Hortonworks Data Platform

Apache Ambari Views

(Nov 28, 2016)

docs.hortonworks.com

Hortonworks Data Platform: Apache Ambari Views

Copyright © 2012-2016 Hortonworks, Inc. All 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.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Table of Contents

1. Using Ambari Views 1
2. Preparing Ambari Server for Views 3
3. Running Ambari Server Standalone 5
    3.1. Prerequisites 5
    3.2. Standalone Server Setup 6
    3.3. Reverse Proxy 7
4. Configuring Views for Kerberos 8
5. Technical Preview: Migrating Hue to Ambari Views 9
    5.1. Requirements for Hue-to-Views Migration 9
    5.2. Creating a Hue-to-Views instance 10
    5.3. Example: Using the Hue-to-Views Migration Tool 13
6. Using the YARN Queue Manager View 15
    6.1. Configuring your Cluster for the Capacity Scheduler View 15
    6.2. Creating a Capacity Scheduler View Instance 15
        6.2.1. User Permissions for YARN Queue Manager View 18
    6.3. Using the YARN Queue Manager View 22
        6.3.1. Setting up Queues 22
        6.3.2. Configuring Queues 26
        6.3.3. Configuring Cluster Scheduler Settings 28
        6.3.4. Applying the Configuration Changes 29
    6.4. Troubleshooting 30
7. Using the Files View 31
    7.1. Configuring Your Cluster 31
    7.2. Creating and Configuring a Files View Instance 32
        7.2.1. Kerberos Settings 33
        7.2.2. Cluster Configuration: Local 34
        7.2.3. Cluster Configuration: Custom 34
        7.2.4. Troubleshooting 34
8. Using the Falcon View 36
    8.1. Configuring Your Cluster 36
        8.1.1. Setup HDFS Proxy User 36
        8.1.2. Setup HDFS User Directory 37
    8.2. Installing and Configuring the Falcon View 37
    8.3. Accessing the Falcon Documentation 41
9. Using the Hive View 42
    9.1. Upgrading Your Hive View 43
    9.2. Configuring Your Cluster 44
        9.2.1. Setup HDFS Proxy User 44
        9.2.2. Setup HDFS User Directory 45
    9.3. Creating the Hive View Instance 45
        9.3.1. Settings and Cluster Configuration 46
        9.3.2. User Permissions for Hive Views 50
        9.3.3. Kerberos Setup for Hive Views 50
    9.4. Using the Hive View 51
        9.4.1. Query Tab 51
        9.4.2. Saved Queries Tab 57
        9.4.3. History Tab 57
        9.4.4. UDF Tab 58

9.5. Upload Table tab 58
9.6. Troubleshooting 60
10. Using the Pig View 61
10.1. Configuring Your Cluster 61
10.1.1. Setup HDFS Proxy User 61
10.1.2. Setup WebHCat Proxy User 62
10.1.3. Setup HDFS User Directory 63
10.2. Creating the Pig View Instance 63
10.2.1. Getting Correct Configuration Values for Manually-Deployed Clusters 65
10.2.2. User Permissions for Pig Views 66
10.2.3. Kerberos Setup for Pig Views 67
10.3. Using the Pig View 68
10.3.1. Writing Pig Scripts 68
10.3.2. Viewing Pig Script Execution History 69
10.3.3. User-Defined Functions (UDFs) Tab 69
11. Using the Slider View 71
11.1. Deploying the Slider View 71
12. Using the SmartSense View 72
12.1. Configuring Your Cluster 72
12.2. Creating the SmartSense View Instance 72
12.3. Using the SmartSense View 74
13. Using the Storm View 75
13.1. Configuring Your Cluster 75
13.2. Creating the Storm View Instance 75
13.3. Using the Storm View 77
13.3.1. Monitoring Storm Cluster Status: the Cluster Summary Page 77
13.3.2. Monitoring Topology Status: the Topology Summary Page 79
13.3.3. Looking Up Configuration Values: the Component Summary Page..... 82
14. Using the Tez View 84
14.1. Configuring Your Cluster for Tez View 84
14.2. Creating or Editing the Tez View Instance 85
14.2.1. User Permissions for Tez Views 87
14.2.2. Kerberos Setup for Tez Views 88
14.3. Using the Tez View 89
14.3.1. Understanding Directed Acyclic Graphs (DAGs), Vertices, and Tasks 89
14.3.2. Identifying the Tez DAG for Your Job 90
14.3.3. Understanding How Your Tez Job Is Executed 91
14.3.4. Identifying Causes of Failed Jobs 92
14.3.5. Viewing All Failed Tasks 93
14.3.6. Using Counters to Identify the Cause of Slow-Performing Jobs 93
15. Using Workflow Designer View -Tech Preview 96
16. Using Zeppelin View - Tech Preview 97

List of Figures

3.1. Configuring Views with your HDP Cluster 6
9.1. Default Hive View Settings 46
9.2. Default Hive View Cluster Configuration 47
9.3. HDFS Service Page in Ambari 49
9.4. Using the Filter to Search Advanced hdfs-site Settings 50
9.5. Granting User Permissions to Hive Views 50
9.6. Hive View Database Explorer 51
9.7. Query Editor 52
9.8. Query Results and Logs in Hive View Query Editor 53
9.9. Query Editor Textual Explain Feature 54
9.10. Query Editor Visual Explain Feature 55
9.11. Tez View Query Debugging Option 56
9.12. Query Editor Error Message Summary Window 56
9.13. Query Editor Error Message Details Window 57
9.14. Saved Queries Tab 57
9.15. History Tab 58
9.16. UDF Tab 58
10.1. Pig View Details and Settings 64
10.2. Pig View Cluster Configuration 64
10.3. HDFS Service Page in Ambari 66
10.4. Using the Filter to Search Advanced hdfs-site Settings 66
10.5. Granting User Permissions to Pig Views 67
10.6. Kerberos Settings for Pig Views 68
10.7. Pig Script Running in the Pig View 69
10.8. Pig View Script History Tab 69
10.9. Pig View UDFs Tab 70
14.1. Tez View Create Instance Page 85
14.2. Tez View Instance Page 87
14.3. Granting User Permissions to Tez Views 88
14.4. SQL Query Execution in Hive 90
14.5. Tez View Column Selector Dialog Box 91
14.6. View Tab in Tez View 92
14.7. DAG Details Window 93
14.8. Tez View All Tasks Tab 93
14.9. Tez View DAG-Level Counters Tab 94
14.10. Tez View Vertex-Level Counters Tab 94
14.11. Tez View Task-Level Counters Tab 94

