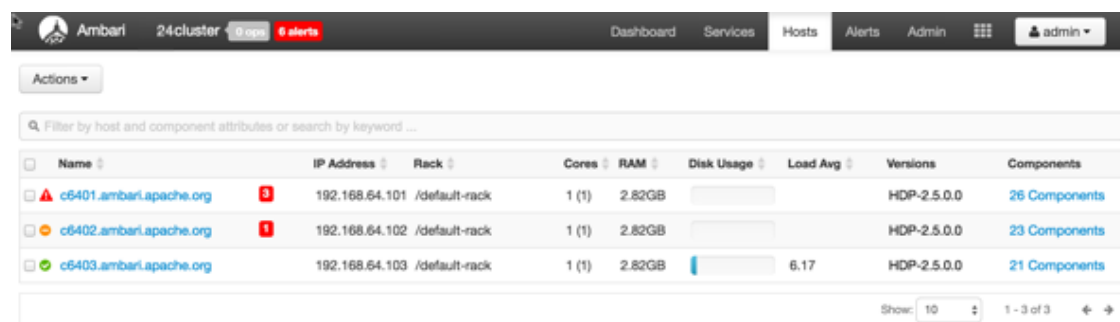
1. Working with Hosts

Use Hosts to view hosts in your cluster on which cluster components run. View individual hosts, listed by fully-qualified domain name, on the Hosts page. Use options in the Actions menu to perform actions on one or more hosts in your cluster. A colored dot beside each host name indicates operating status of each host, as follows:

- Red - At least one master component on that host is down. Hover to see a tooltip that lists affected components.
- 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.

A red condition flag overrides an orange condition flag, which overrides a yellow condition flag. In other words, a host having a master component down may also have other issues. The following example shows three hosts, one having a master component down, one having a slave component down, and one healthy. Warning indicators appear next to hosts having a component down.

View individual hosts, listed by fully-qualified domain name, on the Hosts landing page.



The screenshot shows the Ambari interface with the 'Hosts' tab selected. The table below represents the data shown in the screenshot:

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





To the right of the host name, if any hosts are in [Maintenance Mode](#) or are experiencing [alerts](#), an indicator will be shown.

2. Filtering the Hosts List

You can filter by host and component attributes or search by keyword using the search box. The available search types include:

- **Search by Host Attribute.** Search by host name, IP, host status and other attributes.
- **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, etc).
- **Search by keyword.** Type what you are looking for in the search box and this becomes a text filter.

Examples:

Search	Example
Find all hosts with a DataNode	
Find all the hosts with a DataNode that are stopped	
Find all the hosts with an HDFS component	
Find all the hosts with an HDFS or HBase component	

3. Performing Host-Level Actions

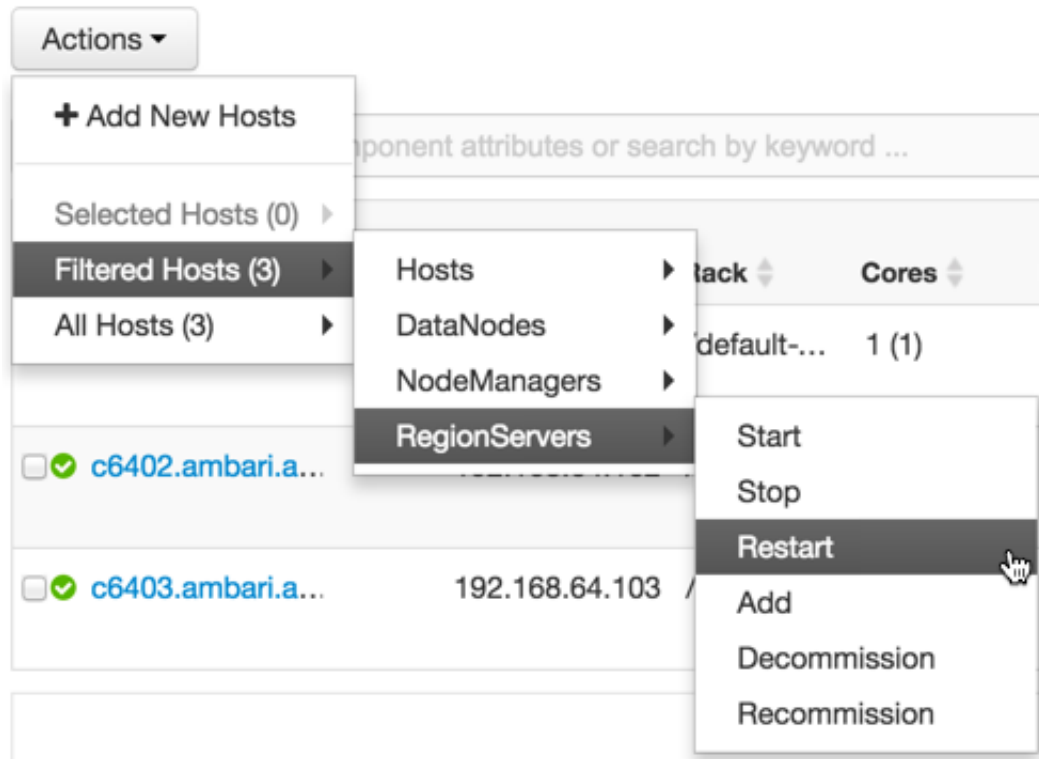
Use Actions to act on one, or multiple hosts in your cluster. Actions performed on multiple hosts are also known as bulk operations.

Actions comprises three menus that list the following option types:

- **Hosts** - lists selected, filtered or all hosts options, based on your selections made using Hosts home and Filters.
- **Objects** - lists component objects that match your host selection criteria.
- **Operations** - lists all operations available for the component objects you selected.

For example, to restart DataNodes on one host:

1. In Hosts, select a host running at least one DataNode.
2. In Actions, choose `Selected Hosts > DataNodes > Restart`, as shown in the following image.



3. Choose OK to confirm starting the selected operation.
4. Optionally, use [Monitoring Background Operations](#) to follow, diagnose or troubleshoot the restart operation.

4. Viewing Components on a Host

To manage components running on a specific host, choose a FQDN on the Hosts page. For example, choose `c6403.ambari.apache.org` in the default example shown. Summary-Components lists all components installed on that host.

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

- Summary:** Shows host details including IP Address (192.168.64.103), Rack (/default-rack), OS (centos6 (x86_64)), Cores (CPU): 1 (1), Disk: 34.5GB/489.68GB (7.05% used), Memory: 2.82GB, Load Avg: 2.42, Heartbeat: 19 minutes ago, and Current Version: 2.5.0.0-687.
- Components:** Lists installed components 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:** Displays six real-time monitoring graphs for the last hour:
 - CPU Usage:** A bar chart showing usage levels, with a peak at 100%.
 - Disk Usage:** A line graph showing disk usage at 372.5 GB, with a secondary value of 186.2 GB.
 - Load:** A line graph showing system load, with a peak of 4.
 - Memory Usage:** A line graph showing memory usage at 3.7 GB, with a secondary value of 1.8 GB.
 - Network Usage:** A bar chart showing network usage, with a peak of 78.1 KB and other values like 58.6 KB, 38.0 KB, and 19.5 KB.
 - Processes:** A line graph showing the number of processes, with a value of 50.

Choose options in `Host Actions`, to start, stop, restart, delete, or turn on maintenance mode for all components installed on the selected host.

Alternatively, choose action options from the drop-down menu next to an individual component on a host. The drop-down menu shows current operation status for each component. For example, you can decommission, restart, or stop the DataNode component (started) for HDFS, by selecting one of the options shown in the following example:

The screenshot shows the 'Components' page in Ambari. It lists various services with their status. A context menu is open over the 'Started' dropdown for the 'Metrics Monitor' component, showing options: Decommission, Restart, Stop, Turn On Maintenance Mode, and Delete.

Component	Status
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

5. Decommissioning Masters and Slaves

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

- DataNodes
- NodeManagers
- RegionServers

Decommissioning executes the following tasks:

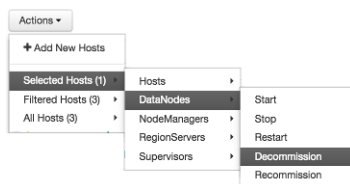
- For DataNodes, safely replicates the HDFS data to other DataNodes in the cluster.
- For NodeManagers, stops accepting new job requests from the masters and stops the component.
- For RegionServers, turns on drain mode and stops the component.

5.1. How to Decommission a Component

To decommission a component using Ambari Web, browse **Hosts** to find the host FQDN on which the component resides.

Using **Actions**, select **HostsComponent Type**, then choose **Decommission**.

For example:



The UI shows "Decommissioning" status while steps process, then "Decommissioned" when complete.



6. How to Delete a Component

To delete a component using Ambari Web, on **Hosts** choose the host FQDN on which the component resides.

1. In **Components**, find a decommissioned component.
2. Stop the component, if necessary.



Note

A decommissioned slave component may restart in the decommissioned state.

3. For a decommissioned component, choose **Delete** from the component drop-down menu.



Note

Restarting services enables Ambari to recognize and monitor the correct number of components.

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.

7. Deleting a Host from a Cluster

Deleting a host removes the host from the cluster. 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 Maintenance Mode, if necessary, for the host.

7.1. How to Delete a Host from a Cluster

1. In Hosts, click on a host name.
2. On the Host-Details page, select Host Actions drop-down menu.
3. Choose Delete.

If you have not completed prerequisite steps, a warning message similar to the following one appears:

Unable to Delete Host X

⚠ This host cannot be deleted since the following components are running:
DataNode, Ganglia Monitor, NodeManager, ZooKeeper Server

To delete this host, you must first stop all the running components listed above. If this host has a DataNode, it should be decommissioned first to prevent data loss.

OK

8. Setting Maintenance Mode

Maintenance Mode supports suppressing alerts and skipping bulk operations for specific services, components and hosts in an Ambari-managed cluster. You typically turn on Maintenance Mode when performing hardware or software maintenance, changing configuration settings, troubleshooting, decommissioning, or removing cluster nodes. You may place a service, component, or host object in Maintenance Mode before you perform necessary maintenance or troubleshooting tasks.

Maintenance Mode affects a service, component, or host object in the following two ways:

- Maintenance Mode suppresses alerts, warnings and status change indicators generated for the object
- Maintenance Mode exempts an object from host-level or service-level bulk operations

Explicitly turning on Maintenance Mode for a service implicitly turns on Maintenance Mode for components and hosts that run the service. While Maintenance Mode On prevents bulk operations being performed on the service, component, or host, you may explicitly start and stop a service, component, or host having Maintenance Mode On.

8.1. Setting Maintenance Mode for Services, Components, and Hosts

For example, examine using Maintenance Mode in a 3-node, Ambari-managed cluster installed using default options. This cluster has one data node, on host c6403. This example describes 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.

8.2. How to Turn On 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.

8.3. How to Turn On Maintenance Mode for a Host

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.

8.4. How to Turn On Maintenance Mode for a Host (alternative using filtering for hosts)

1. Using Hosts, select c6403.ambari.apache.org.
2. In Actions > Selected Hosts > Hosts choose Turn On Maintenance Mode.
3. Choose OK to confirm.

Notice that Maintenance Mode turns on for host c6403.ambari.apache.org.

Your list of Hosts now shows Maintenance Mode On for hosts c6401 and c6403.

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

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

- Hover your cursor over each Maintenance Mode icon appearing in the Hosts list.
 - Notice that hosts c6401 and c6403 have Maintenance Mode On.
 - Notice that on host c6401; HBaseMaster, HDFS client, NameNode, and ZooKeeper Server have Maintenance Mode turned On.
 - Notice on host c6402, that HDFS client and Secondary NameNode have Maintenance Mode On.
 - Notice on host c6403, that 15 components have Maintenance Mode On.
- The following behavior also results:
 - Alerts are suppressed for the DataNode.
 - DataNode is skipped from HDFS Start/Stop/Restart All, Rolling Restart.
 - DataNode is skipped from all Bulk Operations except Turn Maintenance Mode ON/OFF.
 - DataNode is skipped from Start All and / Stop All components.
 - DataNode is skipped from a host-level restart/restart all/stop all/start.

8.5. Maintenance Mode Use Cases

Four common Maintenance Mode Use Cases follow:

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

You want to:

- 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.

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

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

You want:

- No alerts generated by any components in this service.
- To prevent host-level or service-level bulk operations from starting, stopping, or restarting components in this service.

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

3. You turn off a service completely.

You want:

- The service to generate no warnings.
- To ensure that no components start, stop, or restart due to host-level actions or bulk operations.

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

4. A host component is generating alerts.

You want to:

- 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, turn on Maintenance Mode explicitly 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.

9. Adding Hosts to a Cluster

To add new hosts to your cluster, 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 Install Wizard. Follow the prompts, providing information similar to that provided to define the first set of hosts in your cluster.

10. Rack Awareness

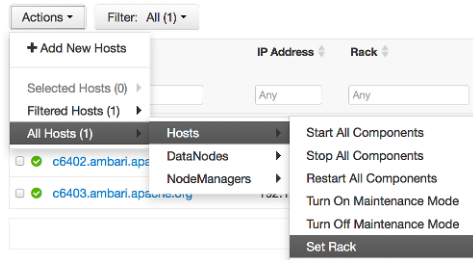
Ambari can manage Rack information for hosts. By setting the Rack ID, Ambari can display the hosts in heatmaps by Rack ID, as well users can filter & find hosts based on Rack ID on the Hosts page.

If HDFS is installed in your cluster, Ambari will pass this Rack ID information to HDFS via 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.

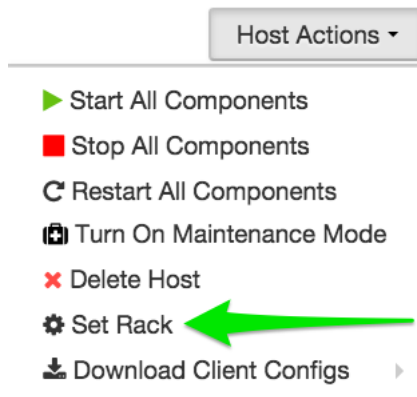
Setting Rack ID

There are two methods in Ambari Web for setting the Rack ID. You can set the Rack ID for hosts in bulk on the Hosts page using the **Actions** menu; and you can set the Rack ID on an individual host by viewing the Host page using the **Host Actions** menu.

To set the Rack ID in bulk on the Hosts page, use the **Actions** menu and select **Hosts** > **Set Rack** (for All, Filtered or Selected hosts).



To set the Rack ID on an individual host, browse to the Host page, use the Host Actions menu and select Set Rack.



Using a Custom Topology Script

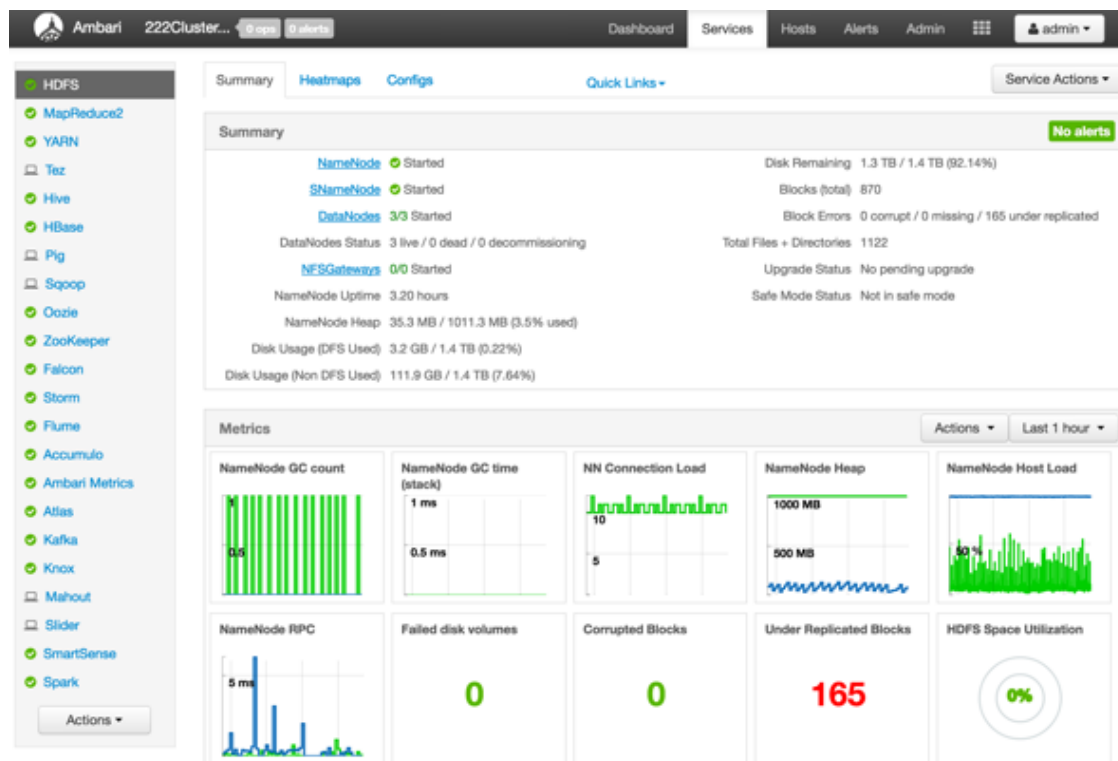
It is possible to not have Ambari manage the Rack information for hosts. Instead, you can use a custom topology script to provide rack information to HDFS and not use the Ambari-generated topology.py script. If you choose to manage Rack information on your own, you will need to **create your own topology script and manage distributing the script to all hosts**. Ambari will also not have any knowledge of host Rack information so heatmaps will not display by Rack in Ambari Web.

To manage Rack information on your own, in the `Services > HDFS > Configs`, modify the `net.topology.script.file.name` property. Set this property value to your own custom topology script (for example `/etc/hadoop/conf/topology.sh`). Distribute that topology script to your hosts and manage the Rack mapping information for your script outside of Ambari.

4. Managing Services

Use **Services** to monitor and manage selected services running in your Hadoop cluster.

All services installed in your cluster are listed in the leftmost **Services** panel.