List of Tables

9.1. Hive View Instance Details 46
9.2. Finding Cluster Configuration Values for the Hive View using Ambari 48
9.3. Hive View Settings for NameNode High Availability 49
9.4. Troubleshooting Hive Views Errors 60
10.1. Finding Cluster Configuration Values for the Pig View in Ambari 65
10.2. Pig View Settings for NameNode High Availability 65
12.1. SmartSense View Instance Details 73
12.2. SmartSense View Instance Settings 74
13.1. Storm View Instance Details 77
13.2. Storm View Instance Settings 77
14.1. Cluster Configurations for Tez View 84
14.2. Cluster Configuration Values for the Tez View in Ambari 86
14.3. Kerberos Settings for Tez Views 88
14.4. Tez Job Status Descriptions 91

List of Examples

8.1. Substitute *#USER#* 40

1. Using Ambari Views

Ambari includes the Ambari Views Framework, which allows for developers to create UI components that "plug into" the Ambari Web interface. Ambari includes a built-in set of Views that are pre-deployed for you to use with your cluster. This guide provides information on configuring the built-in set of Views, as well as information on how to configure Ambari Server for "standalone" operation.

Views can be deployed and managed in the "operational" Ambari Server that is operating your cluster. In addition, Views can be deployed and managed in one or more separate "standalone" Ambari Servers. Running "standalone" Ambari Server instances is useful when users who will access views will not have (and should not) have access to that Ambari Server that is operating the cluster. As well, you can run one or more separate Ambari Server instances "standalone" for a scale-out approach to handling a large number of users. See Running Ambari Standalone for more information.

Important

It is critical that you prepare your Ambari Server for hosting views. It is strongly recommended you increase the amount of memory available to your Ambari Server, and that you run additional "standalone" Ambari Servers to host the views. See Preparing Ambari Server for Views and Running Ambari Server Standalone for more information.

View	Auto-Created*	Description	HDP Stacks	Required Services
YARN Queue Manager	Yes	Provides a visual way to configure YARN capacity scheduler queue capacity.	HDP 2.3 or later	YARN
Files	Yes	Allows you to browse the HDFS file system.	HDP 2.2 or later	HDFS
Hive	Yes	Exposes a way to find, author, execute and debug Hive queries.	HDP 2.3 or later	HDFS, YARN, Hive
Hue-to-Views	No	Supports migrating Hue artifacts to an Ambari View. This View is **Tech Preview**	HDP 2.4 or later	Hue
Pig	No	Provides a way to author and execute Pig Scripts.	HDP 2.2 or later	HDFS, Hive, Pig
Slider	No	A tool to help deploy and manage Slider-based applications. This view has been marked deprecated.	HDP 2.2 or later	HDFS, YARN
SmartSense	Yes	Allows you to capture bundles, set bundle capture schedule, and view and download captured bundles.	HDP 2.0 or later	SmartSense
Storm	No	Supports monitoring Storm cluster status and topologies.	HDP 2.5 or later	Storm
Tez	Yes	View information related to Tez jobs that are executing on the cluster.	HDP 2.2.4.2 or later	HDFS, YARN, Tez
Workflow Designer	No	This View is **Tech Preview**	HDP 2.4 or later	Oozie
Zeppelin	Yes	This View is **Tech Preview**	HDP 2.5 or later	Zeppelin

Ambari "Auto-creates" some views, if the service utilized by that view is added to the cluster. "Auto-create" means that Ambari creates and instance of the view and displays that instance to users via Ambari web, automatically. For other services, an Ambari Admin must create the view instance. For example, if YARN service is added to the cluster, the YARN Queue Manager View displays to Ambari Web users.

Learning More About Views

You can learn more about the Views Framework at the following resources:

Resource	URL
Administering Views	Hortonworks Data Platform Apache Ambari Administration - Managing Views
Ambari Project Wiki	https://cwiki.apache.org/confluence/display/AMBARI/Views
Example Views	https://github.com/apache/ambari/tree/trunk/ambari-views/examples
View Contributions	https://github.com/apache/ambari/tree/trunk/contrib/views

2. Preparing Ambari Server for Views

When hosting multiple views in Ambari, it is **strongly recommended** you increase the amount of memory available to the Ambari Server. Since each view requires it's own memory footprint, increasing the Ambari Server maximum available memory will help support multiple deployed views and concurrent use.

1. On the Ambari Server host, edit the ambari-env.sh file:

   ```
   vi /var/lib/ambari-server/ambari-env.sh
   ```

2. For the `AMBARI_JVM_ARGS` variable, replace the default `-Xmx2048m` with the following:

   ```
   -Xmx4096m -XX:PermSize=128m -XX:MaxPermSize=128m
   ```

3. Restart Ambari Server for this change to take effect.

   ```
   ambari-server restart
   ```

If the Ambari Server instance is configured for HTTPS, a trust store must also be configured so that the deployed views are able to trust the certificate used by the Ambari Server during API communications. The process includes creating a trust store with the certificate that the Ambari Server has been configured to use, and then setting up the Ambari Server to use the newly created trust store. The steps are included below:

1. On the Ambari Server, create a new keystore that will contain the Ambari Server's HTTPS certificate.

   ```
   keytool -import -file <path_to_the_Ambari_Server's_SSL_Certificate> -alias
    ambari-server -keystore ambari-server-truststore
   ```

   When prompted to 'Trust this certificate?' type "yes".

2. Configure the ambari-server to use this new trust store:

   ```
   ambari-server setup-security
   Using python  /usr/bin/python2.6
   Security setup options...
   ===========================================================================
   Choose one of the following options:
     [1] Enable HTTPS for Ambari server.
     [2] Encrypt passwords stored in ambari.properties file.
     [3] Setup Ambari kerberos JAAS configuration.
     [4] Setup truststore.
     [5] Import certificate to truststore.
   ===========================================================================
   Enter choice, (1-5): *4*
   Do you want to configure a truststore [y/n] (y)? *y*
   TrustStore type [jks/jceks/pkcs12] (jks): *jks*
   Path to TrustStore file : *<path to the ambari-server-truststore keystore>*
   Password for TrustStore:
   Re-enter password:
   Ambari Server 'setup-security' completed successfully.
   ```

3. Once configured, the Ambari Server must be restarted for the change to take effect.

```
ambari-server restart
```

3. Running Ambari Server Standalone

You can run one or more separate Ambari Server instances running in “standalone” mode. Running “standalone” Ambari Server instances is useful when users who will access views will not have (and should not) have access to that Ambari Server that is operating the cluster. As well, you can run one or more separate Ambari Server instances “standalone” for a scale-out approach to handling a large amount of users. See Reverse Proxy for more information.

Note

When running Ambari Server Standalone, and the cluster is being operated by an Ambari Server, you will have an option to configure that cluster as a Remote Cluster and then **use the Remote Cluster option** when configuring the view instance.

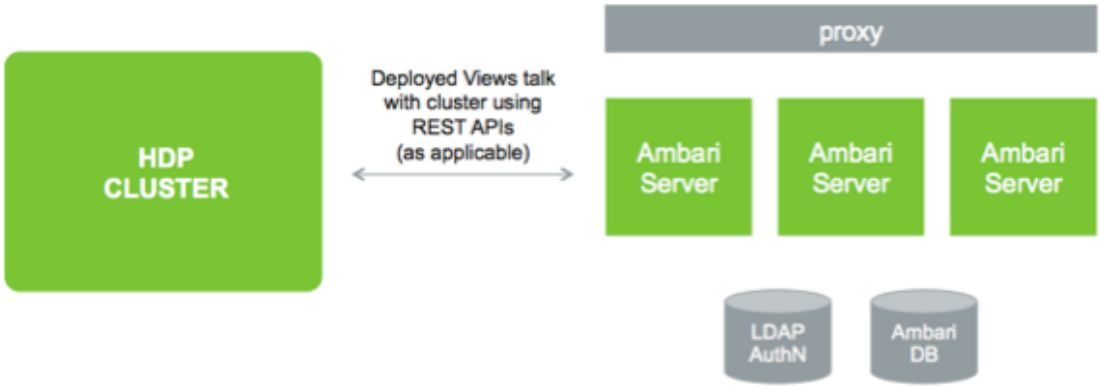
3.1. Prerequisites

There are several requirements that need to be considered when setting up multiple Ambari Server “standalone” instances:

- Ambari Server instances should be the same version.
- The Ambari Server instances should point to the same underlying database. Ensure that it is **not** the same database that is being used by an Operational Ambari Server managing the HDP cluster.
- Ambari database should be scaled and made highly-available, independent of Ambari Server.
- If using an external authentication source (such as LDAP or Active Directory), Ambari Server authentication should be configured the same for all Ambari Server instances.
- If the cluster you are accessing with Views is Kerberos-enabled, you need to configure Ambari and the Views for Kerberos.
- Run the multiple “standalone” Ambari Server instances behind a Reverse Proxy.

After your standalone Ambari Servers are setup and configured, you can configure the views to communicate with your HDP cluster.

Figure 3.1. Configuring Views with your HDP Cluster



3.2. Standalone Server Setup

Setting up a standalone Ambari Server instance is very similar to setting up an operational Ambari Server. Many of the steps are the same, with one key **exception: you do not install a cluster with a standalone Ambari Server**. A standalone Ambari Server does not manage a cluster and does not deploy or communicate with Ambari Agents. The standalone Ambari Server runs as web server instance, serving views for users.

Important

Refer to the Ambari Install Guide for the details steps for setting up an Ambari Server. For a standalone Ambari Server instance, you are not required to install a cluster.

Important

Refer to Managing Views in Hortonworks Data Platform Apache Ambari Administration for information on deploying and configuring Views.

The following table compares the high-level tasks required to setup an operational Ambari Server vs. a standalone Ambari Server.

	Operational Ambari Server	**Standalone Ambari Server**
1	Install ambari -server package	Install ambari -server package
2	Run ambari -server setup (DB, JDK)	Run ambari -server setup (DB, JDK) **Important** Do not share the DB with an Operational Ambari Server.
3	Configure external LDAP authentication	Configure external LDAP authentication
4	Install Cluster	**NA**
5	Deploy views	Deploy views
6	Create + configure view instances	Create + configure view instances
7		(Optional) Repeat for each Ambari Server instance
8		(Optional) Set up proxy for Ambari Server instances

	Operational Ambari Server	Standalone Ambari Server
9		(Optional) Set up SSL for Ambari

3.3. Reverse Proxy

If you require a larger number of users to access Ambari Views, it may be necessary to "scale-out" the Ambari Server by installing and running multiple Ambari Server standalone instances that host Ambari Views and run those instances behind a reverse proxy.

If a reverse proxy fronts the standalone Ambari Server instances, the only requirement is that the reverse proxy honors session affinity, meaning that once a session has been established the reverse proxy routes each subsequent request to the same Ambari server instance. Depending on the reverse proxy implementation, this can be accomplished in a number of different ways, including hashing client IP and using the JSESSIONID header.