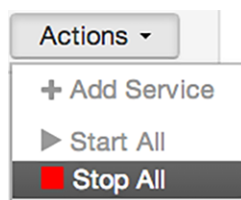
Services supports the following tasks:

- [Starting and Stopping All Services \[24\]](#)
- [Selecting a Service \[24\]](#)
- [Adding a Service \[24\]](#)
- [Editing Service Config Properties \[28\]](#)
- [Performing Service Actions \[32\]](#)
- [Removing A Service \[32\]](#)
- [Monitoring Background Operations \[32\]](#)
- [Operations Audit \[34\]](#)
- [Using Quick Links \[35\]](#)
- [Rolling Restarts \[35\]](#)

- [Managing YARN \[36\]](#)
- [Managing HDFS \[37\]](#)
- [Managing Storm \[38\]](#)

1. Starting and Stopping All Services

To start or stop all listed services at once, select **Actions**, then choose **Start All** or **Stop All**, as shown in the following example:



2. Selecting a Service

Selecting a service name from the list shows current summary, alert, and health information for the selected service. To refresh the monitoring panels and show information about a different service, select a different service name from the list.

Notice the colored dot next to each service name, indicating service operating status and a small, red, numbered rectangle indicating any alerts generated for the service.

3. Adding a Service

The Ambari install wizard installs all available Hadoop services by default. You may choose to deploy only some services initially, then add other services at later times. For example, many customers deploy only core Hadoop services initially. **Add Service** supports deploying additional services without interrupting operations in your Hadoop cluster. When you have deployed all available services, **Add Service** displays disabled.

For example, if you are using HDP 2.2 Stack and did not install Falcon or Storm, you can use the **Add Service** capability to add those services to your cluster.

To add a service, select **Actions > Add Service**, then complete the following procedure using the Add Service Wizard.

3.1. Adding a Service to your Hadoop cluster

This example shows the Falcon service selected for addition.

1. Choose **Services**.

Choose an available service. Alternatively, choose all to add all available services to your cluster. Then, choose **Next**. The Add Service wizard displays installed services highlighted green and check-marked, not available for selection.

Add Service Wizard

ADD SERVICE WIZARD

Choose Services

Assign Masters
 Assign Slaves and Clients
 Customize Services
 Configure Identities
 Review
 Install, Start and Test
 Summary

Choose Services

Choose which services you want to install on your cluster.

Service	Version	Description
<input checked="" type="checkbox"/> HDFS	2.7.1.2.4	Apache Hadoop Distributed File System
<input checked="" type="checkbox"/> YARN + MapReduce2	2.7.1.2.4	Apache Hadoop NextGen MapReduce (YARN)
<input checked="" type="checkbox"/> Tez	0.7.0.2.4	Tez is the next generation Hadoop Query Processing framework written on top of YARN.
<input checked="" type="checkbox"/> Hive	1.2.1.2.4	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service
<input checked="" type="checkbox"/> HBase	1.1.2.2.4	A Non-relational distributed database, plus Phoenix, a high performance SQL layer for low latency applications.
<input checked="" type="checkbox"/> Pig	0.15.0.2.4	Scripting platform for analyzing large datasets
<input checked="" type="checkbox"/> Sqoop	1.4.6.2.4	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases
<input checked="" type="checkbox"/> Oozie	4.2.0.2.4	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 .
<input checked="" type="checkbox"/> ZooKeeper	3.4.6.2.4	Centralized service which provides highly reliable distributed coordination
<input checked="" type="checkbox"/> Falcon	0.6.1.2.4	Data management and processing platform
<input checked="" type="checkbox"/> Storm	0.10.0.2.4	Apache Hadoop Stream processing framework
<input checked="" type="checkbox"/> Flume	1.5.2.2.4	A distributed service for collecting, aggregating, and moving large amounts of streaming data into HDFS
<input checked="" type="checkbox"/> Accumulo	1.7.0.2.4	Robust, scalable, high performance distributed key/value store.
<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	0.5.0.2.4	Atlas Metadata and Governance platform
<input checked="" type="checkbox"/> Kafka	0.9.0.2.4	A high-throughput distributed messaging system
<input checked="" type="checkbox"/> Knox	0.6.0.2.4	Provides a single point of authentication and access for Apache Hadoop services in a cluster
<input checked="" type="checkbox"/> Mahout	0.9.0.2.4	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 type="checkbox"/> Ranger	0.5.0.2.4	Comprehensive security for Hadoop
<input type="checkbox"/> Ranger KMS	0.5.0.2.4	Key Management Server
<input checked="" type="checkbox"/> Slider	0.60.0.2.4	A framework for deploying, managing and monitoring existing distributed applications on YARN.
<input checked="" type="checkbox"/> SmartSense	1.2.1.0-161	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	1.6.0.2.4	Apache Spark is a fast and general engine for large-scale data processing.

Next →



Note

Ambari 2.0 supports adding Ranger and Spark services, using the Add Services Wizard.

<input checked="" type="checkbox"/> Ranger	0.4.0	Comprehensive security for Hadoop
<input type="checkbox"/> Slider	0.60.0.2.2	A framework for deploying, managing and monitoring existing distributed applications on YARN.
<input checked="" type="checkbox"/> Spark	1.2.0.2.2	Apache Spark is a fast and general engine for large-scale data processing.
<input type="checkbox"/> Kafka	0.8.1.2.2	A high-throughput distributed messaging system
<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

For more information about installing Ranger, see [Installing Ranger](#).

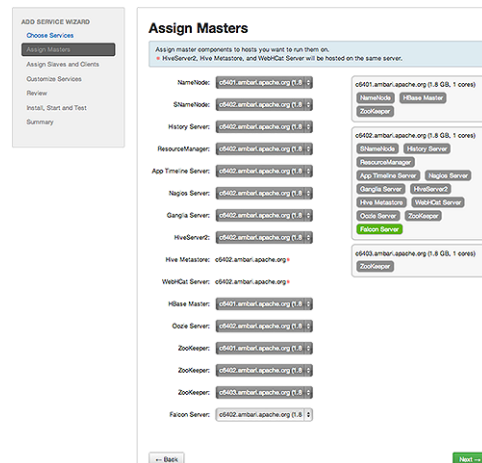
For more information about Installing Spark, see [Installing Spark](#).

- In **Assign Masters**, confirm the default host assignment. Alternatively, choose a different host machine to which master components for your selected service will be added. Then, choose Next.

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.

Using the drop-down, choose an alternate host name, if necessary.

- A green label located on the host to which its master components will be added, or
- An active drop-down list on which available host names appear.



- In **Assign Slaves and Clients**, accept the default assignment of slave and client components to hosts. Then, choose **Next**.

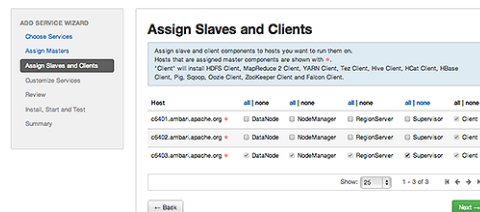
Alternatively, select hosts on which you want to install slave and client components. You must select at least one host for the slave of each service being added.

Host Roles Required for Added Services

Service Added	Host Role Required
YARN	NodeManager

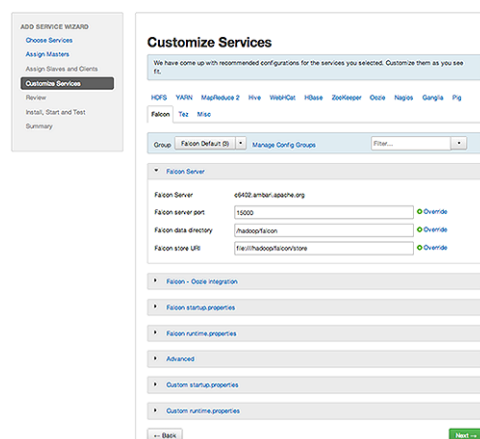
Service Added	Host Role Required
HBase	RegionServer

The Add Service Wizard skips and disables the Assign Slaves and Clients step for a service requiring no slave nor client assignment.

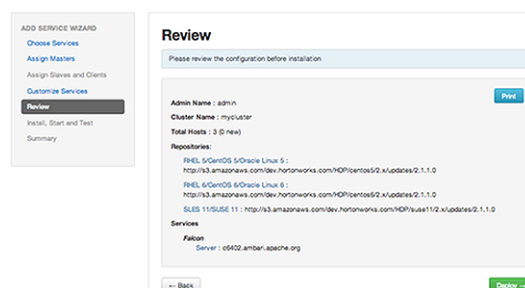


- In **Customize Services**, accept the default configuration properties.

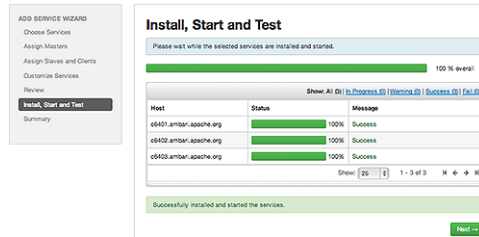
Alternatively, edit the default values for configuration properties, if necessary. Choose **Override** to create a configuration group for this service. Then, choose **Next**.



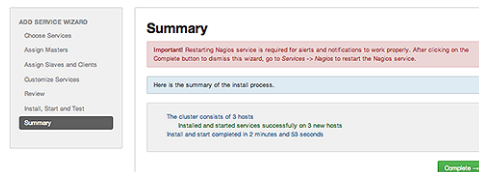
- In **Review**, make sure the configuration settings match your intentions. Then, choose **Deploy**.



- Monitor the progress of installing, starting, and testing the service. When the service installs and starts successfully, choose **Next**.



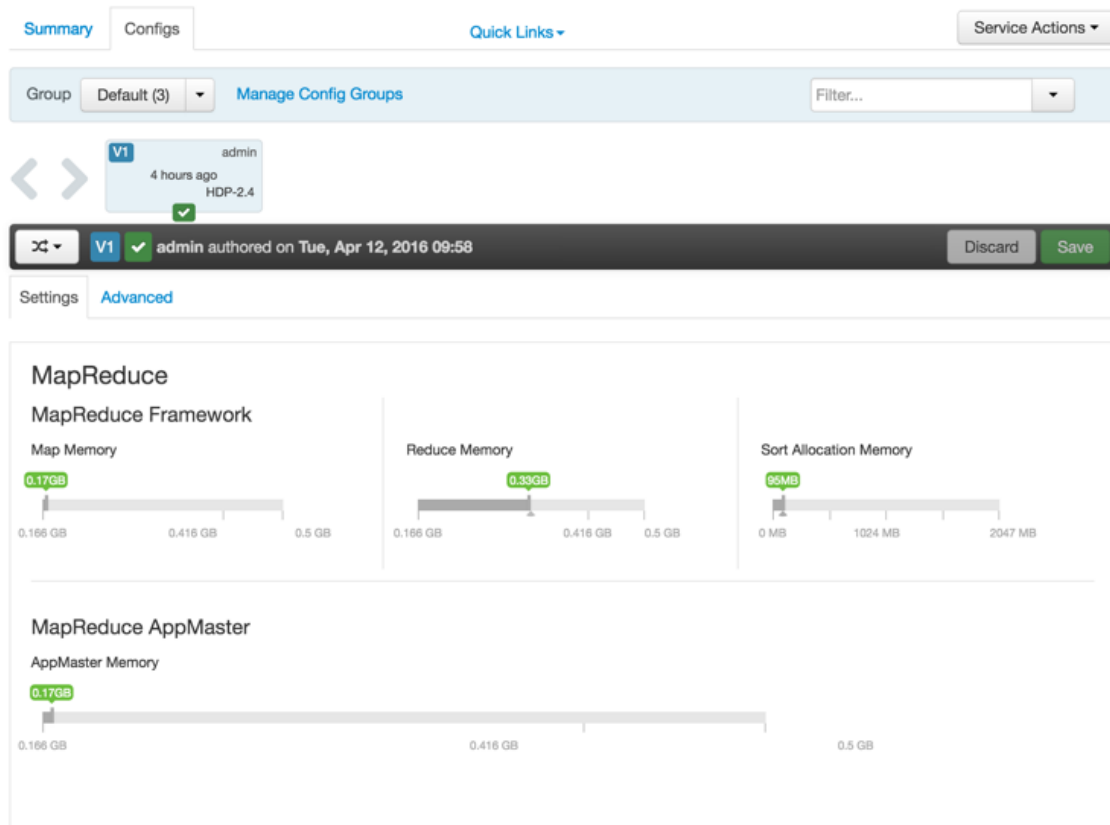
7. Summary displays the results of installing the service. Choose **Complete**.



8. Restart any other components having stale configurations.

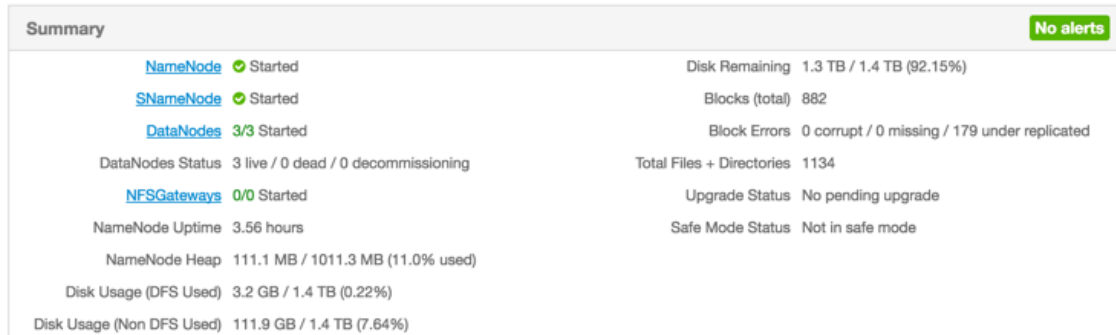
4. Editing Service Config Properties

Select a service, then select **Configs** to view and update configuration properties for the selected service. For example, select MapReduce2, then select Configs. Expand a config category to view configurable service properties. For example, select General to configure Default virtual memory for a job's map task.



5. Viewing Service Summary and Alerts

After you select a service, the **Summary** tab displays basic information about the selected service.

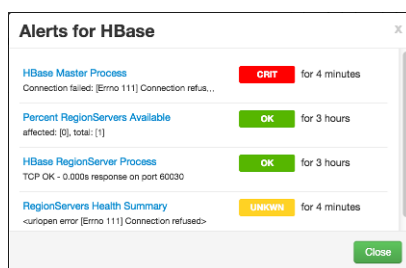


Select one of the **View Host** links, as shown in the following example, to view components and the host on which the selected service is running.

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

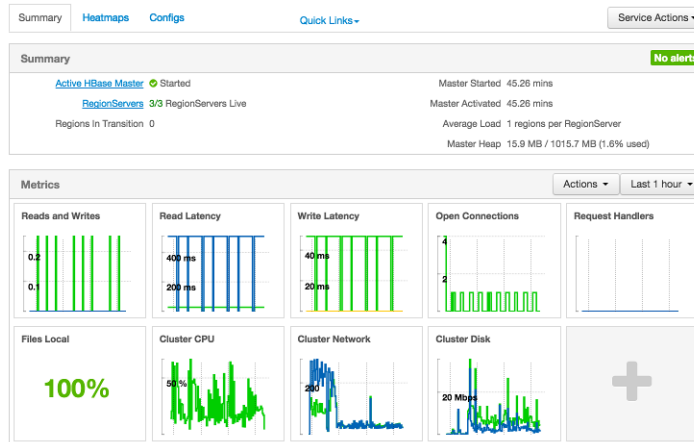
5.1. Alerts and Health Checks

On each Service page, in the Summary area, click **Alerts** to see a list of all health checks and their status for the selected service. Critical alerts are shown first. Click the text title of each alert message in the list to see the alert definition. For example, On the HBase > Services, click Alerts. Then, in Alerts for HBase, click HBase Master Process.



5.2. Modifying the Service Dashboard

Depending on the Service, the Summary tab includes a Metrics section which is by default populated with important service metrics to monitor.



This section of Metrics is customizable. You can add and remove widgets from the Dashboard as well as create new widgets. Widgets can be **private** only to you and your dashboard or **shared** in a Widget Browser library for other Ambari users to add/remove the widget from their Dashboard.



Important

You must have the Ambari Metrics service installed to be able to view, create, and customize the Service Dashboard. Only HDFS, Hive, HBase, and YARN have customizable service dashboards.

5.2.1. Adding or Removing 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.

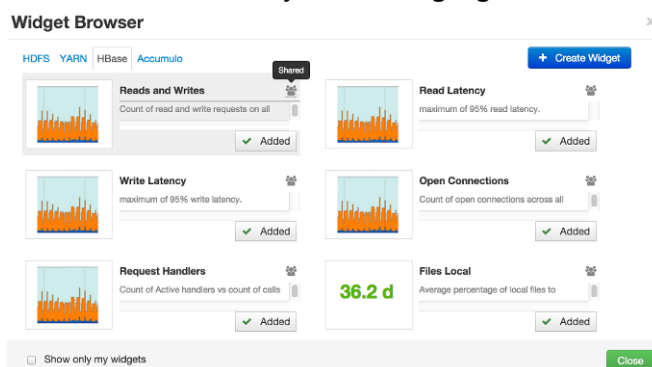
3. If you want to only display the widgets you have created, click the “Show only my widgets” checkbox to filter the Widget Browser.
4. If a widget is already added to your dashboard, it is shown as **Added**. Click to remove.
5. If a widget is not already added, you can click **Add**.

5.2.2. Creating a Widget

1. Click on the “ + ” to launch the Widget Browser. Click the **Create Widget** button. Alternatively, you can choose the Actions menu in the Metrics header to **Create Widget**. This launches the Create Widget wizard.
2. Select the type of widget to create.
3. Depending on the service and type of widget, you can select metrics and use operators to create an Expression that will be displayed in the widget. A preview of the widget is displayed as you build the expression.
4. Enter the widget name and description. Optionally choose to Share the widget. Sharing the widget makes the widget available to all Ambari users for this cluster. Once a widget is shared, other Ambari Admins or Cluster Operators can modify or delete the widget. This cannot be undone.

5.2.3. 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.
6. Deleting a shared widget removes the widget from all users. This cannot be undone.

5.2.4. Export Widget Graph Data

You can export the metrics data from widget graphs using the Export capability. This capability is available for graph-type widgets.

1. Mouse over the widget graph. You will see an **Export** icon. Alternatively, if you click on the widget graph to zoom into the graph, the **Export** icon is present in the upper-right of the dialog.

2. Click on the **Export** icon. You can choose to export in CSV or JSON format.
3. Select the format and the download will begin.

5.2.5. Setting Display Timezone

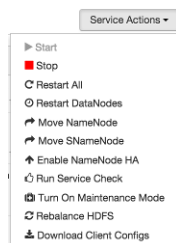
You can set the timezone used for displaying metrics data in widget graphs.