Important

Using multiple Ambari Server instances and a reverse proxy in front of those instances is **not supported** for an operational Ambari Server. It is only supported for standalone Ambari Server instances (i.e. Ambari instances that are not managing a cluster).

4. Configuring Views for Kerberos

If the cluster your views will communicate with is Kerberos-enabled, you need to configure the Ambari Server instance(s) for Kerberos and be sure to configure the views to work with Kerberos.

Refer to the Set Up Kerberos for Ambari for the instructions on how to configure Ambari Server for Kerberos. Be sure to configure all standalone Ambari Server instances for Kerberos.

Important

Be sure to install the Kerberos client utilities on the Ambari Server so that Ambari can kinit.

RHEL/CentOS/Oracle Linux

```
yum install krb5-workstation
```

SLES

```
zypper install krb5-client
```

Ubuntu/Debian

```
apt-get install krb5-user krb5-config
```

Once your Ambari Server is setup for Kerberos, be sure to follow the specific instructions with each view on how to configure the view for Kerberos and the cluster for Kerberos access from the view. Also, if the view requires HDFS or WebHCat to be configured for a proxy user, **instead of using the ambari-server daemon user as the proxy user, you must user primary Kerberos principal**. For example, if you configure Ambari Server for Kerberos principal **ambari-server@EXAMPLE.COM**, this value would be **ambari-server**.

5. Technical Preview: Migrating Hue to Ambari Views

Because some users are using the open-source web interface Hue, rather than the Ambari Views Framework, to use Hortonworks Data Platform (HDP) components, Ambari 2.4 introduces a Technical Preview of the Hue-to-Views Migration tool, which is specifically designed to migrate existing Apache Hue artifacts to an Ambari view.

This chapter describes how to configure for, create, and use this Technical Preview tool:

- Requirements for a Hue-to-Views migration
- Creating a Hue-to-Views instance
- Using the Hue-to-Views Migration Tool

5.1. Requirements for Hue-to-Views Migration

Prerequisites

- Hue service must have a network connection to an Ambari Server serving the Hue-to-Views migration tool.
- Ambari Server must be enabled as a views server.
- Database access rights must be granted to the Hue back-end database (mysql, oracle, and postgresql).

Supported Artifacts and Expectations

The Hue-to-Views migration tool supports migrating the following artifacts:

Hive

- Saved Queries
- Query History

Pig

- Saved scripts
- Pig Jobs

Important

Scripts maintain the same status in the Ambari Views Framework as they had in Hue. The Hue-to-Views Migration tool does not validate scripts.

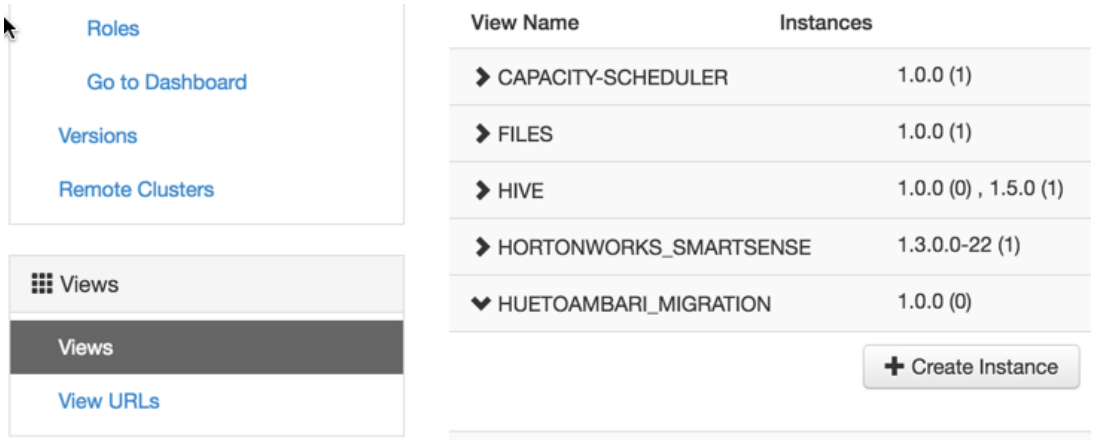
Some hive queries (mysql version 0.4.0) may fail to migrate, and cause the Hue-to-Views migration to stop.

The Hue-to-Views migration tool does not support HA. You must provide the current, active namenode for the target cluster regarding the Webhdfs URI for Ambari.

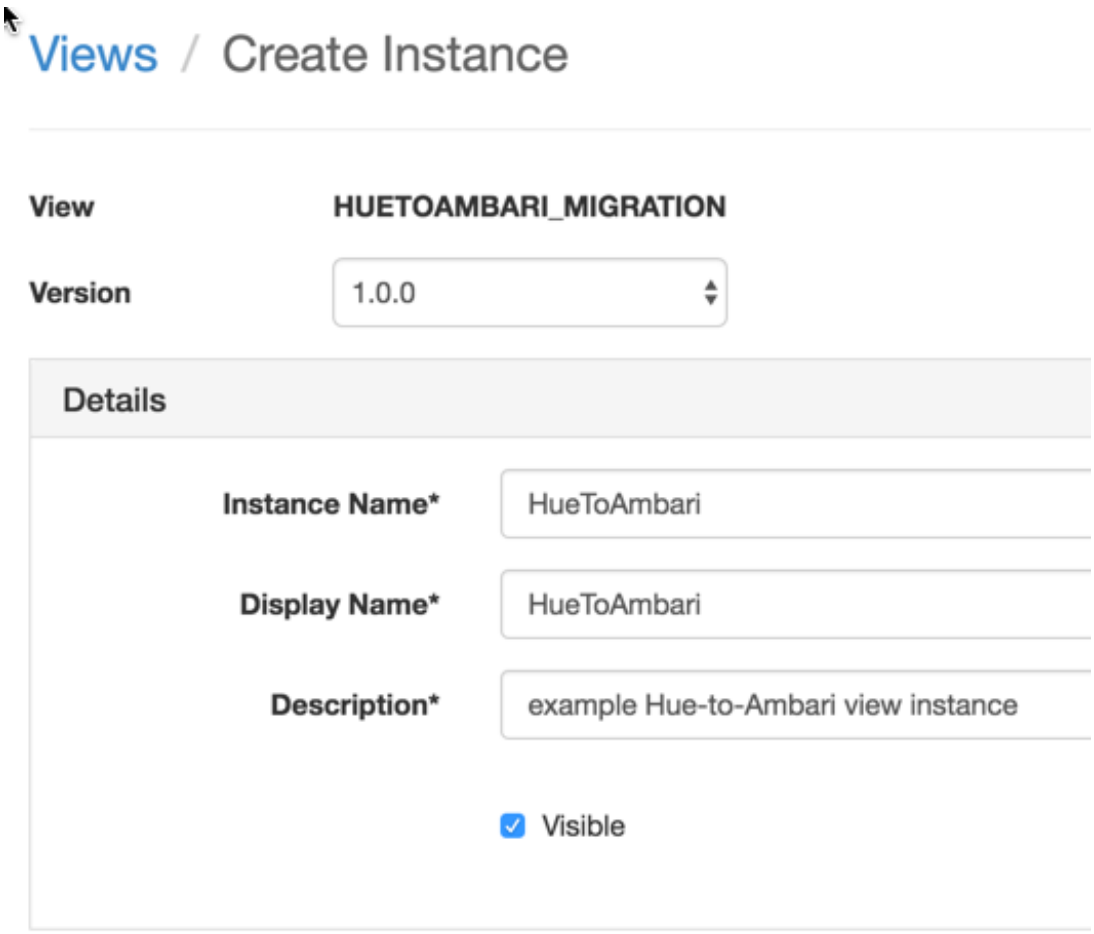
5.2. Creating a Hue-to-Views instance

To create an instance of a Hue-to-Views view on the Ambari Dashboard, use Manage Ambari:

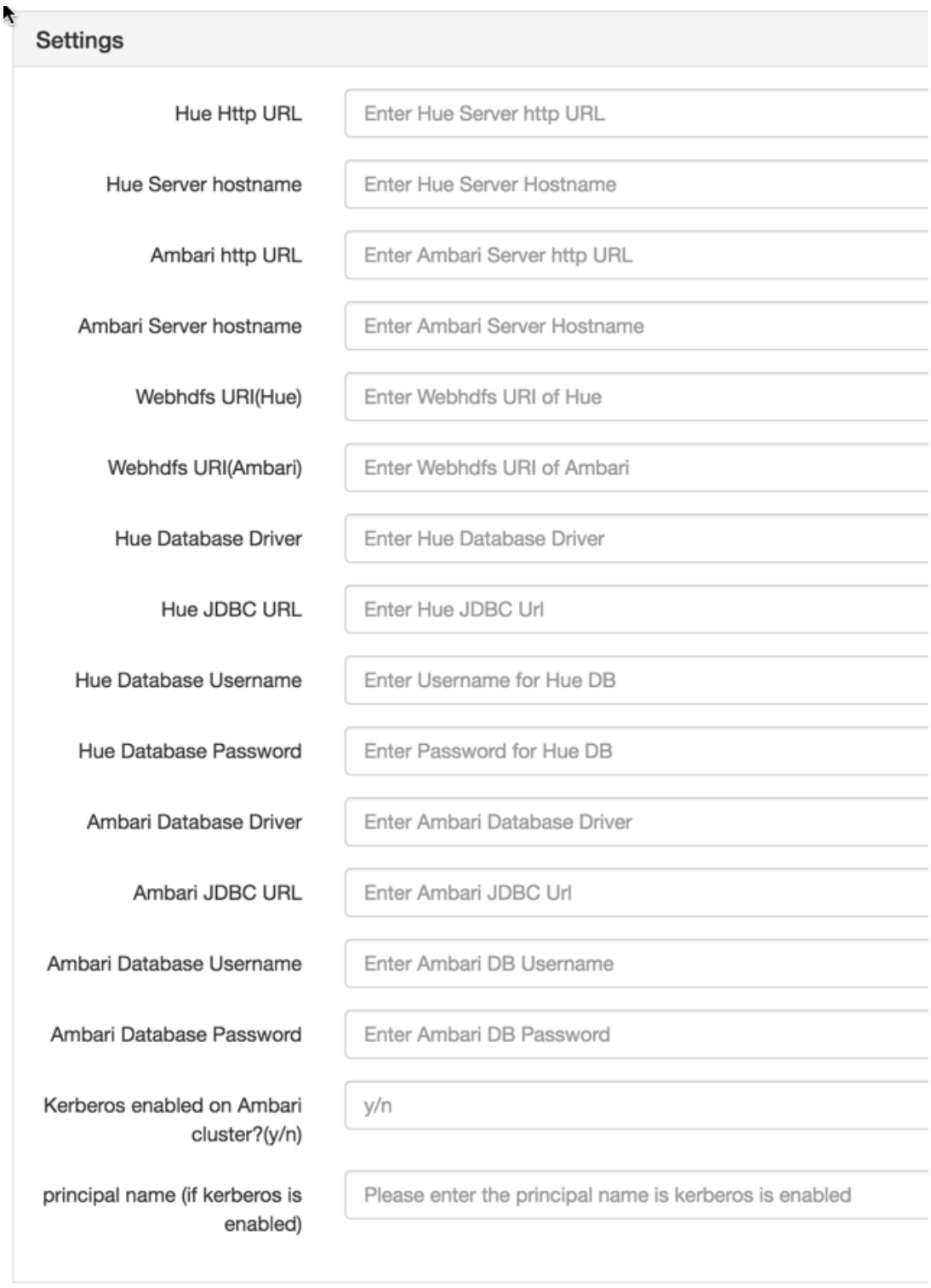
1. In Ambari Web, select admin > Manage Ambari.
2. On the Ambari Admin page, click Views.
3. In View Name, browse to HUETOAMBARI_MIGRATION and expand.
4. Click Create Instance



5. On Views/Create Instance, provide required values for the instance name, display name, and description.



6. Provide required values for all Hue-to-Views migration settings.



7. Click Save.

The new, HUETOAMABARI view displays in the list of Ambari Views. To use the new view, click Go To Dashboard.