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

The Ambari Web UI will reload and graphs will be displayed using the timezone you have set.

6. Performing Service Actions

Manage a selected service on your cluster by performing service actions. In **Services**, select the **Service Actions** drop-down menu, then choose an option. Available options depend on the service you have selected. For example, HDFS service action options include:



Optionally, choose **Turn On Maintenance Mode** to suppress alerts generated by a service before performing a service action. 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. For more information about how Maintenance Mode affects bulk operations for host components, see [Setting Maintenance Mode](#).

7. Removing A Service

To remove a service, browse to the **Service** and select **Service Actions > Delete**. You will be prompted to remove any dependent services first and to stop all components for the service.

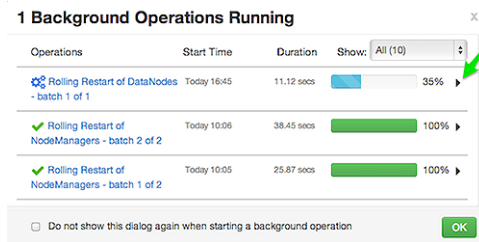
Once the service is stopped, you **must confirm** the delete to proceed. **This operation is not reversible and all configuration history will be lost.**

8. Monitoring Background Operations

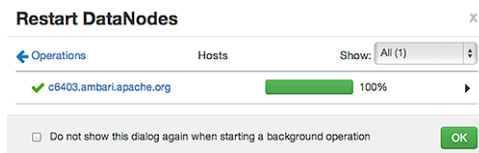
Optionally, use Background Operations to monitor progress and completion of bulk operations such as rolling restarts.

Background Operations opens by default when you run a job that executes bulk operations.

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



2. After restarts complete, Select the right-arrow, or a host name, to view log files and any error messages generated on the selected host.



3. You can choose to copy, open, or view logs for this operation directly from the Background Ops UI.

Choose Host Logs to view log information for the Host on the [Host Log Details tab](#).

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

Optionally, select the option to not show the bulk operations dialog.

9. Operations Audit

When you perform operations in Ambari, such as user login/logout, stopping/starting a service, and adding/removing a service, Ambari creates entries in an audit log. Using the audit log, you can determine who performed the operation and when the operation was performed as well as other, operation-specific information. You can find the Ambari Audit log at: `/var/log/ambari-server/ambari-audit.log`



Note

Making a Service **Configuration change** creates an entry in the audit log, and creates a specific log file `ambari-config-changes.log` for configuration changes that provides even more detail on the change. For example:

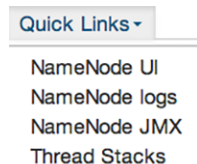
```

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'

```

10. 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 native NameNode GUI, NameNode logs, the NameNode JMX output, and thread stacks for the HDFS service. Quick Links are not available for every service.



11. Rolling Restarts

When you restart multiple services, components, or hosts, use rolling restarts to distribute the task. A rolling restart stops, then starts multiple, running slave components such as DataNodes, NodeManagers, RegionServers, or Supervisors, using a batch sequence. You set rolling restart parameter values to control the number of, time between, tolerance for failures, and limits for restarts of many components across large clusters.

To run a rolling restart:

1. Select a Service, then link to a lists of specific components or hosts that **Require Restart**.
2. Select **Restart**, then choose a slave component option.
3. Review and set values for **Rolling Restart Parameters**.
4. Optionally, reset the flag to only restart components with changed configurations.
5. Choose **Trigger Restart**.

Use [Monitor Background Operations](#) to monitor progress of rolling restarts.



Important

Rolling Restarts of DataNodes is recommended to only be performed during a cluster maintenance window.

11.1. Setting Rolling Restart Parameters

When you choose to restart slave components, 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 3-node cluster restarts one component at a time, waits two minutes between restarts, will proceed if only one failure occurs, and restarts all existing components that run this service.

If you trigger a rolling restart of components, Restart components with stale configs defaults to true. If you trigger a rolling restart of services, Restart services with stale configs defaults to 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

Rolling restart parameter values must satisfy the following criteria:

Validation Rules for Rolling Restart Parameters

Parameter	Required	Value	Description
Batch Size	Yes	Must be an integer > 0	Number of components to include in each restart batch.
Wait Time	Yes	Must be an integer > = 0	Time (in seconds) to wait between queuing each batch of components.
Tolerate up to x failures	Yes	Must be an integer > = 0	Total number of restart failures to tolerate, across all batches, before halting the restarts and not queuing batches.

11.2. Aborting a Rolling Restart

To abort future restart operations in the batch, choose 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

✓ 66403.ambari.apache.org 100%

Do not show this dialog again when starting a background operation

12. Managing YARN

This section contains information on performing operations specific to YARN.

- [Refreshing YARN Capacity Scheduler \[36\]](#)



Note

See [ResourceManager High Availability](#) for more information on setting up YARN for high availability.

12.1. Refreshing YARN Capacity Scheduler

This topic describes how to “refresh” the Capacity Scheduler from Ambari in cases where you have added or modified existing queues. After you modify the Capacity Scheduler

configuration, YARN supports refreshing the queues without requiring you to restart your ResourceManager. The “refresh” operation is valid if you have made no destructive changes to your configuration. Removing a queue is an example of a destructive change.

1. In Ambari Web, browse to **Services > YARN > Summary**.
2. Select **Service Actions**, then choose **Refresh YARN Capacity Scheduler**.
3. Confirm you would like to perform this operation.
4. The refresh operation is submitted to the YARN ResourceManager.



Important

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.

13. Managing HDFS

This section contains information on performing operations specific to HDFS.

- [Rebalancing HDFS \[37\]](#)
- [Tuning Garbage Collection \[37\]](#)



Note

See [NameNode High Availability](#) for more information on setting up HDFS for high availability.

13.1. Rebalancing HDFS

This topic describes how you can initiate an HDFS rebalance from Ambari. HDFS provides a “balancer” utility to help balance the blocks across DataNodes in the cluster.

1. In Ambari Web, browse to **Services > HDFS > Summary**.
2. Select **Service Actions**, then choose **Rebalance HDFS**.
3. Enter the **Balance Threshold** value as a percentage of disk capacity.
4. Click **Start** to begin the rebalance. You can check rebalance progress or cancel a rebalance in process by opening the **Background Operations** dialog in Ambari.

13.2. 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 the old generation is almost fully occupied.

Initiating it in advance allows garbage collection to complete before the old generation is full, and thus avoids Full GC (which may result in "stop-the-world" pause behavior).

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 CMS should be triggered. Basically, it allows the creation of a buffer in heap, which can be filled with data while CMS is running. This percent should be back-calculated from the speed with which memory is consumed in the old generation during production load. If this percent is set too low, the CMS will run too often; if it is set too high, the CMS will be triggered too late and [concurrent mode failure](#) may occur. The default setting for `-XX:CMSInitiatingOccupancyFraction` is 70, which means that the application should utilize less than 70% of the old generation.

To modify the NameNode CMS GC parameters:

1. Using **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.

14. Managing Storm

See the [Storm Guide](#) for information about installing, configuring, and using Storm in a production environment.

15. Managing Apache Atlas

For more information about Atlas configuration, see [Installing and Configuring Atlas Using Ambari](#).

When you update the Atlas configuration settings in Ambari, Ambari marks the services that require restart, and you can select **Actions > Restart All Required** to restart all services that require a restart.



Important

Apache Oozie requires a restart after an Atlas configuration update, but may not be included in the services marked as requiring restart in Ambari. Select **Oozie > Service Actions > Restart All** to restart Oozie along with the other services.

5. Managing Service High Availability

Ambari provides the ability to configure the High Availability features available with the HDP Stack services. This section describes how to enable HA for the various Stack services.

- [NameNode High Availability \[39\]](#)
- [ResourceManager High Availability \[52\]](#)
- [HBase High Availability \[55\]](#)
- [Hive High Availability \[55\]](#)
- [Oozie High Availability \[57\]](#)
- [Apache Atlas High Availability \[58\]](#)

1. NameNode High Availability

To ensure that a NameNode in your cluster is always available if the primary NameNode host fails, enable and set up NameNode High Availability on your cluster using Ambari Web.

Follow the steps in the Enable NameNode HA Wizard.

For more information about using the Enable NameNode HA Wizard, see [How to Configure NameNode High Availability](#).

1.1. How To Configure NameNode High Availability

1. Check to make sure you have at least three hosts in your cluster and are running at least three ZooKeeper servers.
2. Check to make sure that the HDFS and ZooKeeper services are not in Maintenance Mode.

These services will be stopped and started 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.

3. In Ambari Web, select **Services > HDFS > Summary**.
4. Select **Service Actions** and choose **Enable NameNode HA**.
5. The Enable HA Wizard launches. This wizard describes the set of automated and manual steps you must take to set up NameNode high availability.
6. **Get Started** : This step gives you an overview of the process and allows you to select a Nameservice ID. You use this Nameservice ID instead of the NameNode FQDN once HA has been set up. Click **Next** to proceed.

ENABLE NAMENODE 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 →

- 7. Select Hosts** : Select a host for the additional NameNode and the JournalNodes. The wizard suggest options that you can adjust using the drop-down lists. Click **Next** to proceed.

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)

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

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

8. **Review** : 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

9. **Create Checkpoints** : Follow the instructions in the step. You need to log in to your **current** NameNode host to run the commands to put your NameNode into safe mode and create a checkpoint. When Ambari detects success, the message on the bottom of the window changes. 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 -o 'hdfs dfsadmin -safemode enter'
```
3. Once in Safe Mode, create a Checkpoint:

```
sudo su -l hdfs -o 'hdfs dfsadmin -saveCheckpoint'
```
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 -->](#)

10. Configure Components : The wizard configures your components, displaying progress bars to let you track the steps. Click **Next** to continue.

Configure Components

Please proceed to the next step.

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

[Next](#)

11. Initialize JournalNodes : Follow the instructions in the step. You need to login to your **current** NameNode host to run the command to initialize the JournalNodes. When Ambari detects success, the message on the bottom of the window changes. 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 -o 'hdfs namenode -initializeSharedEdits'
```
3. You will be able to proceed once Ambari detects that the JournalNodes have been initialized successfully.

JournalNodes initialized [Next -->](#)

12. Start Components : The wizard starts the ZooKeeper servers and the NameNode, displaying progress bars to let you track the steps. Click **Next** to continue.

Start Components

Please proceed to the next step.

- ✓ Start ZooKeeper Servers
- ✓ Start NameNode

[Next](#)

13. Initialize Metadata : Follow the instructions in the step. 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 **Next** when you have completed the two commands. A **Confirmation** pop-up window displays, reminding you to do both steps. Click **OK** to confirm.

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 namecode -bootstrapstandby"
```

Please proceed once you have completed the steps above.

Next →

14. Finalize HA Setup : The wizard the setup, displaying progress bars to let you track the steps. Click **Done** to finish the wizard. After the Ambari Web GUI reloads, you may see some alert notifications. Wait a few minutes until the services come back up. If necessary, restart any components using Ambari Web.

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

15. 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.

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`

16. 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-electors.zk.op.retries=120`

1.2. How to Roll Back NameNode HA

To roll back NameNode HA to the previous non-HA state use the following step-by-step manual process, depending on your installation.

1. [Stop HBase \[44\]](#)
2. [Checkpoint the Active NameNode \[45\]](#)
3. [Stop All Services \[45\]](#)
4. [Prepare the Ambari Server Host for Rollback \[45\]](#)
5. [Restore the HBase Configuration \[46\]](#)
6. [Delete ZooKeeper Failover Controllers \[47\]](#)
7. [Modify HDFS Configurations \[47\]](#)
8. [Recreate the secondary NameNode \[49\]](#)
9. [Re-enable the secondary NameNode \[50\]](#)
10. [Delete All JournalNodes \[50\]](#)
11. [Delete the Additional NameNode \[51\]](#)
12. [Verify the HDFS Components \[51\]](#)
13. [Start HDFS \[52\]](#)

1.2.1. Stop HBase

1. From Ambari Web, go to the Services view and select HBase.
2. Choose **Service Actions > Stop**.

3. Wait until HBase has stopped completely before continuing.

1.2.2. Checkpoint the Active NameNode

If HDFS has been in use **after** you enabled NameNode HA, but you wish to revert back 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 back, you can skip this step and move on to [Stop All Services](#).

- If Kerberos security has **not** been enabled on the cluster:

On the Active NameNode host, execute the following commands to save the namespace. You must be the HDFS service user to do this.

```
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:

```
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'
```

Where `<HDFS_USER>` is the HDFS service user; for example `hdfs`, `<HOSTNAME>` is the Active NameNode hostname, and `<REALM>` is your Kerberos realm.

1.2.3. Stop All Services

Browse to **Ambari Web > Services**, then choose **Stop All** in the Services navigation panel. You must wait until all the services are completely stopped.

1.2.4. Prepare the Ambari Server Host for Rollback

Log into the Ambari server host and set the following environment variables to prepare for the rollback procedure:

Variable	Value
<code>export AMBARI_USER=AMBARI_USERNAME</code>	Substitute the value of the administrative user for Ambari Web. The default value is <code>admin</code> .
<code>export AMBARI_PW=AMBARI_PASSWORD</code>	Substitute the value of the administrative password for Ambari Web. The default value is <code>admin</code> .
<code>export AMBARI_PORT=AMBARI_PORT</code>	Substitute the Ambari Web port. The default value is <code>8080</code> .
<code>export AMBARI_PROTO=AMBARI_PROTOCOL</code>	Substitute the value of the protocol for connecting to Ambari Web. Options are <code>http</code> or <code>https</code> . The default value is <code>http</code> .
<code>export CLUSTER_NAME=CLUSTER_NAME</code>	Substitute the name of your cluster, set during the Ambari Install Wizard process. For example: <code>mycluster</code> .
<code>export NAMENODE_HOSTNAME=NN_HOSTNAME</code>	Substitute the FQDN of the host for the non-HA NameNode. For example: <code>nn01.mycompany.com</code> .

Variable	Value
export ADDITIONAL_NAMENODE_HOSTNAME=ANN_HOSTNAME	Substitute the FQDN of the host for the additional NameNode in your HA setup.
export SECONDARY_NAMENODE_HOSTNAME=SNN_HOSTNAME	Substitute the FQDN of the host for the secondary NameNode for the non-HA setup.
export JOURNALNODE1_HOSTNAME=JOUR1_HOSTNAME	Substitute the FQDN of the host for the first Journal Node.
export JOURNALNODE2_HOSTNAME=JOUR2_HOSTNAME	Substitute the FQDN of the host for the second Journal Node.
export JOURNALNODE3_HOSTNAME=JOUR3_HOSTNAME	Substitute the FQDN of the host for the third Journal Node.

Double check that these environment variables are set correctly.

1.2.5. Restore the HBase Configuration

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

1. To check if your current HBase configuration needs to be restored, on the Ambari Server host:

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

Where the environment variables you set up in [Prepare the Ambari Server Host for Rollback](#) substitute for the variable names.

Look for the configuration property `hbase.rootdir`. If the value is set to the NameService ID you set up using the **Enable NameNode HA** wizard, you need to revert the `hbase-site` configuration set up back to non-HA values. If it points instead to a specific NameNode host, it does not need to be rolled back and you can go on to [Delete ZooKeeper Failover Controllers](#).

For example:

```
"hbase.rootdir": "hdfs://<name-service-id>:8020/apps/hbase/data"
```

The `hbase.rootdir` property points to the NameService ID and the value needs to be rolled back `"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.

2. If you need to 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.sh -
u <AMBARI_USER> -p<AMBARI_PW> -port <AMBARI_PORT> set
localhost <CLUSTER_NAME> hbase-site hbase.rootdir hdfs://
<NAMENODE_HOSTNAME>:8020/apps/hbase/data
```

Where the environment variables you set up in [Prepare the Ambari Server Host for Rollback](#) substitute for the variable names.

3. Verify that the `hbase.rootdir` property has been restored properly. On the Ambari Server host:

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

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

1.2.6. Delete ZooKeeper Failover Controllers

You may need to delete ZooKeeper (ZK) Failover Controllers.

1. To check if you need to delete ZK Failover Controllers, on the Ambari Server host:

```
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
```

If this returns an empty `items` array, you may proceed to [Modify HDFS Configuration](#). Otherwise you must use the following DELETE commands:

2. To delete all ZK Failover Controllers, on the Ambari Server host:

```
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
```

3. Verify that the ZK Failover Controllers have been deleted. On the Ambari Server host:

```
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.

1.2.7. Modify HDFS Configurations

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

1. To check if you need to modify your `hdfs-site` configuration, on the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -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.

2. To delete these properties, execute the following for **each property** you found. On the Ambari Server host:

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

Where you replace `property_name` with the name of **each** of the properties to be deleted.

3. Verify that all of the properties have been deleted. On the Ambari Server host: `/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> get localhost <CLUSTER_NAME> hdfs-site`

None of the properties listed above should be present.

4. To check if you need to modify your `core-site` configuration, on the Ambari Server host: `/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p <AMBARI_PW> -port <AMBARI_PORT> get localhost <CLUSTER_NAME> core-site`

5. If you see the property `ha.zookeeper.quorum`, it must be deleted. On the Ambari Server host:

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

6. If the property `fs.defaultFS` is set to the NameService ID, it must be reverted back to its non-HA value. For example:

```
"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>" The property
fs.defaultFS does not need to be changed as it points to a specific NameNode, not
to a NameService ID
```

7. To revert the property `fs.defaultFS` to the NameNode host value, on the Ambari Server host:

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

8. Verify that the `core-site` properties are now properly set. On the Ambari Server host:

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

The property `fs.defaultFS` should be set to point to the NameNode host and the property `ha.zookeeper.quorum` should not be there.

1.2.8. Recreate the secondary NameNode

You may need to recreate your secondary NameNode.

1. To check to see if 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 go on to [Re-enable secondary NameNode](#).

2. Recreate your secondary NameNode. On the Ambari Server host: `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>`

3. Verify that the secondary NameNode now exists. 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
```

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

1.2.9. Re-enable the secondary NameNode

- To re-enable the secondary NameNode, 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
```

- If this returns 200, go to [Delete All JournalNodes](#).
- If this returns 202, wait a few minutes and run the following command on the Ambari Server host:

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

When "state" : "INSTALLED" is in the response, go on to the next step.

1.2.10. Delete All JournalNodes

You may need to delete any JournalNodes.