For example configuration settings and descritpions, see, Example: Using the Hue-to-Views Migration Tool.

5.3. Example: Using the Hue-to-Views Migration Tool

In a test environment, configure an example, 3-node cluster using the following settings:

- Hue Server url : c6401.ambari.apache.org
- Hue NameNode URI: c6402.ambari.apache.org
- NameNode port: 50070
- Hue Database Name(mysql): Huedb
- Hue Database username(mysql): hue
- Hue Database Password(mysql): hue
- Ambari Hostname: c6402.ambari.apache.org
- Ambari Database Name (postgresql): ambari
- Ambari Database username (postgresql): ambari
- Ambari Database Password (postgresql): bigdata

.

Property	Description	Syntax	Example
Hue Http URL	Https url where Hue server is located	<hue http url>	http://c6401.ambari.apache.org:8000/
Hue Server hostname	hostname on which Hue runs (the host from which you migrate data)	<hue hostname>	c6401.ambari.apache.org
Ambari http URL	http url of the Ambari Server	<ambari http url>	http://c6402.ambari.apache.org:8080/
Ambari Server hostname	hostname of the Ambari server	<ambari hostname>	c6402.ambari.apache.org
Webhdfs URI (Hue)	Namenode URI of Hue	webhdfs://<hostname>:50070	webhdfs://c6402.ambari.apache.org:50070
Webhdfs URI (Ambari)	NameNode URI of Ambari	webhdfs://<hostname>:50070	webhdfs://c6402.ambari.apache.org:50070
Hue Database Driver	JDBC Driver to access Hue DB	<db driver>	com.mysql.jdbc.Driver
Hue JDBC URL	JDBC Url to access Hue DB	jdbc:<dbtype>://<hostname>/<db name>	jdbc:mysql://c6401.ambari.apache.org/huedb
Hue Database Username	Hue Database Username	<db username>	hue
Hue Database Password	Hue Database Password	<db password>	hue
Ambari Database Driver	JDBC Driver to access Ambari DB	<db driver>	org.postgresql.Driver

Property	Description	Syntax	Example
Ambari JDBC URL	JDBC Url to access Ambari DB	jdbc:<dbtype>://<hostname>/<db name>	jdbc:postgresql://c6402.ambari.apache.org:5432/ambari
Ambari Database Username	database username for Ambari	<db username>	ambari
Ambari Database Password	database password for the Ambari database	<db password>	bigdata
Kerberos enabled on Ambari cluster? (y/n)	(y/n) for Kerberos	<y/n>	n
Principal name (if Kerberos enabled)	If Kerberos is enabled, you must provide Principal Name	<principal username>	ambari-cl1

For more information about how to obtain driver class information, see https://docs.oracle.com/javase/7/docs/api/java/sql/DriverManager.html.

6. Using the YARN Queue Manager View

The Yarn Capacity Scheduler allows for multiple tenants in an HDP cluster to share compute resources according to configurable workload management policies.

The YARN Queue Manager View is designed to help hadoop operators configure these policies for YARN. In the View, operators can create hierarchical queues and tune configurations for each queue to define an overall workload management policy for the cluster.

In this section:

- Configuring your Cluster for the YARN Queue Manager View
- Creating a Capacity Scheduler View Instance
- Using the Capacity Scheduler View
- Troubleshooting

6.1. Configuring your Cluster for the Capacity Scheduler View

The Capacity Scheduler View requires that the cluster is managed by Ambari – the view utilizes the Ambari Server API.

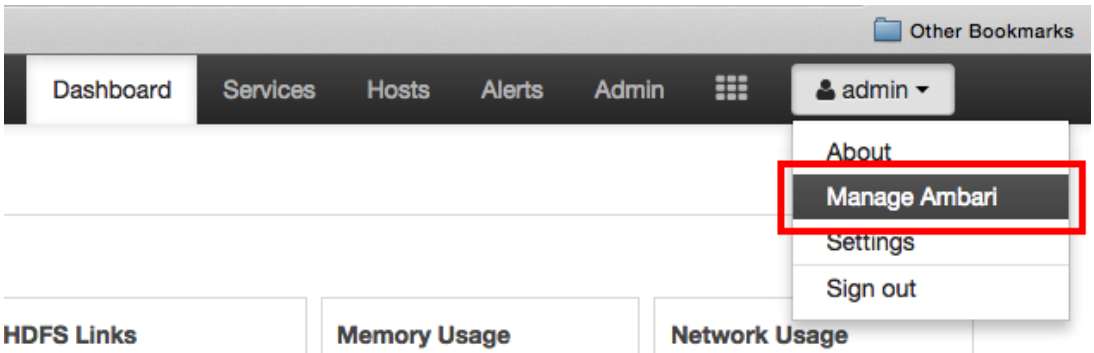
6.2. Creating a Capacity Scheduler View Instance

When you deploy a cluster using Ambari, a Capacity Scheduler View instance is automatically created. If you do not need to reconfigure the Ambari-created cluster, proceed to the next section, Using the YARN Queue Manager View.

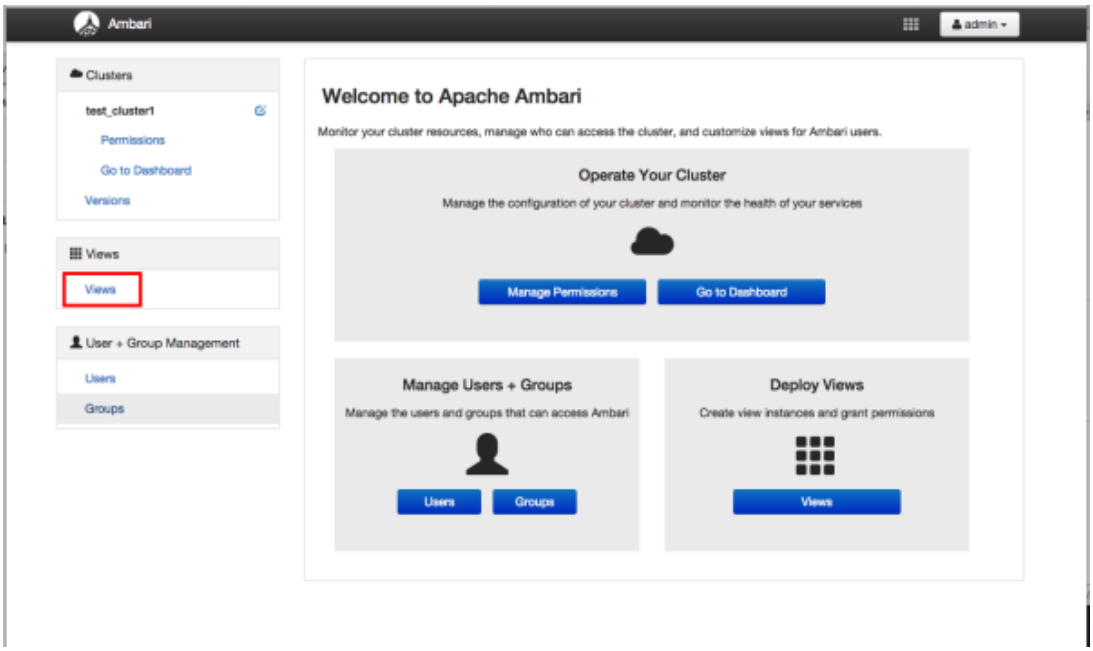
If you have deployed your cluster manually, or if you need to re-configure the Ambari-created YARN Queue Manager View, you can use the information in this section to create and configure a view instance.

Use the following steps to set up a Capacity Scheduler / YARN Queue Manager view instance.

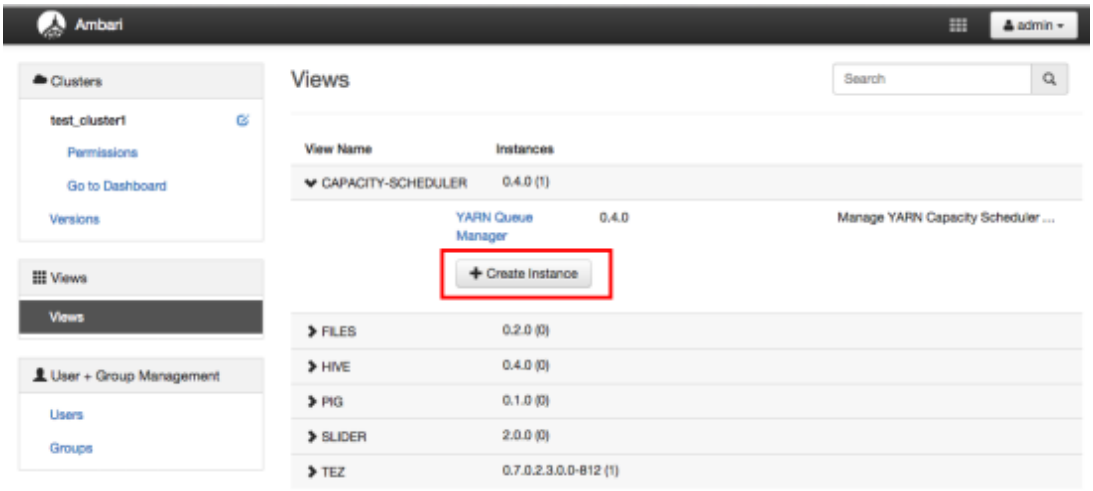
1. Select **admin > Manage Ambari** in the Ambari Web top menu.



2. On the Manage Ambari page, click **Views**.



3. On the Views page, click **CAPACITY-SCHEDULER**, then click **Create Instance**.



4. In the Details box on the Create Instance page, type in an instance name, display name, and a description for the view.