1. To 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.

2. To delete the JournalNodes, on the Ambari Server host:

```
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
```

3. 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.

1.2.11. Delete the Additional NameNode

You may need to delete your Additional NameNode.

1. To 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.

2. To delete the Additional NameNode that was set up for HA, on the Ambari Server host:

```
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
```

3. 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.

1.2.12. Verify the HDFS Components

Make sure you have the correct components showing in HDFS.

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

- SNameNode
- DataNodes

You should **not** see any line for JournalNodes.

1.2.13. Start HDFS

1. In the **Ambari Web UI**, select **Service Actions**, then choose **Start**.

Wait until the progress bar shows that the service has completely started and has passed the service checks.

If HDFS does not start, you may need to repeat the previous step.

2. To start all of the other services, select **Actions > Start All** in the **Services** navigation panel.

2. ResourceManager High Availability



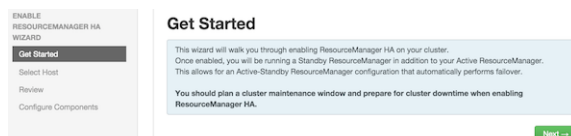
Note

This feature is available with HDP 2.2 or later.

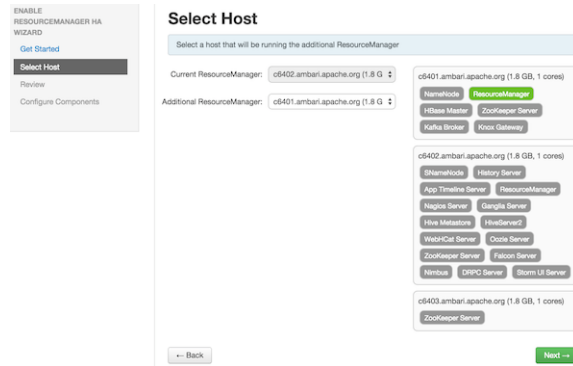
The following topic explains [How to Configure ResourceManager High Availability](#).

2.1. How to Configure ResourceManager High Availability

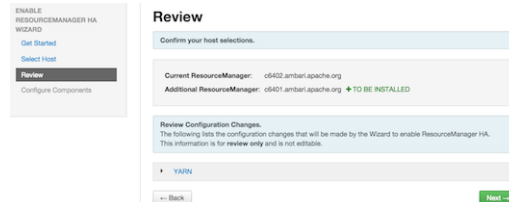
1. Check to make sure you have at least three hosts in your cluster and are running at least three ZooKeeper servers.
2. In Ambari Web, browse to **Services > YARN > Summary**. Select **Service Actions** and choose **Enable ResourceManager HA**.
3. The Enable ResourceManager HA Wizard launches. The wizard describes a set of automated and manual steps you must take to set up ResourceManager High Availability.
4. **Get Started**: This step gives you an overview of enabling ResourceManager HA. Click **Next** to proceed.



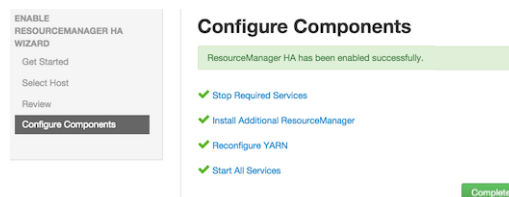
5. **Select Host**: The wizard shows you the host on which the current ResourceManager is installed and suggests a default host on which to install an additional ResourceManager. Accept the default selection, or choose an available host. Click **Next** to proceed.



6. **Review Selections:** The wizard shows you the host selections and configuration changes that will occur to enable ResourceManager HA. Expand YARN, if necessary, to review all the YARN configuration changes. Click **Next** to approve the changes and start automatically configuring ResourceManager HA.



7. **Configure Components:** The wizard configures your components automatically, displaying progress bars to let you track the steps. After all progress bars complete, click **Complete** to finish the wizard.



2.2. How to Disable ResourceManager High Availability

Use the following instructions to disable ResourceManager High Availability. You will be deleting one ResourceManager and keeping 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.



Important

These steps involve using the Ambari REST API. Be sure to test and verify these steps in a test environment prior to executing against a production environment.

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. For example, `yarn-site.json`.

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

where **ambari.server** is the hostname of your Ambari Server and **cluster.name** is the name of your cluster.

3. Modify the following priorities in the `yarn-site.json` file:

Property	Value
<code>yarn.resourcemanager.ha.enabled</code>	Change the value to false .
<code>yarn.resourcemanager.ha.rm-ids</code>	Delete this property.
<code>yarn.resourcemanager.hostname.rm1</code>	Delete this property.
<code>yarn.resourcemanager.hostname.rm2</code>	Delete this property.
<code>yarn.resourcemanager.webapp.address.rm1</code>	Delete this property.
<code>yarn.resourcemanager.webapp.address.rm2</code>	Delete this property.
<code>yarn.resourcemanager.webapp.https.address.rm1</code>	Delete this property.
<code>yarn.resourcemanager.webapp.https.address.rm2</code>	Delete this property.
<code>yarn.resourcemanager.cluster-id</code>	Delete this property.
<code>yarn.resourcemanager.ha.automatic-failover.zk-base-path</code>	Delete this property.

4. Verify the following properties in the `yarn-site.json` file are set to the ResourceManager hostname you will be keeping:

Property	Value
<code>yarn.resourcemanager.hostname</code>	<ResourceManager hostname>
<code>yarn.resourcemanager.admin.address</code>	<ResourceManager hostname>
<code>yarn.resourcemanager.webapp.address</code>	<ResourceManager hostname>
<code>yarn.resourcemanager.resource-tracker.address</code>	<ResourceManager hostname>
<code>yarn.resourcemanager.scheduler.address</code>	<ResourceManager hostname>
<code>yarn.resourcemanager.webapp.https.address</code>	<ResourceManager hostname>
<code>yarn.timeline-service.webapp.address</code>	<ResourceManager hostname>
<code>yarn.timeline-service.webapp.https.address</code>	<ResourceManager hostname>
<code>yarn.timeline-service.address</code>	<ResourceManager hostname>
<code>yarn.log.server.url</code>	<ResourceManager hostname>

5. Search the `yarn-site.json` file and remove any references to the ResourceManager hostname that you will be 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.sh set
ambari.server cluster.name yarn-site yarn-site.json
```

where **ambari.server** is the hostname of your Ambari Server and **cluster.name** is the name of your cluster.

- 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
```

where **ambari.server** is the hostname of your Ambari Server and **cluster.name** is the name of your cluster and **hostname** is the hostname of the ResourceManager you are deleting.

- In Ambari Web, start the ZooKeeper service. 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
```

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

3. HBase High Availability

During the HBase service install, depending on your component assignment, Ambari installs and configures one HBase Master component and multiple RegionServer components. To setup high availability for the HBase service, you can run two or more HBase Master components by [adding an HBase Master component](#). Once running two or more HBase Masters, HBase uses ZooKeeper for coordination of the active Master.

3.1. Adding an HBase Master Component

- In **Ambari Web**, browse to **Services > HBase**.
- In Service Actions, select the **+ Add HBase Master** option.
- Choose the host to install the additional HBase Master, then choose **Confirm Add**.

Ambari installs the new HBase Master and reconfigure HBase to handle multiple Master instances.

4. Hive High Availability

The Hive service has multiple, associated components. The primary Hive components are: Hive Metastore and HiveServer2. To setup high availability for the Hive service, you can run two or more of each of those components.



Note

This feature is available with HDP 2.2 or later.



Important

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.

4.1. Adding a Hive Metastore Component

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



Important

If you have Acid enabled in Hive, make sure that the **Run Compactor** setting is enabled (set to True) **on only one** Hive metastore host. To manage configuration settings for specific hosts using Ambari Web, select **Services > Hive > Configs > Manage Config Groups**, and set the Run Compactor setting to True on only one Hive Metastore host. For more information, see [Using Host Config Groups](#).

4.2. Adding a HiveServer2 Component

1. In Ambari Web, browse to the host where you would like to install another HiveServer2.
2. On the Host page, choose **+Add**.
3. Select **HiveServer2** from the list.
4. Ambari installs the new HiveServer2.

4.3. Adding a WebHCat Component">

1. In Ambari Web, browse to the host where you would like to install another WebHCat Server.
2. On the **Host** page, choose **+Add**.
3. Select **WebHCat** from the list.
4. Ambari installs the new WebHCat.
5. Ambari installs the component and reconfigures Hive to handle multiple Hive Metastore instances.

5. Storm High Availability



Note

This feature is available with HDP 2.3 or later.

To setup high availability for the Storm Nimbus server, you can run two or more Nimbus components by adding a Nimbus component from Ambari.

5.1. Adding a Nimbus Component

1. In Ambari Web, browse to **Services > Storm**.
2. In **Service Actions**, select the **+ Add Nimbus** option.
3. Choose the host to install the additional Nimbus, then choose **Confirm Add**.

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

6. Oozie High Availability



Note

This feature is available with HDP 2.2 or later.

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



Important

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 and therefore, you must use an existing relational database. When using Derby for the Oozie Server, you will not have an option to add Oozie Server components to your cluster.



Important

High availability for Oozie requires the use of an external Virtual IP Address or Load Balancer to direct traffic to the Oozie servers.

6.1. Adding an Oozie Server Component

1. In **Ambari Web**, browse to the host where you would like to install another Oozie Server.
2. On the **Host** page, click the **+Add** button.

3. Select **Oozie Server** from the list and Ambari will install the new Oozie Server.
4. After configuring your external Load Balancer, update the oozie configuration.
5. Browse to **Services > Oozie > Configs** and in oozie-site add the following:

Property	Value
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,or
oozie.base.url	http://<loadbalancer.hostname>:11000/oozie

6. In oozie-env, uncomment OOZIE_BASE_URL property and change value to point to the Load Balancer. For example:

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

7. Restart Oozie service for the changes to take affect.
8. Update HDFS configs for the Oozie proxy user. Browse to **Services > HDFS > Configs** and in **core-site** update the `hadoop.proxyuser.oozie.hosts` property to include the newly added Oozie Server host. Hosts should be comma separated.
9. Restart all needed services.

7. Apache Atlas High Availability

Use the following steps to set up High Availability (HA) for Apache Atlas.



Note

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

1. Select **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, select **Add > Atlas Metadata Server** and add the new Atlas Metadata Server. Ambari adds the new Atlas Metadata Server in a Stopped state.
3. Select **Atlas > Configs > Advanced**, then select **Advanced application-properties**. Append the `atlas.rest.address` property with a comma and the value for the new Atlas Metadata Server: `,http(s):<host_name>:<port_number>`.



Note

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.

4. 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 as this attempts to first stop the newly added Atlas Server, which at this point does not contain any configurations in `/etc/atlas/conf`.

5. On the Ambari dashboard, select **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`

6. Restart the following services that contain Atlas hooks (in order to refresh the configuration files):

- Hive
- Storm
- Falcon
- Sqoop
- Oozie

When you update the Atlas configuration settings in Ambari, Ambari marks the services that require restart, and you can select **Actions > Restart All Required** to restart all services that require a restart.



Important

Apache Oozie requires a restart after an Atlas configuration update, but may not be included in the services marked as requiring restart in Ambari. Select **Oozie > Service Actions > Restart All** to restart Oozie along with the other services.

6. Managing Configurations

Use Ambari Web to manage your HDP component configurations. Select any of the following topics:

- [Configuring Services \[60\]](#)
- [Using Host Config Groups \[60\]](#)
- [Customizing Log Settings \[62\]](#)
- [Downloading Client Configs \[63\]](#)
- [Service Configuration Versions \[63\]](#)

1. Configuring Services

Select a service, then select **Configs** to view and update configuration properties for the selected service. For example, select MapReduce2, then select Configs. Expand a config category to view configurable service properties.

1.1. Updating Service Properties

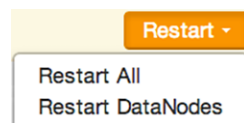
1. Expand a configuration category.
2. Edit values for one or more properties that have the Override option.
Edited values, also called stale configs, show an Undo option.
3. Choose Save.

1.2. Restarting Components

After editing and saving a service configuration, Restart indicates components that you must restart.

Select the Components or Hosts links to view details about components or hosts requiring a restart.

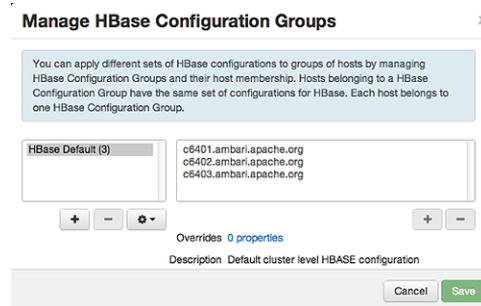
Then, choose an option appearing in Restart. For example, options to restart YARN components include:



2. Using 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

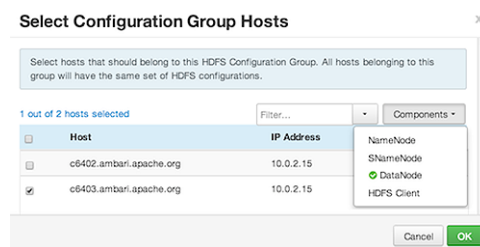
configuration settings, each host belongs to one configuration group that has default configuration settings for the HDFS service. In Configs, select **Manage Config Groups**, to create new groups, re-assign hosts, and override default settings for host components you assign to each group.



To create a Configuration Group:

1. Choose **Add New Configuration Group**.
2. Name and describe the group, then choose **Save**.
3. Select a Config Group, then choose **Add Hosts to Config Group**.
4. Select Components and choose from available Hosts to add hosts to the new group.

Select Configuration Group Hosts enforces host membership in each group, based on installed components for the selected service.



5. Choose **OK**.
6. In Manage Configuration Groups, choose **Save**.

To edit settings for a configuration group:

1. In Configs, choose a Group.
2. Select a Config Group, then expand components to expose settings that allow Override.
3. Provide a non-default value, then choose 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:

- Select an existing configuration group (to which the property value override provided in step 3 will apply), or
- Create a new configuration group (which will include default properties, plus the property override provided in step 3).
- Then, choose OK.

5. In Configs, choose Save.

3. Customizing Log Settings

Ambari Web displays default logging properties in **Service Configs > Custom log 4j Properties**. Log 4j properties control logging activities for the selected service.

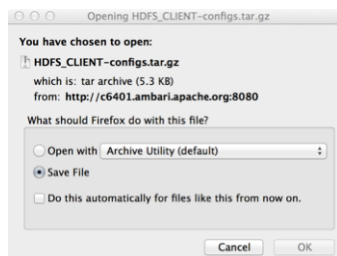
Restarting components in the service pushes the configuration properties displayed in Custom log 4j Properties to each host running components for that service. If you have customized logging properties that define how activities for each service are logged, you will see refresh indicators next to each service name after upgrading to Ambari 1.5.0 or higher. Make sure that logging properties displayed in Custom log 4j Properties include any customization. Optionally, you can create configuration groups that include custom logging

properties. For more information about saving and overriding configuration settings, see [Updating Service Config Properties](#).

4. Downloading Client Configs

For Services that include client components (for example Hadoop Client or Hive Client), you can download the client configuration files associated with that client from Ambari.

- In Ambari Web, browse to the Service with the client for which you want the configurations.
- Choose **Service Actions**.
- Choose **Download Client Configs**. You are prompted for a location to save the client configs bundle.



- Save the bundle.

5. Service Configuration Versions

Ambari provides the ability to manage configurations associated with a Service. You can make changes to configurations, see a history of changes, compare + revert changes and push configuration changes to the cluster hosts.

- [Basic Concepts \[63\]](#)
- [Terminology \[64\]](#)
- [Saving a Change \[64\]](#)
- [Viewing History \[65\]](#)
- [Comparing Versions \[66\]](#)
- [Reverting a Change \[67\]](#)
- [Versioning and Host Config Groups \[67\]](#)

5.1. Basic Concepts

It's important to understand how service configurations are organized and stored in Ambari. Properties are grouped into Configuration Types (config types). A set of config types makes up the set of configurations for a service.

For example, the HDFS Service includes the following config types: hdfs-site, core-site, hdfs-log4j, hadoop-env, hadoop-policy. If you browse to **Services > HDFS > Configs**, the configuration properties for these config types are available for edit.

Versioning of configurations is performed at the service-level. Therefore, when you modify a configuration property in a service, Ambari will create a Service Config Version. The figure below shows V1 and V2 of a Service Configuration Version with a change to a property in Config Type A. After making the property change to Config Type A in V1, V2 is created.



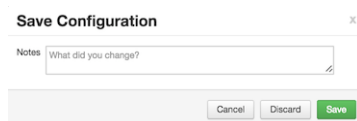
5.2. Terminology

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

Term	Description
Configuration Property	Configuration property managed by Ambari, such as NameNode heapsize or replication factor.
Configuration Type (Config Type)	Group of configuration properties. For example: hdfs-site is a Config Type.
Service Configurations	Set of configuration types for a particular service. For example: hdfs-site and core-site Config Types are part of the HDFS Service Configuration.
Change Notes	Optional notes to save with a service configuration change.
Service Config Version (SCV)	Particular version of configurations for a specific service. Ambari saves a history of service configuration versions.
Host Config Group (HCG)	Set of configuration properties to apply to a specific set of hosts. Each service has a default Host Config Group, and custom config groups can be created on top of the default configuration group to target property overrides to one or more hosts in the cluster. See Managing Configuration Groups for more information.

5.3. Saving a Change

1. Make the configuration property change.
2. Choose Save.
3. You are prompted to enter notes that describe the change.



- Click Save to confirm your change. Cancel will not save but instead returns you to the configuration page to continuing editing.

To revert the changes you made and not save, choose Discard.

To return to the configuration page and continue editing without saving changes, choose Cancel.

5.4. Viewing History

Service Config Version history is available from Ambari Web in two places: On the Dashboard page under the Config History tab; and on each Service page under the Configs tab.

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

Service	Config Group	Created	Author	Notes
V4 HDFS	HDFS Default Current	Thu, Nov 06, 2014 08:10	admin	Created from service config version V2
V3 HDFS	HDFS Default	Thu, Nov 06, 2014 08:08	admin	adjusted NN heapsize
V3 ZooKeeper	deleted	Wed, Nov 05, 2014 19:02	admin	Created from service config version V4
V2 ZooKeeper	ZooKeeper Default Current	Wed, Nov 05, 2014 19:01	admin	Created from service config version V1
V2 ZooKeeper	deleted	Wed, Nov 05, 2014 19:00	admin	No notes
V1 ZooKeeper	deleted	Wed, Nov 05, 2014 19:00	admin	No notes
V2 ZooKeeper	ZooKeeper Default	Wed, Nov 05, 2014 19:00	admin	No notes
V2 ZooKeeper	ZooKeeper Default	Wed, Nov 05, 2014 19:00	admin	sync time
V1 Pig	Pig Default Current	Wed, Nov 05, 2014 17:21	admin	Initial configurations for Pig
V1 Hive	Hive Default Current	Wed, Nov 05, 2014 17:21	admin	Initial configurations for Hive

The most recent configuration changes are shown on the **Service > Configs** tab. Users can navigate the version scrollbar left-right to see earlier versions. This provides a quick way to access the most recent changes to a service configuration.

Summary | Configs | Service Actions

Group: ZooKeeper Default (1) | Manage Config Groups | Filter...

Version scrollbar: V6 (16 hours ago), V5 (16 hours ago), V4 (16 hours ago), V3 (16 hours ago), V2 (16 hours ago), V1 (16 hours ago)

Current version: V4 Current admin authored on Wed, Nov 05, 2014 19:01 | Discard | Save

ZooKeeper Server

- ZooKeeper Server hosts: o6401.ambari.apache.org
- ZooKeeper directory: /hadoop/zookeeper
- Length of single Tick: 2000 ms
- Ticks to allow for sync at Init: 10
- Ticks to allow for sync at Runtime: 5
- Port for running ZK Server: 2181

Advanced zookeeper-env

Advanced zookeeper-logs

Click on any version in the scrollbar to view, and hover to display an option menu which allows you compare versions and perform a revert. Performing a revert 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

5.5. Comparing Versions

When navigating the version scroll area on the **Services > Configs** tab, you can hover over a version to display options to view, compare or revert.

The screenshot shows the Ambari interface for configuring ZooKeeper. At the top, there are tabs for 'Summary' and 'Configs', and a 'Service Actions' dropdown. Below this, the 'Group' is set to 'ZooKeeper Default (1)'. A scrollable list of configuration versions is shown, with 'V6' selected as the 'Current' version. A comparison window is open, showing 'Comparing V6 - V2' with a 'Make V2 Current' button. The comparison details for 'ZooKeeper Server' are visible, showing 'Ticks to allow for sync at Runtime' with values 5 for V6 (Current) and 6 for V2. A 'Filter...' dropdown is also visible at the top right of the version list.

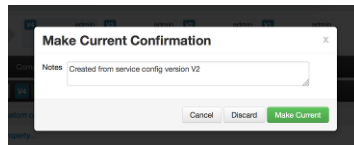
To perform a compare between two service configuration versions:

1. Navigate to a specific configuration version. For example "V6".
2. Using the version scrollbar, find the version you would like to compare against "V6". For example, if you want to compare V6 to V2, find V2 in the scrollbar.
3. Hover over the version to display the option menu. Click "Compare".
4. Ambari displays a comparison of V6 to V2, with an option to revert to V2.
5. Ambari also filters the display by only "Changed properties". This option is available under the Filter control.

The screenshot shows a 'Filter...' dropdown menu with the following options: 'Overridden properties', 'Final properties', 'Changed properties' (which is selected and has a green checkmark), 'Show property issues', and 'Show property warnings'.

5.6. Reverting a Change

You can revert to an older service configuration version by using the “Make Current” feature. The “Make Current” will actually create a new service configuration version with the configuration properties from the version you are reverting – it is effectively a “clone”. After initiating the Make Current operation, you are prompted to enter notes for the new version (i.e. the clone) and save. The notes text will include text about the version being cloned.

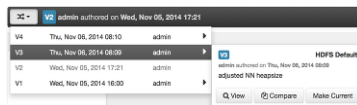


There are multiple methods to revert to a previous configuration version:

- View a specific version and click the **Make V* Current** button.



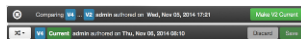
- Use the version navigation dropdown and click the **Make Current** button.



- Hover on a version in the version scrollbar and click the **Make Current** button.



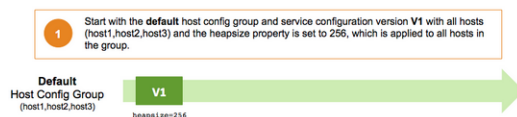
- Perform a comparison and click the **Make V* Current** button.



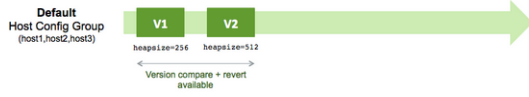
5.7. Versioning and Host Config Groups

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. Same with custom config groups.

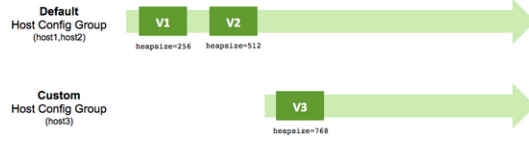
The following example describes a flow where you have multiple host config groups and create service configuration versions in each config group.



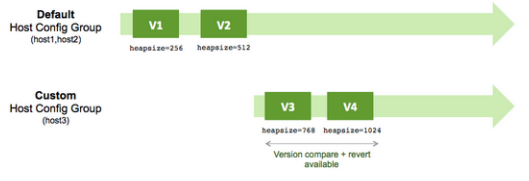
2 Modify the heapsize property to 512 and save the change. Ambari increments the service configuration version to V2. Users can compare and revert between these versions.



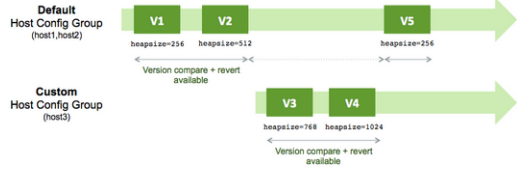
3 Create a new custom host config group and place host3 into the group. For this particular host group, override the heapsize to be 768, creating version V3. The host group (i.e. host3) inherits all properties from default config group and heapsize override from custom group.



4 Modify the custom host config group by adjusting the heapsize to be 1024, creating version V4. Users can compare and revert between these versions.



5 Modify the default host config group by adjusting the heapsize to be 256, creating version V5. Users can compare and revert between these versions inside of their config group.



7. Administering the Cluster

From the cluster dashboard, use the Admin options to view information about [Stack and Versions](#), [Service Accounts](#), and to Enable [Kerberos](#) security.



Note

For more information about administering your Ambari Server, see [Hortonworks Data Platform Apache Ambari Administration](#).

1. Stack and Versions

The **Stack** tab includes information about the Services installed and available in the cluster Stack. Browse the list of Services and click **Add Service** to start the wizard to install Services into your cluster.

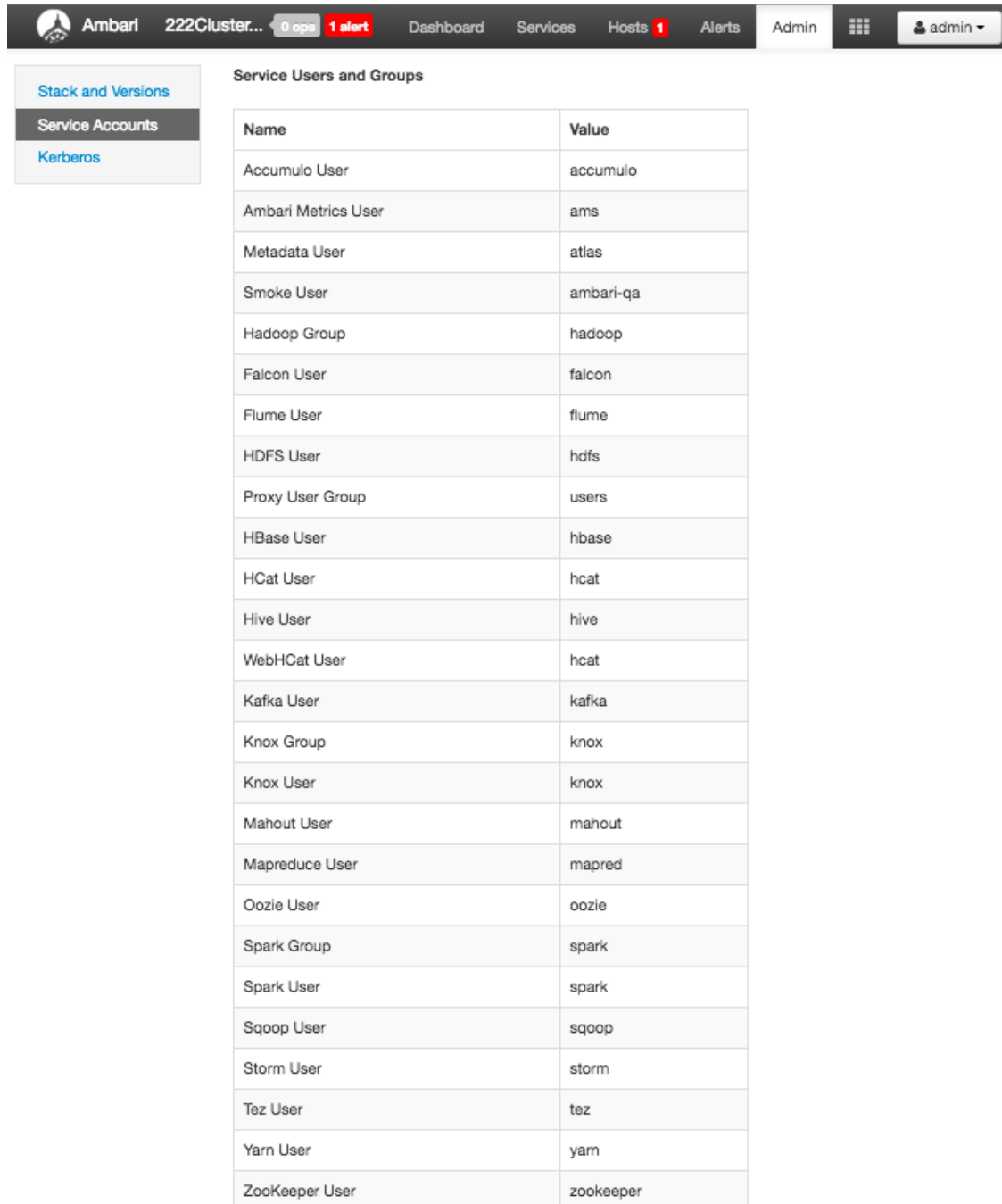
The **Versions** tab includes information about what version of software is currently running and installed in the cluster. This section also exposes the capability to perform an automated cluster upgrade.

For more information on registering and managing versions, see [Hortonworks Data Platform Apache Ambari Administration](#).

For more information on performing a cluster upgrade, see [Hortonworks Data Platform Apache Ambari Upgrade](#).

2. Service Accounts

To view the list of users and groups used by the cluster services, choose **Admin > Service Accounts**.



The screenshot shows the Ambari Admin console interface. At the top, there is a navigation bar with the Ambari logo, cluster name '222Cluster...', and various status indicators like '0 ops' and '1 alert'. Below the navigation bar, there is a sidebar with menu items: 'Stack and Versions', 'Service Accounts', and 'Kerberos'. The main content area is titled 'Service Users and Groups' and contains a table with the following data:

Name	Value
Accumulo User	accumulo
Ambari Metrics User	ams
Metadata User	atlas
Smoke User	ambari-qa
Hadoop Group	hadoop
Falcon User	falcon
Flume User	flume
HDFS User	hdfs
Proxy User Group	users
HBase User	hbase
HCat User	hcat
Hive User	hive
WebHCat User	hcat
Kafka User	kafka
Knox Group	knox
Knox User	knox
Mahout User	mahout
Mapreduce User	mapred
Oozie User	oozie
Spark Group	spark
Spark User	spark
Sqoop User	sqoop
Storm User	storm
Tez User	tez
Yarn User	yarn
ZooKeeper User	zookeeper

3. Kerberos

If Kerberos has not been enabled in your cluster, click the Enable Kerberos button to launch the Kerberos wizard. For more information on configuring Kerberos in your cluster, see [Hortonworks Data Platform Apache Ambari Security](#). Once Kerberos is enabled, you can:

- [How To Regenerate Keytabs \[71\]](#)
- [How To Disable Kerberos \[71\]](#)

3.1. How To Regenerate Keytabs

1. Browse to **Admin > Kerberos**.
2. Click the **Regenerate Kerberos** button.
3. Confirm your selection to proceed.
4. Optionally, you can regenerate keytabs for only those hosts that are missing keytabs. For example, hosts that were not online/available from Ambari when enabling Kerberos.
5. Once you confirm, Ambari will connect to the KDC and regenerate the keytabs for the Service and Ambari principals in the cluster.
6. Once complete, **you must restart all services for the new keytabs to be used.**



Note

Ambari requires the Kerberos Admin credentials in order to regenerate the keytabs. If the credentials are not available to Ambari, you will be prompted to enter the KDC Admin username and password. For more information on configuring Kerberos in your cluster, see [Hortonworks Data Platform Apache Ambari Security](#).

3.2. How To Disable Kerberos

1. Browse to **Admin > Kerberos**.
2. Click the **Disable Kerberos** button.
3. Confirm your selection to proceed. Cluster services will be stopped and the Ambari Kerberos security settings will be reset.
4. To re-enable Kerberos, click Enable Kerberos and follow the wizard steps. For more information on configuring Kerberos in your cluster, see [Hortonworks Data Platform Apache Ambari Security](#).

8. Monitoring and Alerts

Ambari monitors cluster health and can alert you in the case of certain situations to help you identify and troubleshoot problems. You manage how alerts are organized, under which conditions notifications are sent, and by which method. This section provides information on:

- [Managing Alerts \[72\]](#)
- [Configuring Notifications \[75\]](#)
- [List of Predefined Alerts \[79\]](#)

1. Managing 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 watch in the cluster. For example, if your cluster includes HDFS, there is an alert definition to watch the "DataNode Process". An instance of that alert definition is created for each DataNode in the cluster.



Note

Refer to [List of Predefined Alerts](#) for a list of the alerts that are configured by Ambari by default.

Using Ambari Web, you can browse the list of alerts defined for your cluster under the **Alerts** tab. You can search and filter alert definitions by current status, by last status change and the service the alert definition is associated with among other things. Click on the **alert definition name** to view the details about that alert, modify the alert properties (such as check interval and thresholds) and 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, CRITICAL but there are also severities for UNKNOWN and NONE. Alert notifications will be sent on alert status changes (for example, going from OK to CRITICAL). See [Configuring Notifications](#) for more information about notifications.

Modifying an Alert

General properties for an alert include Name, Description and Check Interval. The Check Interval defines the frequency Ambari will check the alert status. For example, a "1 minute" interval means Ambari checks the alert status every "1 minute".

The configuration options for thresholds depend on the **Alert Type**. See [Alert Types](#) for more information.

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 will take effect on all alert instances at the next check interval.

Enabling or Disabling 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.

1.1. Alert Types

Alert thresholds and the threshold units are dependent on alert type. The following table lists the types of alerts, their possible status and if the thresholds are configurable:

Alert Type	Description	Threshold Units
WEB	Connects to a Web URL. Alert status is based on the HTTP response code.	seconds
PORT	Connects to a port. Alert status is based on response time.	seconds
METRIC	Checks the value of a service metric. Units vary, based on the metric being checked.	varies
AGGREGATE	Aggregates the status for another alert.	%
SCRIPT	Executes a script to handle the alert check.	varies
SERVER	Executes a server-side runnable class to handle the alert check.	varies
RECOVERY	Ambari Agents handle the check for process restarts after terminating unexpectedly.	varies

WEB Alert Type

WEB alerts watch a Web URL on a given component and the alert status is determined based on the HTTP response code. Therefore, you cannot change what HTTP response codes determine the thresholds for WEB alerts. You can customize the response text for each thresholds and the overall web connection timeout. A connection timeout is considered a CRITICAL alert. The response code and corresponding status for WEB alerts:

- OK status if Web URL responds with code under 400.
- WARNING status if Web URL responds with code 400 and above.
- CRITICAL status if Ambari cannot connect to Web URL.

PORT Alert Type

PORT alerts check the response time to connect to a given a port and the threshold units are based on seconds.

METRIC Alert Type

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. The thresholds are adjustable and the units for each threshold are metric-dependent. For example, in the case of "CPU utilization" alerts, the unit is "%". And in the case of "RPC latency" alerts, the unit is "milliseconds (ms)".

AGGREGATE Alert Type

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. The threshold units are "%".

SCRIPT Alert Type

SCRIPT alerts execute a script and the script determines status such as OK, WARNING or CRITICAL. You can customize the response text and values for the various properties and thresholds for the SCRIPT alert.

SERVER Alert Type

SERVER alerts execute a server-side runnable class which determines the alert status such as OK, WARNING or CRITICAL.

RECOVERY Alert Type

RECOVERY alerts are handled by the Ambari Agents that are watching for process restarts. The alert status such as OK, WARNING and CRITICAL are based on the number of times a process is being restarted automatically. This is useful to know in cases where processes are terminating and Ambari is automatically restarting.

1.2. Alert Check Counts

Ambari allows you to set the number of alert checks to perform before dispatching a notification. If the alert state changes during a check, Ambari will attempt to check the condition again a number of times (i.e. the "check count") before dispatching a notification. Increase this number if your environment experiences transient issues resulting in false alerts. The alert state change is still recorded in the `/var/log/ambari-server/ambari-alerts.log` as a "SOFT" state change. If after a number of checks the alert condition is still triggered, the state change will be considered "HARD" and notifications will be sent.

You can set the check count value globally for all alerts, or override for individual alerts. Use the override for individual alerts in case you have a specific alert (or alerts) that are experiencing transient issues.

Modifying the Global Alert Check Count

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

4. Click **Save**.

Modifying the Alert Check Count for Individual Alerts

1. Browse to the Alerts section in Ambari Web.
2. Select the alert that 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**.



Note

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



Note

Changes made using the Ambari UI to the Global alert check count may require a few seconds to appear in the Ambari UI for individual alerts.

2. Configuring Notifications

With Alert Groups and Notifications, you can create groups of alerts and setup notification targets for each group. This way, you can notify different parties interested in certain sets of alerts via different methods. For example, you might want your Hadoop Operations team to receive all alerts via EMAIL, regardless of status. And at the same time, have your System Administration team receive all RPC and CPU related alerts that are Critical only via SNMP. To achieve this scenario, you would have an Alert Notification that handles Email for all alert groups for all severity levels, and you would have a different Alert Notification group that handles SNMP on critical severity for an Alert Group that contains the RPC and CPU alerts.

Ambari defines a set of default Alert Groups for each service installed in the cluster. For example, you will see a group for HDFS Default. These groups cannot be deleted and the alerts in these groups are not modifiable. If you choose not to use these groups, just do not set a notification target for them.

Creating or Editing Notifications

1. Browse to the Alerts section in Ambari Web.
2. Under the **Actions** menu, click **Manage Notifications**.
3. The list of existing notifications is shown.
4. Click + to "Create new Alert Notification". The Create Alert Notification dialog is displayed.
5. Enter the notification name, select the groups to which the notification should be assigned (all or a specific set), select the Severity levels that this notification responds to, include a description, and choose the method for notification (EMAIL or SNMP).

- For **EMAIL**: provide information about your SMTP infrastructure such as SMTP Server, Port, To/From address and if authentication is required to relay messages through the server. You can add custom properties to the SMTP configuration based on the [Javamail SMTP options](#).

Parameter	Description
Email To	A comma-separated list of one or more email addresses 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 "from" alert email.
Use Authentication	Check if your SMTP Server requires authentication in order to relay messages. Be sure to also provide the username and password credentials.

- For **SNMP**: select the SNMP version, Community, Host, and Port where the SNMP trap should be sent. Also, the OID parameter must be configured properly for SNMP trap context. If no custom, or enterprise-specific OID will be used, we recommend the following:

Parameter	Description
OID	1.3.6.1.4.1.18060.16.1.1
Hosts	A comma-separated list of one or more Host FQDNs of where to send the trap.
Port	The port where snmptrapd is listening on the Hosts.

6. After completing the notification, click **Save**.

Creating or Editing Alert Groups

1. Browse to the Alerts section in Ambari Web.
2. From the Actions menu, choose Manage Alert Groups
3. The list of existing groups (default and custom) is shown.
4. Choose + to "Create Alert Group". Enter the Group a 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.

Dispatching Notifications

When an alert is enabled and the alert status changes (for example, from OK to CRITICAL or CRITICAL to OK), Ambari will send a notification (depending on how the user has configured notifications).

For **EMAIL** notifications: Ambari will send an email digest that includes all alert status changes. For example: if two alerts go CRITICAL, Ambari sends one email that says "Alert A is CRITICAL and Ambari B alert is CRITICAL". Ambari will not send another email notification until status has changed again.

For **SNMP** notifications: Ambari will fire an SNMP trap per alert status change. For example: if two alerts go CRITICAL, Ambari will fire two SNMP traps, one for each alert going OK -> CRITICAL. When the alert changes status from CRITICAL -> OK, another trap is sent.

Viewing Alert Status Log

In addition to dispatching alert notifications, Ambari writes alert status changes to a log on the Ambari Server host. Alert status changes will be written to the log regardless if EMAIL or SNMP notifications are configured.

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 will include the time of the status change, the alert status, the alert definition name and the response text. For example:

```
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
```

2.1. Customizing Notification Templates

The notification template content produced by Ambari is tightly coupled to a notification type. Email and SNMP both have customizable templates that can be used 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 bundled inside of Ambari Server JAR. 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 so the template is not exposed on the disk. But we can use that file as a reference example.

When you customize the alert template, you are effectively overriding the template bundled by default. To override the alert templates XML:

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. Any notification types defined in the new template will 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.

```
alerts.template.file=/foo/var/alert-templates-custom.xml
```

4. Save the file and restart Ambari Server.



Important

Some alert notification types, such as EMAIL, automatically combine all pending alerts into a single outbound notification ("digest"). Others, like

SNMP, never combine pending alerts and will always create a 1:1 notification for every alert in the system ("individual"). All alert notification types are specified in the same alert templates file, but the specific alert template for each notification type will most likely vary greatly.

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:

Variable	Description
<code>\$alert.getAlertDefinition()</code>	The definition that the alert is an instance of.
<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 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.

Variable	Description
<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.

3. List of Predefined Alerts

- [HDFS Service Alerts \[80\]](#)
- [HDFS HA Alerts \[83\]](#)
- [NameNode HA Alerts \[84\]](#)
- [YARN Alerts \[84\]](#)
- [MapReduce2 Alerts \[85\]](#)
- [HBase Service Alerts \[86\]](#)
- [Hive Alerts \[87\]](#)
- [Oozie Alerts \[87\]](#)
- [ZooKeeper Alerts \[87\]](#)

- [Ambari Alerts \[88\]](#)
- [Ambari Metrics Alerts \[88\]](#)

3.1. 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.	Some DataNodes are down and the replicas that are missing blocks are only on those DataNodes. The corrupt/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.	For critical data, use a replication factor of 3. Bring up the failed DataNodes with missing or corrupt blocks. Identify the files associated with the missing or corrupt blocks by running the Hadoop fsck command. Delete the corrupt files and recover them from backup, if it exists.
NFS Gateway Process	PORT	This host-level alert is triggered if the NFS Gateway process cannot be confirmed to be up and listening on the network.	NFS Gateway is down.	Check for dead 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.	Cluster storage is full. If cluster storage is not full, DataNode is full.	If cluster still has storage, use Balancer to distribute the data to relatively less-used datanodes. 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 Balancer.
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, given in seconds.	DataNode process is down or not responding. DataNode are not down but is not listening to the correct network port/address.	Check for dead DataNodes in Ambari Web. Check for any errors in the DataNode logs (/var/log/hadoop/hdfs) and restart the DataNode, if necessary. Run the netstat-tuplpn 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 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 top command to determine which processes are consuming excess CPU. Reset the offending process.

Alert	Alert Type	Description	Potential Causes	Possible Remedies
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 cluster still has storage, use Balancer to distribute the data to relatively less used DataNodes. 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 Balancer.
Percent DataNodes Available	AGGREGATE	This alert is triggered if the number of down 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 dead DataNodes in Ambari Web. Check for any errors in the DataNode logs (/var/log/hadoop/hdfs) and restart the DataNode hosts/processes. Run the netstat-tuplpn 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 of uncommitted transactions is beyond a certain threshold.	Too much time elapsed since last NameNode checkpoint. Uncommitted transactions beyond threshold.	Set NameNode checkpoint. Review threshold for uncommitted transactions.
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.

Alert	Alert Type	Description	Potential Causes	Possible Remedies
				After adding more storage, run 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 dead 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 in percent.		
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 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 datanode port has grown beyond the specified threshold within a given period. This alert will monitor Hourly and Daily periods.		

Alert	Alert Type	Description	Potential Causes	Possible Remedies
NameNode Service RPC Processing Latency	SCRIPT	This service-level alert is triggered if the deviation of RPC latency on 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.		

3.2. 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 dead.
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. On each host running NameNode, run the netstat-tupln 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.

3.3. 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 dead.
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. On each host running NameNode, run the netstat-tupln 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.

3.4. 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 dead 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 dead NodeManagers. Check for any errors in the NodeManager logs (/var/log/hadoop/yarn) and restart the NodeManagers hosts/processes, as necessary. Run the netstat-tupln command to check if the NodeManager process is bound to the correct network port.

Alert	Alert Type	Description	Potential Causes	Possible Remedies
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 top 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 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.

3.5. MapReduce2 Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
HistoryServer 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.
HistoryServer RPC latency	METRIC	This host-level alert is triggered if the HistoryServer operations RPC latency exceeds the configured critical threshold. Typically an increase	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.

Alert	Alert Type	Description	Potential Causes	Possible Remedies
		in the RPC processing time increases the RPC queue length, causing the average queue wait time to increase for NameNode operations.		
HistoryServer 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 top command to determine which processes are consuming excess CPU. Reset the offending process.
HistoryServer 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.

3.6. 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 down because there were problems in the dependent services, ZooKeeper or HDFS. GC paused the RegionServer for too long and the RegionServers lost contact with Zookeeper.	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. 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.	The HBase master process is down. The HBase master has shut itself down because there were problems in the dependent services, ZooKeeper or HDFS.	Check the dependent services. Look at the master log files (usually /var/log/hbase/*.log) for further information. Look at the configuration files (/etc/hbase/conf). Restart the master.
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.	Use the top command to determine which processes are consuming excess CPU Reset the offending process.
RegionServers Health Summary	This service-level alert is triggered if there are unhealthy RegionServers.	The RegionServer process is down on the host. The RegionServer process is up and running but not listening on the correct network port (default 60030).	Check for dead RegionServer in Ambari Web.

Alert	Description	Potential Causes	Possible Remedies
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.	The RegionServer process is down on the host. The RegionServer process is up and running but not listening on the correct network port (default 60030).	Check for any errors in the logs (/var/log/hbase/) and restart the RegionServer process using Ambari Web. Run the netstat-tuplpn command to check if the RegionServer process is bound to the correct network port.

3.7. 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.	HiveServer2 process is not running. HiveServer2 process is not responding.	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.	The Hive Metastore service is down. The database used by the Hive Metastore is down. The Hive Metastore host is not reachable over the network.	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.	The WebHCat server is down. The WebHCat server is hung and not responding. The WebHCat server is not reachable over the network.	Restart the WebHCat server using Ambari Web.

3.8. 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 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.

3.9. 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.	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.

Alert	Alert Type	Description	Potential Causes	Possible Remedies
		It aggregates the results of ZooKeeper process checks.		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 netstat-tupln command to check if the ZooKeeper server process is bound to the correct network port.

3.10. Ambari Alerts

Alert	Alert Type	Description	Potential Causes	Possible Remedies
Host Disk Usage	SCRIPT	This host-level alert is triggered if the amount of disk space used on a host goes above specific thresholds (50% warn, 80% crit).	The amount of free disk space left is low.	Check host for disk space to free or add more storage.
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.

3.11. 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	The Metrics Collector process is not running.	Check the Metrics Collector is running.

Alert	Description	Potential Causes	Possible Remedies
	configured critical threshold, given in seconds.		
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.	Review Tuning Ambari Metrics for information on tuning 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-Start 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.	Review Tuning Ambari Metrics for information on tuning 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 a given host.
Percent Metrics Monitors Available	This is an AGGREGATE alert of the Metrics Monitor Status.	Metrics Monitors are down.	Check the Metrics Monitors.

9. Using Ambari Core Services

This chapter describes how to use and configure the following Ambari Core Services:

- [Ambari Metrics \[90\]](#)
- [Ambari Log Search \(Technical Preview\) \[128\]](#)
- [Ambari Infra \[133\]](#)

1. Ambari Metrics

The **Ambari Metrics System (AMS)** is a system for collecting, aggregating and serving Hadoop and system metrics in Ambari-managed clusters.

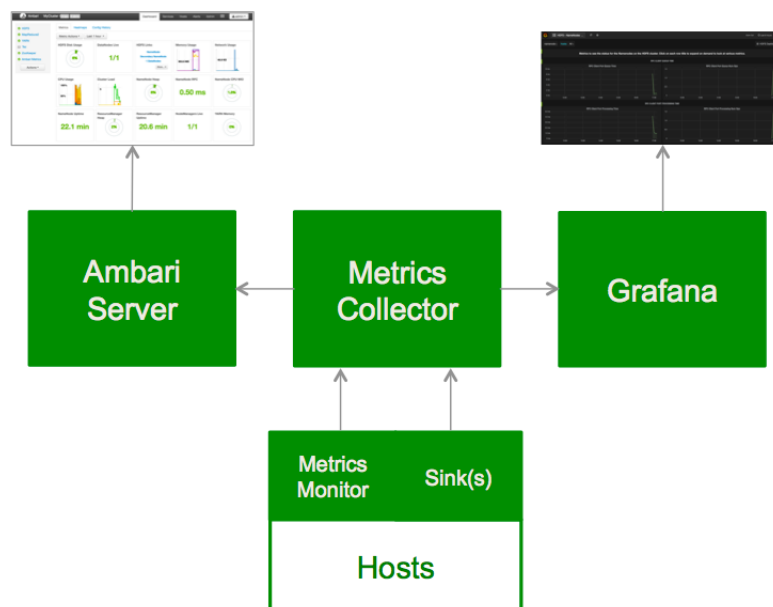
- [AMS Architecture \[90\]](#)
- [Using Grafana \[91\]](#)
- [Performance Tuning \[123\]](#)
- [Moving the Metrics Collector \[127\]](#)
- [\(Optional\) Enabling Individual Region, Table, and User Metrics for HBase \[127\]](#)

1.1. AMS Architecture

AMS has three primary components: **Metrics Monitors**, **Hadoop Sinks**, **Metrics Collector**, and **Grafana**.

- The **Metrics Monitors** are installed and run on each host in the cluster to collect system-level metrics and publish to the **Metrics Collector**.
- The **Hadoop Sinks** plug into the various 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 **Sinks**.
- The **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 diagram provides a high-level illustration of how the components of AMS work together to collect metrics and make those metrics available to Ambari.



1.2. Using Grafana

The **Ambari Metrics System** includes Grafana with pre-built dashboards for advanced visualization of cluster metrics.

- [Accessing Grafana \[92\]](#)
- [Viewing Grafana Dashboards \[93\]](#)
- [Viewing Selected Metrics on Grafana Dashboards \[93\]](#)
- [Viewing Metrics for Selected Hosts \[94\]](#)
- [HDFS Dashboards \[95\]](#)
- [YARN Dashboards \[97\]](#)
- [Hive Dashboards \[100\]](#)
- [Hive LLAP Dashboards \[102\]](#)
- [HBase Dashboards \[106\]](#)
- [Kafka Dashboards \[115\]](#)
- [Storm Dashboards \[117\]](#)
- [System Dashboards \[118\]](#)
- [NiFi Dashboard \[121\]](#)
- [Changing the Grafana Admin Password \[122\]](#)

- [Set Up HTTPS for Grafana \[122\]](#)



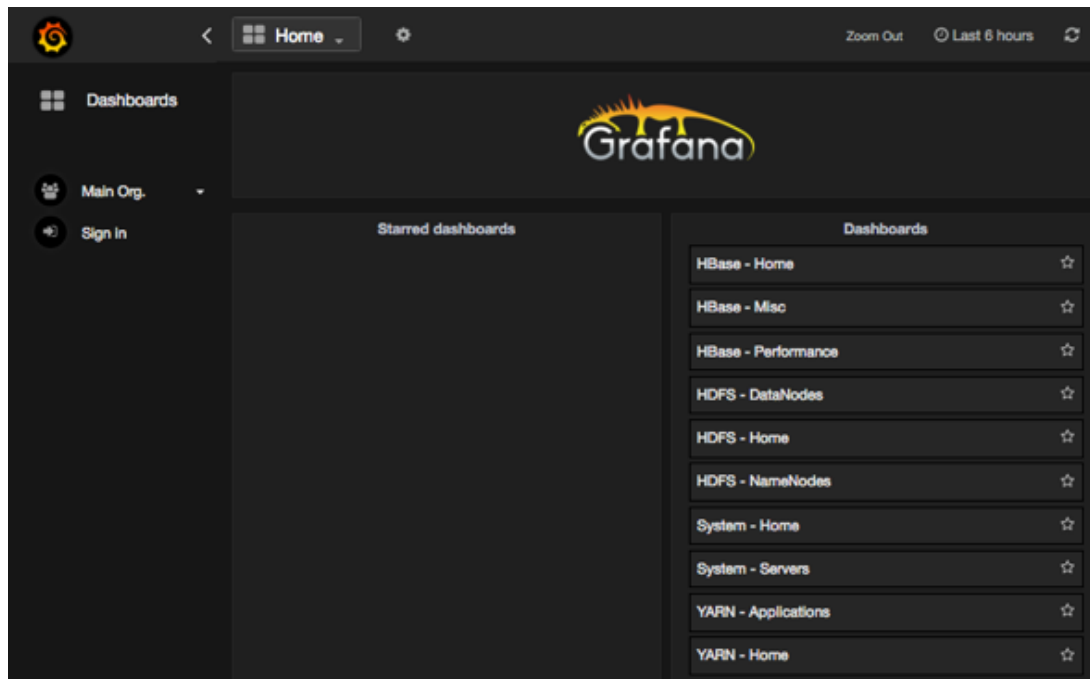
Note

You can visit <http://grafana.org/> to learn more about Grafana.

1.2.1. Accessing Grafana

You can access the Grafana UI as part of the Ambari Metrics System.

1. In Ambari Web, browse to **Services > Ambari Metrics > Summary**.
2. Select **Quick Links** and then choose **Grafana**.
3. A new tab will open in your browser with the Grafana UI.



Note

You have read-only access to Grafana. You will not be able to modify the pre-built dashboards or add new dashboards.

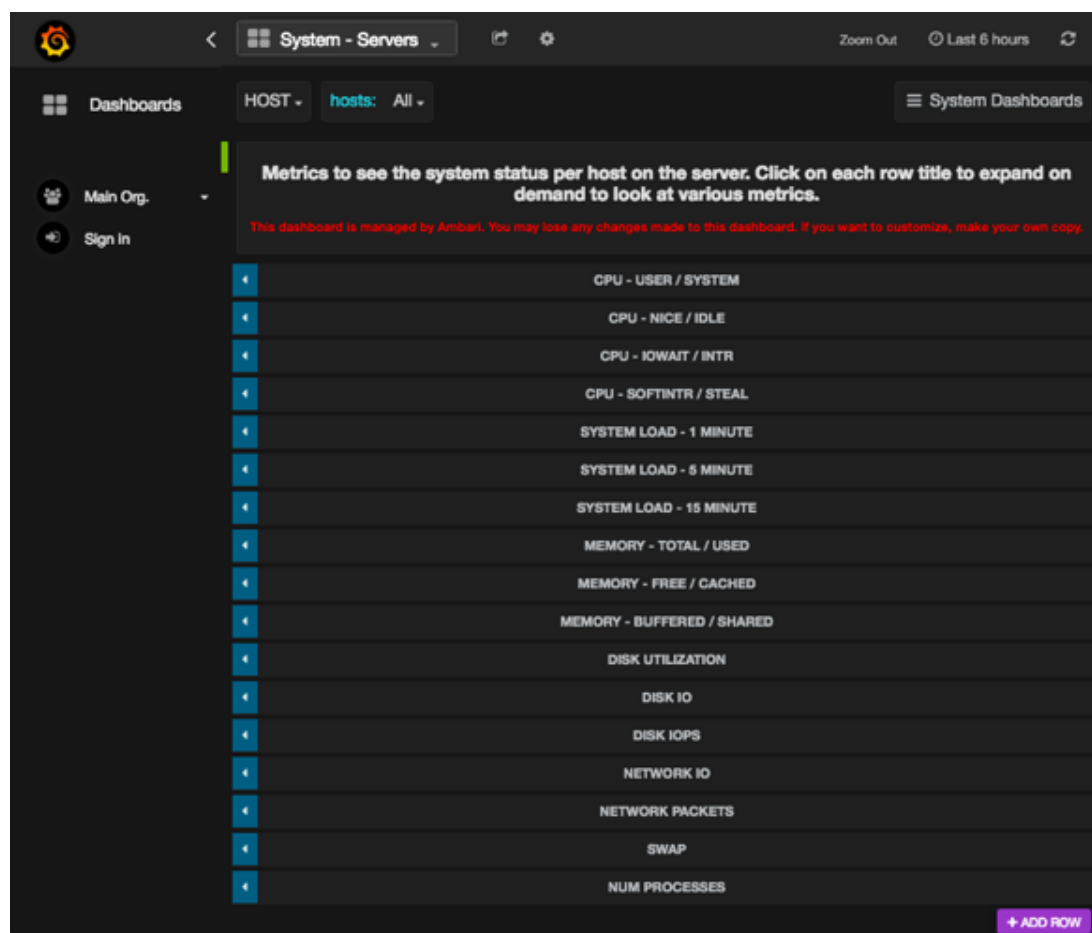
Dashboard	Description
HDFS	Provides information about HDFS performance and file activity. See HDFS Dashboards for more information.
YARN	Provides information about YARN application and queue activity. See YARN Dashboards for more information.
HBase	Provides information about HBase performance and file activity. See HBase Dashboards for more information.
System	Provides information about overall system performance and file activity. See System Dashboards for more information.

Dashboard	Description
NiFi	Provides information about NiFi JVM, thread, FlowFile, and byte activity. See NiFi Dashboard for more information.

1.2.2. Viewing Grafana Dashboards

To view a dashboard that displays multiple metrics for an HDP component, select a dashboard name in the Grafana UI. For example:

1. In Grafana, browse to **Dashboards**.
2. Select **System - Servers**.
3. The **System - Servers** dashboard opens in your browser.



1.2.3. Viewing Selected Metrics on Grafana Dashboards

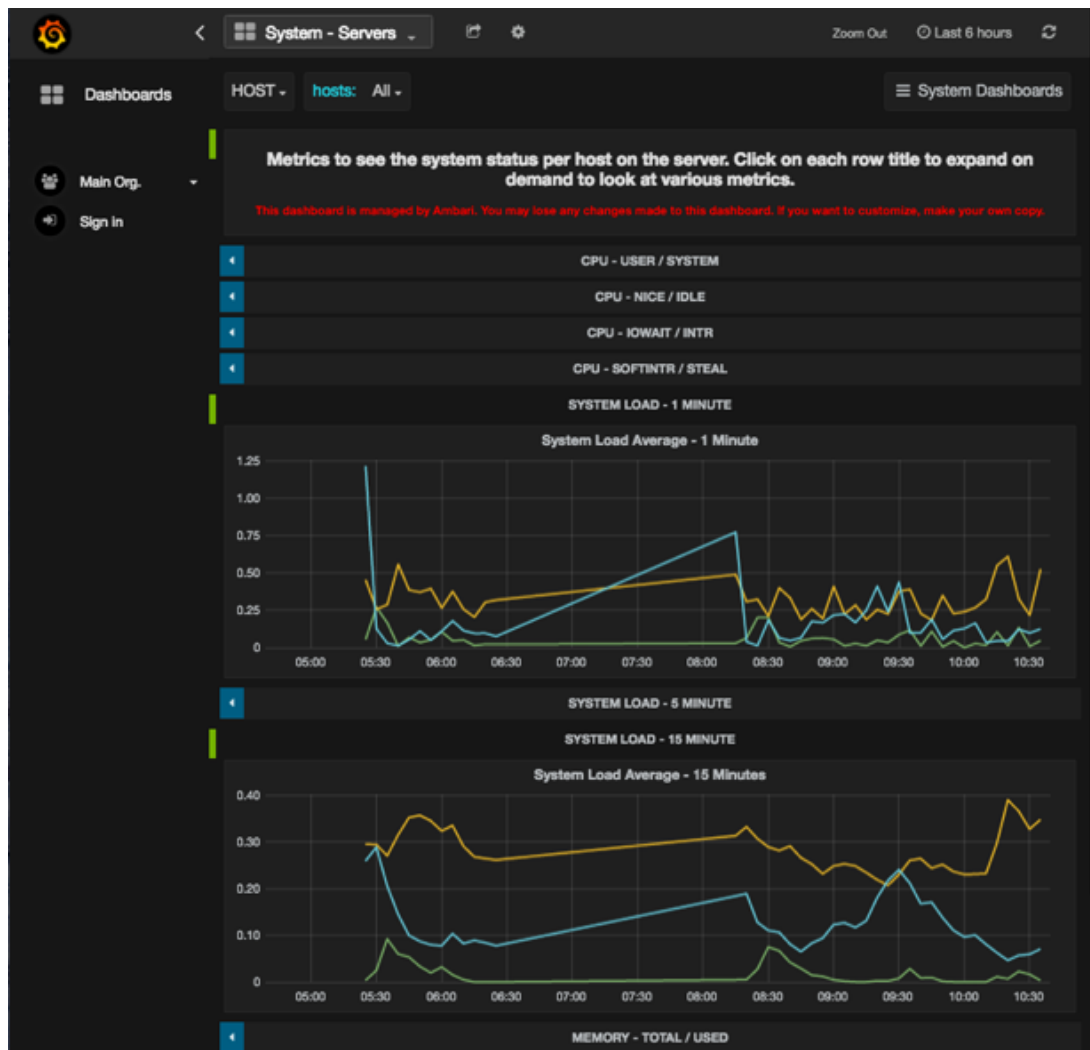
To view one or more metrics in a selected dashboard, expand one or more rows in the dashboard. For example:

1. In the System - Servers dashboard, click a row name.

For example, click System Load Average - 1 Minute.

2. The row expands to display a chart that shows metrics information.

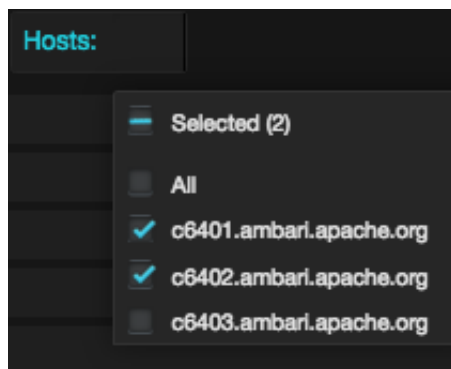
The following example shows the System Load Average - 1 Minute and the System Load Average - 15 Minute rows expanded.



1.2.4. Viewing Metrics for Selected Hosts

Grafana shows metrics for all hosts in your cluster by default. You can limit the displayed metrics by selecting one or more host(s) in your cluster, using the Hosts menu.

1. Expand **Hosts**.
2. Select one or more host names.
3. A checkmark appears next to selected host names, as shown in the following example:



Note

Selections in Host menu apply to all metrics in the current dashboard. Grafana refreshes the current dashboards when you select a new set of hosts.

1.2.5. HDFS Dashboards

The following Grafana dashboards are available for HDFS:

- [HDFS - Home \[95\]](#)
- [HDFS - NameNodes \[96\]](#)
- [HDFS - DataNodes \[97\]](#)
- [HDFS - Users \[97\]](#)

1.2.5.1. HDFS - Home

Metrics that show operating status for HDFS.



Note

In NameNode HA setup, this metric, and all the following metrics, include metrics collected from both the active and 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.

Row	Metrics	Description
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).

1.2.5.2. 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.
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.

Row	Metrics	Description
	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.

1.2.5.3. 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.
	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.

1.2.5.4. 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.

1.2.6. YARN Dashboards

The following Grafana dashboards are available for YARN:

- [YARN - Home \[98\]](#)
- [YARN - Applications \[98\]](#)
- [YARN - MR JobHistory Server \[98\]](#)
- [YARN - MR JobHistory Server \[98\]](#)
- [YARN - NodeManagers \[98\]](#)

- [YARN - Queues \[99\]](#)
- [YARN - ResourceManager \[100\]](#)

1.2.6.1. 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.

1.2.6.2. 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).
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.

1.2.6.3. 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.

1.2.6.4. 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.

Row	Metrics	Description
	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.
LOG4J METRICS	LOG ERROR	Number of ERROR logs.
	LOG FATAL	Number of FATAL logs.

1.2.6.5. 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.

Row	Metrics	Description
	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.

1.2.6.6. 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.
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.

1.2.6.7. 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.

1.2.7. Hive Dashboards

The following Grafana dashboards are available for Hive:

- [Hive - Home \[101\]](#)
- [Hive - HiveMetaStore \[101\]](#)
- [Hive - HiveServer2 \[102\]](#)

1.2.7.1. 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.
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.

1.2.7.2. 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.

1.2.7.3. 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 roundtrip (get_all_databases)	Time taken to process a low-cost call made by health checks to the metastore embedded in this HiveServer2. Data for this metric may not be available if HiveServer2 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. Data 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.

1.2.8. Hive LLAP Dashboards

The following Grafana dashboards are available for Hive LLAP. The LLAP Heat map dashboard and the LLAP Overview dashboard let you quickly see where the hotspots are among the LLAP daemons. If you find an issue and want to drill down into a specific system, use the LLAP Daemon dashboard.

Note that for all Hive LLAP dashboards, the 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 \[102\]](#)
- [Hive LLAP - Overview \[103\]](#)
- [Hive LLAP - Daemon \[105\]](#)

1.2.8.1. Hive LLAP - Heatmap

The heat map shows all the nodes that are running LLAP daemons with percentage summary for available executors and cache. This dashboard lets you identify the hotspots in the cluster in terms of executors and cache.

NOTE: The values in the table are color coded based on threshold: if the threshold is more than 50%, it is Green; 20-50%, it is Yellow; less than 20%, it 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.

Row	Metrics	Description
	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.

1.2.8.2. Hive LLAP - Overview

Shows the aggregated information across all of the clusters. For example, the total cache memory from all the nodes. This dashboard allows you to see that your cluster is configured and running correctly. For example, you might have configured 10 nodes but only see executors and cache accounted for 8 nodes running.

If you find an issue in this dashboard, 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

Row	Metrics	Description
		<p>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</p> <p>Evicted: Lower priority requests are evicted if the slots are filled by higher priority requests</p>
	Total Execution Slots	<p>Shows the total execution slots, the number of free or available slots, and number of slots occupied in the wait queue across all nodes.</p> <p>Ideally, the threads available (blue) result should be the same as the threads that are occupied in the queue result.</p>
	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)	<p>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.</p> <p>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.</p>
	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	<p>Shows the average amount of Java Virtual Machine (JVM) heap memory used across all nodes.</p> <p>If the heap usage keeps increasing, you might run out of memory and the task failure count would also increase.</p>
	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.

Row	Metrics	Description
		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. 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.

1.2.8.3. Hive LLAP - Daemon

Metrics that show operating status for Hive LLAP Daemons.

Row	Metrics	Description
Executor Metrics	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.
	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-	Shows the 95th percentile latencies for time taken to kill tasks due to pre-emption in 300 second intervals.

Row	Metrics	Description
	empted Task (300s interval)	
	Max Time To Kill Task Pre-empted Task	Shows the maximum time taken to kill a task due to pre-emption.
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).

1.2.9. 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 \[106\]](#)
- [HBase - RegionServers \[107\]](#)
- [HBase - Misc \[112\]](#)
- [HBase - Tables \[114\]](#)
- [HBase - Users \[115\]](#)



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.

1.2.9.1. 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.

Row	Metrics	Description
	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.
REGIONSERVER RPC	RegionServer RPC Throughput - Average	Average sent and received bytes from the RPC across all RegionServers.

1.2.9.2. 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.
	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.

Row	Metrics	Description
	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
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.

Row	Metrics	Description
	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.
	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.

Row	Metrics	Description
	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.
	WAL - Append Latencies - Median	Median latency for Write-Ahead-Log append operation to the filesystem in the RegionServer.
	WAL - Append Latencies - 75th Percentile	75th 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.

Row	Metrics	Description
	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.
LOCALITY	Percent Files Local	Percentage of files served from the local DataNode for the RegionServer.

1.2.9.3. HBase - Misc

The HBase - Misc dashboards display miscellaneous metrics related to the HBase cluster. A couple of use cases for these metrics is to debug authentication and authorization issues and exceptions thrown from 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.
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.

1.2.9.4. HBase - Tables

These metrics reflect data on the table level. The HBase - Tables dashboards and data help you compare load distribution and resource usage among tables in a cluster at different points of time.

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).

Row	Metrics	Description
	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).
	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).

1.2.9.5. HBase - Users

The HBase - Users dashboards display metrics and detailed data on a per-user basis across the cluster. Click the second drop-down arrow in the upper-left corner to pick and choose a single user, a group of users, or all users. You can change your user selection on the drop-down menu 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).

1.2.10. Kafka Dashboards

The following Grafana dashboards are available for Kafka:

- [Kafka - Home \[115\]](#)
- [Kafka - Hosts \[116\]](#)
- [Kafka - Topics \[117\]](#)

1.2.10.1. 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.

Row	Metrics	Description
CONTROLLER/LEADER COUNT & REPLICAS 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.
	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.
REPLICAS FETCHER MANAGER	ReplicaFetcherManager MaxLag	The maximum lag in messages between the follower and leader replicas.

1.2.10.2. Kafka - Hosts

Metrics that show operating status for Kafka cluster on a per broker level.

Use the drop-downs to select:

- a Kafka broker
- a host
- whether to view largest (top) or smallest (bottom) values
- the number of values that you want to view
- an 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.
REPLICAS 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.

Row	Metrics	Description
	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.

1.2.10.3. 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.

1.2.11. Storm Dashboards

The following Grafana dashboards are available for Storm:

- [Storm - Home \[117\]](#)
- [Storm - Topology \[117\]](#)
- [Storm - Components \[118\]](#)

1.2.11.1. 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.

1.2.11.2. 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.
	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.

1.2.11.3. 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.

1.2.12. System Dashboards

The following Grafana dashboards are available for System:

- [System - Home \[119\]](#)

- [System - Servers \[120\]](#)

1.2.12.1. 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.
	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 UTILITZATION	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 $SUM(Avg \text{ value from each host} + \text{Interpolated value for 2 missing hosts})/100$.

- Sum/Total implies the sum of all values in a timeslice (30 seconds) from all hosts in the cluster. The same interpolation rule applies.

1.2.12.2. 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.
	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.

Row	Metrics	Description
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.

1.2.13. NiFi Dashboard

The following Grafana dashboard is available for NiFi:

- [NiFi-Home \[121\]](#)

1.2.13.1. 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.

Row	Metrics	Description
	Bytes Queued	Displays the number of bytes of FlowFile data queued in a NiFi processor connection.

1.2.14. Changing the Grafana Admin Password

When Ambari initially installed the Ambari Metrics service, you set the default Grafana Admin password.

If you need to change this password after install, you have to change the password directly in Grafana, and then make the change in the Ambari Metrics configuration to match the new password to use. Use these instructions to perform the password change.

1. In **Ambari Web**, browse to **Services > Ambari Metrics** and select the **Quick Links** and then choose **Grafana**. This will open the Grafana UI in read-only mode.
2. Once in Grafana, click **Sign In** in the left column.
3. Log in as the admin user, using the current password.
4. Click on the admin label in the left column to view the admin Profile. Click on Change password in the top column.
5. Enter the old (current) password and enter (and confirm) the new password. Click the Change Password button.
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.

1.2.15. Set Up HTTPS for Grafana

If you want to limit access to the Grafana to HTTPS connections, you need to provide a certificate. While it is possible to use a self-signed certificate for initial trials, they are not suitable for production environments. After your certificate is in place, you must run a special setup command.

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, 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 ownership and permissions the certification and key files so they will be 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
```

5. In **Ambari Web**, browse to **Services > Ambari Metrics > Configs**.

6. Update the following properties:

Section	Property	Value
Advanced ams-grafana-ini	protocol	https
Advanced ams-grafana-ini	cert_file	/etc/ambari-metrics-grafana/conf/ams-grafana.crt
Advanced ams-grafana-ini	cert-Key	/etc/ambari-metrics-grafana/conf/ams-grafana.key

7. Save the configuration and restart the services as prompted.

1.3. Performance Tuning

To get optimal performance from the Ambari Metrics System, review the following Metrics Collector configuration options.

- [Metrics Collector Modes \[123\]](#)
- [Aggregated Metrics TTL Settings \[124\]](#)
- [Memory Settings \[125\]](#)
- [General Guidelines \[126\]](#)

1.3.1. Metrics Collector Modes

The **Metrics Collector** is built using Hadoop technologies such as HBase, Phoenix, and ATS. The Collector can store metrics data on the local filesystem, 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 will capture and write metrics to the local file system on the host where the Collector is running. As well, all the Collector runs in a single process on that host.



Important

When running in **embedded mode**, you should confirm the "hbase.rootdir" and "hbase.tmp.dir" 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.
```

Refer to [General Guidelines](#) for more information on Disk Space recommendations.

Another critical factor in embedded mode is the TTL settings, which manage how much data will be stored. Refer to [Aggregated Metrics TTL](#) for more information on these settings.

When the Collector is configured for **distributed** mode, the Collector writes metrics to HDFS and the components will run in distributed processes. This mode helps manage CPU and memory consumption.

To switch the **Metrics Collector** from **embedded** mode to **distributed** mode, in **Ambari Web**, browse to **Services > Ambari Metrics > Configs**, make the following changes, then restart the Metrics Collector.

Configuration Section	Property	Description	Value
General	Metrics Service operation mode (<code>timeline.metrics.service.operation.mode</code>)	Designates whether to run in distributed or embedded mode.	distributed
Advanced ams-hbase-site	<code>hbase.cluster.distributed</code>	Indicates AMS will run in distributed mode.	true
Advanced ams-hbase-site	<code>hbase.rootdir</code> see <i>note 1</i>	The HDFS directory location where metrics will be stored.	<code>hdfs://\$NAMENODE_FQDN:8020/apps/ams/metrics</code>

Note 1: If your cluster is configured for a highly-available NameNode, set the `hbase.rootdir` value to use the HDFS nameservice, instead of the NameNode hostname:

```
hdfs://hdfsnameservice/apps/ams/metrics
```

Optionally, existing data can be migrated from the local store to HDFS prior to switching to distributed mode.

1. Create HDFS directory for ams user. For example:

```
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. Perform the configuration changes above to switch to distributed mode.

5. Start the Metrics Collector.

Note about HBase Cluster Metrics: 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 cluster performance on the individual region, table, or user level.

1.3.2. Aggregated Metrics TTL Settings

AMS provides configurable Time To Live configuration for aggregated metrics. The TTL settings are available in Ambari Metrics > Configs > Advanced ams-site and have the

".ttl" suffix. Each property name is self explanatory and controls the amount of time to keep metrics at the specified aggregation level before they are purged. The values for these TTL's are set in seconds. In an example where you are running a single-node sandbox and want to ensure that no values are stored for more than 7 days to save on local disk space, you would set any property ending in ".ttl" that has a value greater than 604800, 7 days in seconds, to 604800. That would ensure that properties such as `timeline.metrics.cluster.aggregator.daily.ttl` that controls the daily aggregation TTL, which by default stores data for 2 years, will only store daily aggregations for 604800 seconds, or 7 days. Reducing the TTL values helps significantly reduce the total amount of storage used for metric storage. Those that matter most for reducing the total amount of disk space used for AMS are:

- `timeline.metrics.cluster.aggregator.minute.ttl` - Controls minute level aggregated metrics TTL
- `timeline.metrics.host.aggregator.ttl` - Controls host-based precision metrics TTL

It's important to note that these settings should be set during installation. If these settings need to be changed post-installation, they have to be set using the HBase shell. This is a current limitation imposed by Phoenix that is resolved in Ambari 2.1.2. To change these TTL settings in Ambari 2.1.1 and earlier, the HBase shell is used to connect to the embedded HBase instance that is part of AMS. Run the following command from the Collector host.

```
/usr/lib/ams-hbase/bin/hbase --config /etc/ams-hbase/conf shell
```

Once connected to HBase each of the following tables needs to be updated with the appropriate TTL values that are being changed. The table below maps a specific ".ttl" property in **Ambari Metrics > Configs > Advanced `ams-site`** to the actual HBase table.

Property	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>

For each table that needs to be updated, alter the TTL value as follows:

```
hbase(main):000:0> alter 'METRIC_RECORD_DAILY', { NAME => '0', TTL => 604800}
```

1.3.3. Memory Settings

Since AMS uses multiple components (such as HBase and Phoenix) for metrics storage and query, there are multiple tunable properties for tuning memory use. The following table lists each memory configuration.

Configuration	Property	Description
Advanced <code>ams-env</code>	<code>metrics_collector_heapsize</code>	Heap size configuration for the Collector.

Configuration	Property	Description
Advanced ams-hbase-env	hbase_regionserver_heapsize	Heap size configuration for the single AMS HBase Region Server.
Advanced ams-hbase-env	hbase_master_heapsize	Heap size configuration for the single AMS HBase Master.
Advanced ams-hbase-env	regionserver_xmn_size	Maximum value for the young generation heap size for the single AMS HBase RegionServer.
Advanced ams-hbase-env	hbase_master_xmn_size	Maximum value for the young generation heap size for the single AMS HBase Master.

1.3.4. General Guidelines

The Metrics Collector Mode, TTL Settings, Memory Settings, and disk space requirements for AMS are dependent 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	metrics_collector_heap_size=1024 hbase_regionserver_heapsize=512 hbase_master_heapsize=512 hbase_master_xmn_size=128
PoC	1-5	5GB	embedded	Reduce TTLs to 30 Days	metrics_collector_heap_size=1024 hbase_regionserver_heapsize=512 hbase_master_heapsize=512 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

Cluster Environment	Host Count	Disk Space	Collector Mode	TTL	Memory Settings
					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

1.4. Moving the Metrics Collector

Use this procedure to move the Ambari Metrics Collector to a new host. For information and guidelines on tuning the Ambari Metrics Service, see [Performance Tuning](#).

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
```

where **ambari.server** is the Ambari Server host, **cluster.name** is your Cluster Name, and **metrics.collector.hostname** is the host running the 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
```

where **ambari.server** is the Ambari Server host, **cluster.name** is your Cluster Name, and **metrics.collector.hostname** is the host that will run the Metrics Collector.

4. In **Ambari Web**, go the Host page where you installed the new Metrics Collector. Click to Install the Metrics Collector component from the Host page.
5. In **Ambari Web**, start the **Ambari Metrics** service.
6. For every service, use **Ambari Web** > **Service Actions** > **Restart All** to start sending metrics to the new collector.

1.5. (Optional) Enabling Individual Region, Table, and User Metrics for HBase

Ambari disables some HBase metrics (per region, per table, and per user) by default. HBase metrics can be numerous and can cause performance issues. HBase RegionServer metrics are available by default.

If you want Ambari to collect per-region, per-table, and per-user metrics about HBase, you can do the following. It is **highly recommended** that you test turning on this option and confirm that your AMS performance is acceptable.

1. On the Ambari Server, browse to:

```
/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. 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 will need to re-apply this change to the template file.

2. Ambari Log Search (Technical Preview)

This chapter describes the Technical Preview release of **Ambari Log Search**.

- [Log Search Architecture \[128\]](#)
- [Installing Log Search \[129\]](#)
- [Using Log Search \[130\]](#)



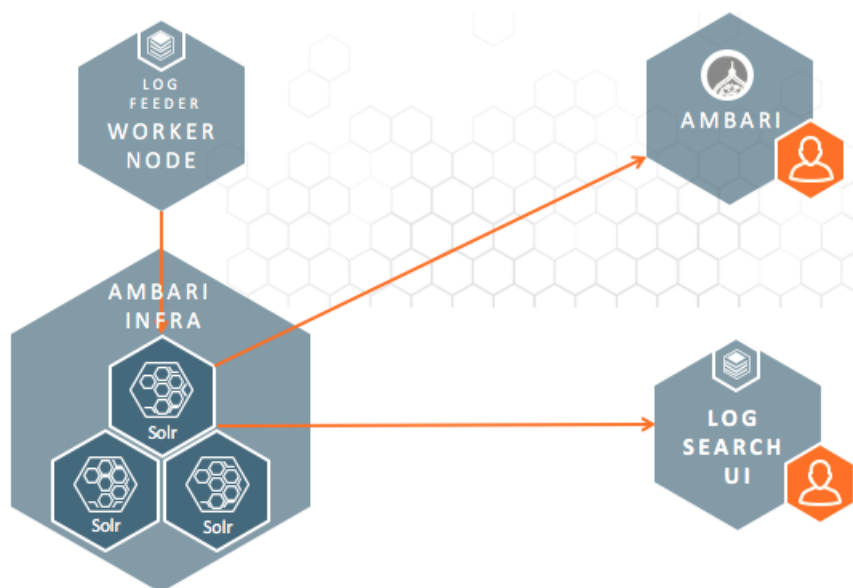
Note

Log Search is currently a Technical Preview and should be restricted for use in non-production clusters with less than 150 nodes only.

2.1. Log Search Architecture

Ambari Log Search provides distributed log parsing, and 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 comprise the Log Search solution:

- [Log Feeder \[129\]](#)
- [Log Search Server \[129\]](#)



2.1.1. Log Feeder

The Log Feeder component parses component logs. A Log Feeder is deployed to every node in the cluster and is aware of the location and format of all component logs on that host. When started, the Log Feeder starts parsing all known component logs, and sends those logs to the Apache Solr instances, managed by the Ambari Infra service, to be indexed.

By default, only FATAL,ERROR, and WARN level logs are captured by the Log Feeder. Log levels beyond FATAL,ERROR, and WARN can be temporarily or permanently added

using either the Log Search UI's filter settings  (for temporary log level capture), or through the Log Search configuration in Ambari.

2.1.2. Log Search Server

The Log Search Server hosts the Log Search UI web application, and provides the API that is used by Ambari and the Log Search UI to access the indexed component logs. The Log Search UI is used to visualize, explore, and search indexed component logs. When accessed, the Log Search UI will prompt for authentication. A valid Ambari User (local user, or LDAP user) is required to login.

2.2. Installing Log Search

Log Search is a built-in service in Ambari 2.4 and can be added during a new installation using the [+Add Service](#) menu. The Log Feeders will be automatically installed on all nodes in the cluster; there is no need to selectively place them on specific hosts. The Log Search Server can be placed on any host in the cluster, but it is recommended to be installed on the same server as the Ambari Server.

2.3. Using Log Search

Using **Ambari Log Search** includes the following activities:

- [Accessing Log Search \[130\]](#)
- [Using Log Search To Troubleshoot \[131\]](#)
- [Viewing Service Logs \[132\]](#)
- [Viewing Access Logs \[132\]](#)

2.3.1. Accessing Log Search

Once Log Search has been installed, there are three ways to search the logs that have been indexed.

- [Ambari Background Ops Log Search Link \[130\]](#)
- [Host Detail Logs Tab \[131\]](#)
- [Log Search UI \[131\]](#)

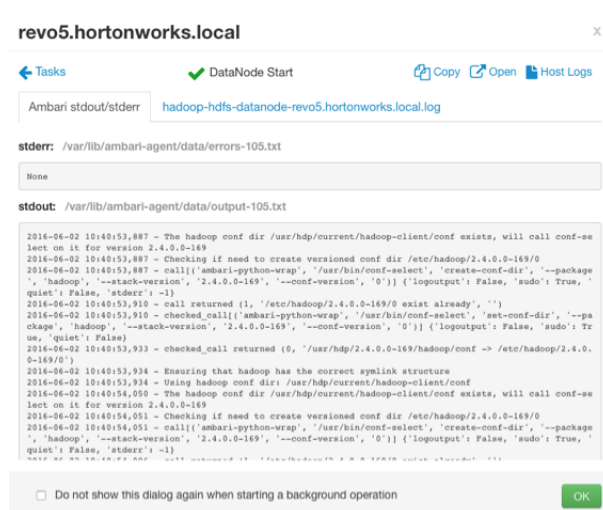


Note

Single Sign On (SSO) between Ambari and Log Search is not currently available in the Tech Preview.

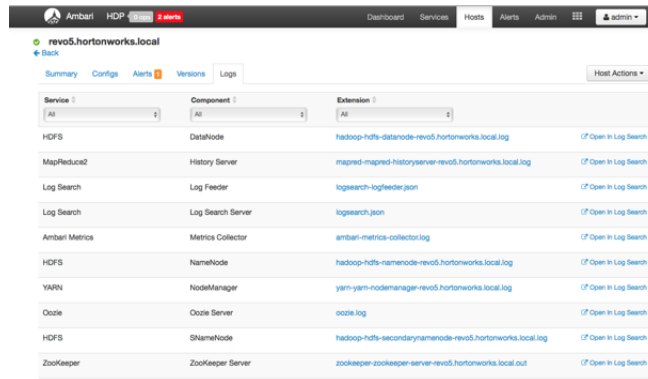
2.3.1.1. Ambari Background Ops Log Search Link

When performing lifecycle operations such as starting or stopping services, having the logs on hand when an operation fails is critical to understand why the operation has failed. For each start, or stop operation, component logs are now available in a new tab in the [Background Ops](#) dialog titled with the name of the components log file. A link in that same dialog will take you to the Host Detail Logs Tab that shows a list of all the log files that have been indexed and can be viewed for a specific host.



2.3.1.2. Host Detail Logs Tab

A **Logs** tab has been added to each host detail page and contains a list of log files organized by service, component, and type that have been indexed and can be viewed. Each of these files can be opened and searched using a link to the Log Search UI.



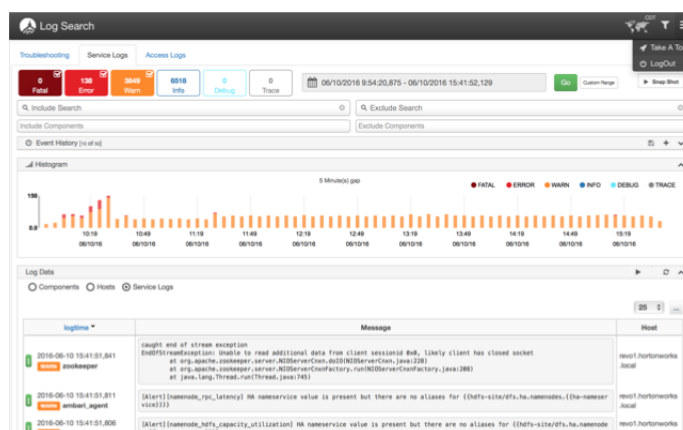
Service	Component	Extension	
HD FS	DataNode	hadoop-hdfs-datanode-rev05.hortonworks.local.log	Open In Log Search
MapReduce	History Server	mapred-mapred-historyserver-rev05.hortonworks.local.log	Open In Log Search
Log Search	Log Feeder	logsearch-logfeeder.json	Open In Log Search
Log Search	Log Search Server	logsearch.json	Open In Log Search
Ambari Metrics	Metrics Collector	ambari-metrics-collector.log	Open In Log Search
HD FS	NameNode	hadoop-hdfs-namenode-rev05.hortonworks.local.log	Open In Log Search
YARN	NodeManager	yarn-yarn-nodemanager-rev05.hortonworks.local.log	Open In Log Search
Oozie	Oozie Server	oozie.log	Open In Log Search
HD FS	SecondaryNameNode	hadoop-hdfs-secondarynamenode-rev05.hortonworks.local.log	Open In Log Search
ZooKeeper	ZooKeeper Server	zookeeper-zookeeper-server-rev05.hortonworks.local.out	Open In Log Search

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.

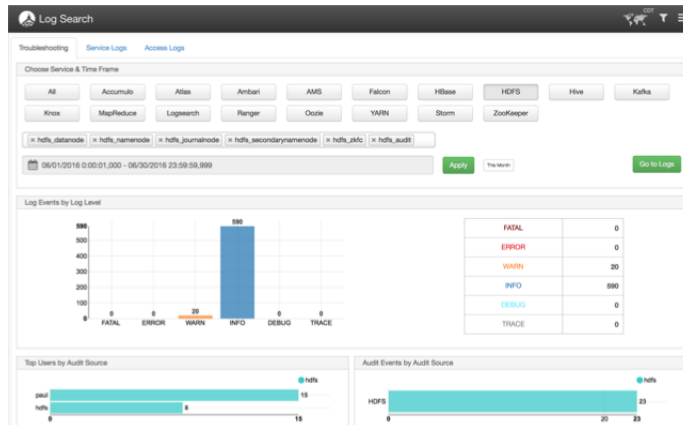
To see a guided tour of Log Search UI features, choose **Take a Tour** from the Log Search UI main menu. Click **Next** to view each topic in the guided tour series.



2.3.2. Using Log Search To Troubleshoot

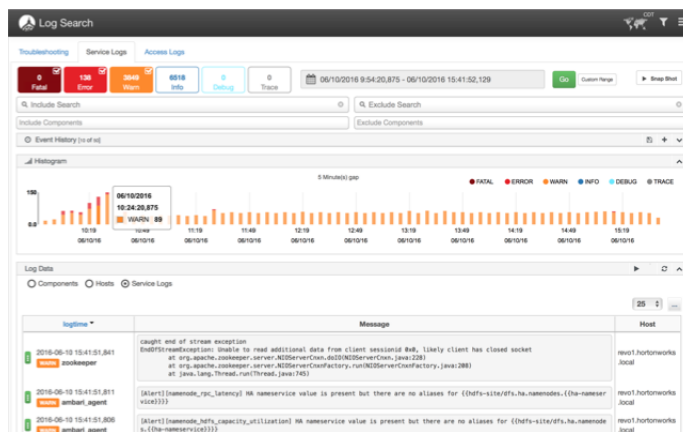
When trying to find logs related to a specific problem, the **Troubleshooting** tab in the UI helps guide you to the right logs by selecting the Service, Components, and Time Frame

related to the problem you are troubleshooting. For example you can select HDFS and the UI will automatically search for HDFS related components. You can select a time frame of Yesterday, Last Week, or a specific day/time range. Each of these filters will be used to preview logs that were gathered that match those constraints. When ready to look at the logs that match those filters, you can click on **Go to Logs**.



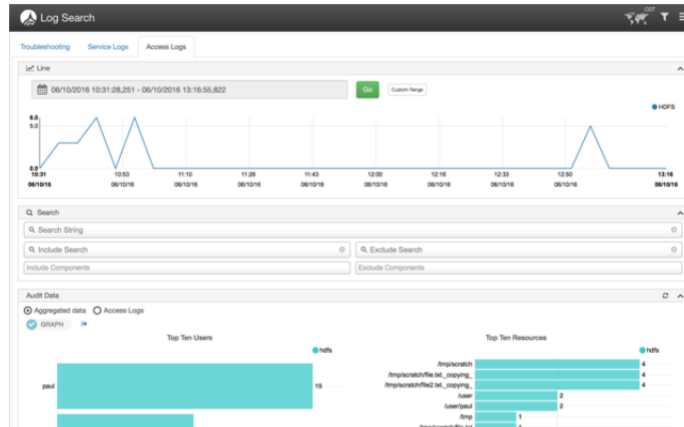
2.3.3. Viewing Service Logs

The **Service Logs** tab is useful when looking to search across all component logs for specific keywords, filter for log levels, components, and time ranges. The UI is organized to allow you to quickly see how many logs were captured for each log level across the entire cluster, search for keywords, include and exclude components, and see which logs match your search query.



2.3.4. Viewing Access Logs

When troubleshooting HDFS related issues, it can be very helpful to search for and spot trends in HDFS access by users. The **Access Logs** tab allows you to view HDFS Audit log entries by time, and 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 be helpful for looking for anomalies or hot/cold data sets.



3. Ambari Infra

Many services in HDP depend on core services such as Apache Solr for indexing data. For example, Atlas uses indexing services for tagging lineage free text search, and Ranger uses indexing for audit data. The role of Ambari Infra is to provide these common shared services for stack components.

In this release, 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, but Hortonworks recommends that you install multiple Infra Solr Instances in order to provide distributed indexing and search for Atlas, Ranger, and LogSearch (Technical Preview). This can be accomplished by simply adding additional Ambari Infra Solr Instances to existing cluster hosts through Ambari's **+Add Service** capability. The number of Infra Solr Instances to deploy is dependent on the number of nodes in the cluster and the services deployed.

Due to the fact that the 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, Atlas, Ranger, and Log Search (Technical Preview) are dependent on the Ambari Infra Service.



Note

The Infra Solr Instance is only intended for use by HDP components; Hortonworks does not support the use of the Infra Solr Instance by third party components or applications.