Note

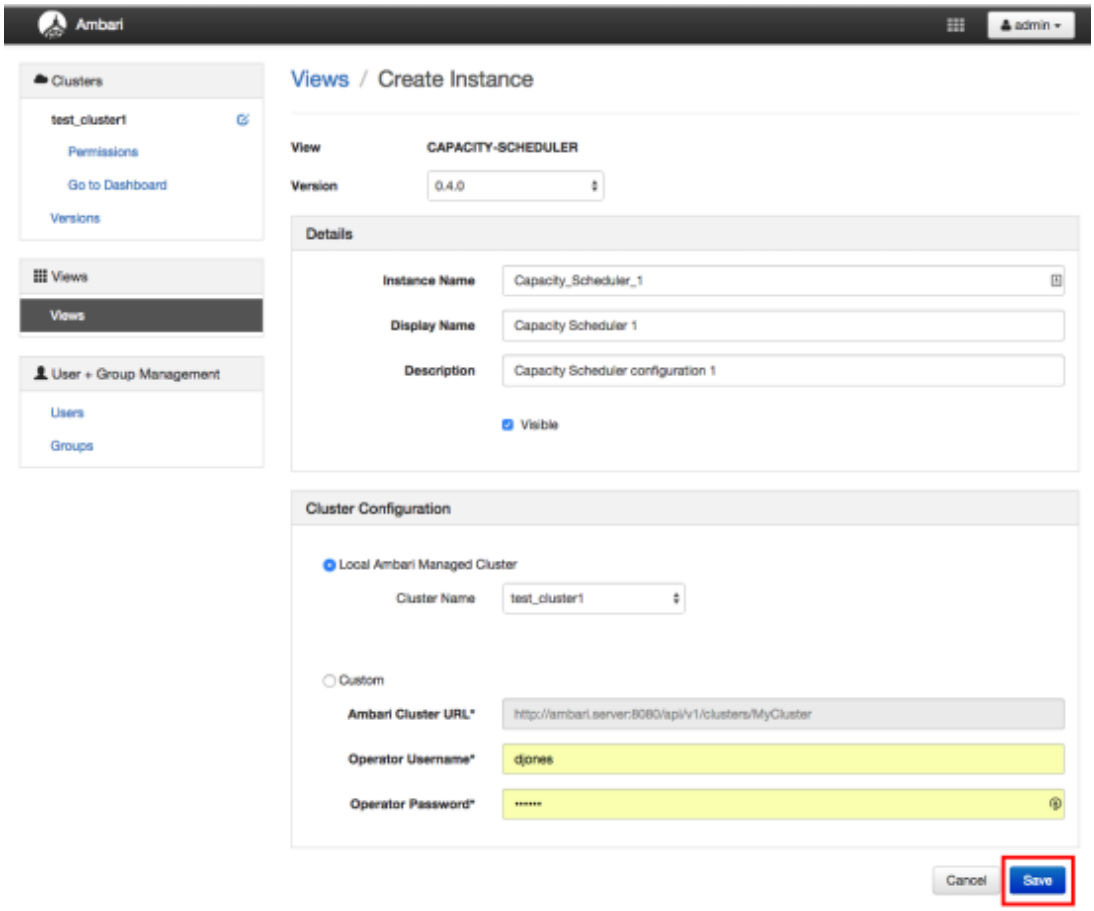
The instance name cannot contain spaces or special characters.

5. In the Cluster Configuration box on the Create Instance page, configure the view to communicate with the HDP cluster.

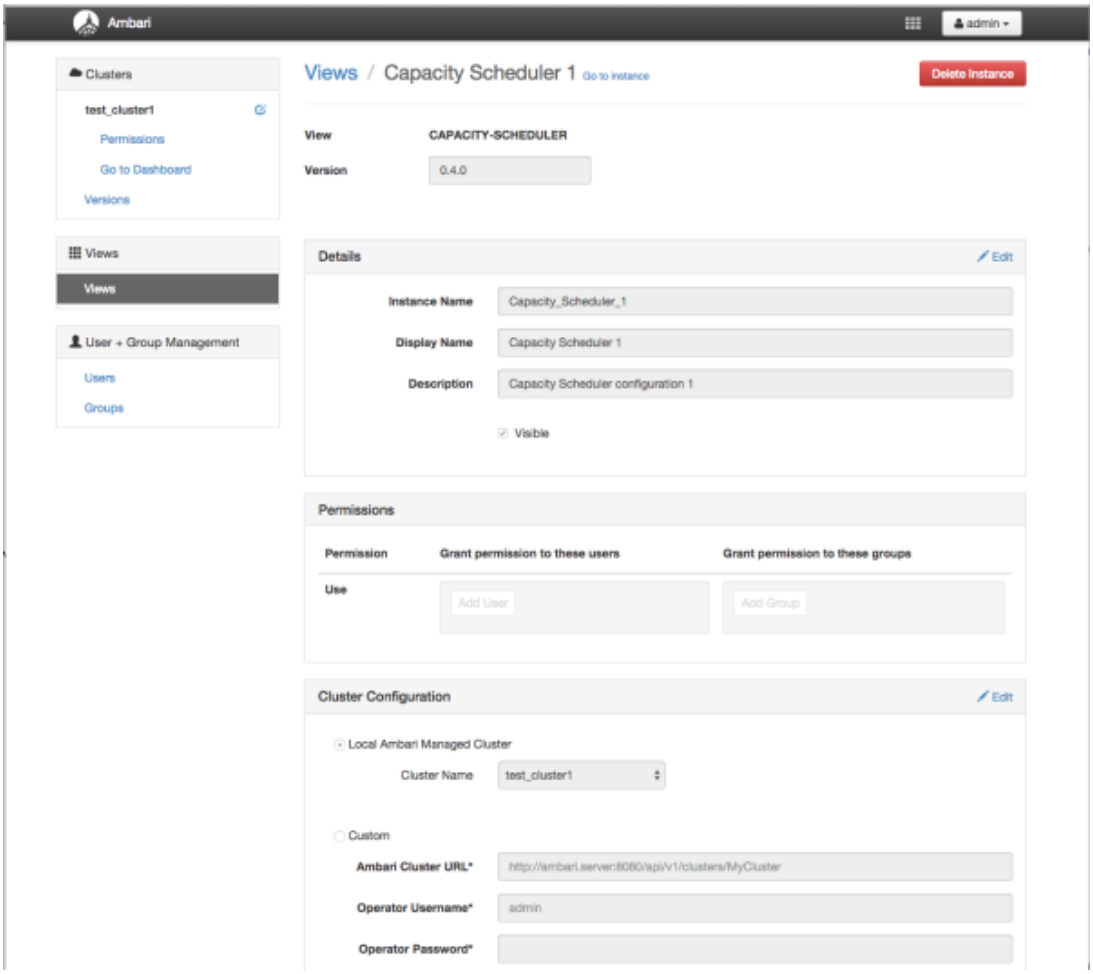
   - For HDP clusters that are local (managed by the local Ambari Server), select the **Local Ambari Managed Cluster** option, then select the local cluster name.

- To configure the view to work with HDP clusters that are remote (not part of this Ambari Server instance), select the **Custom** option, then specify the remote Ambari cluster API URL and the Ambari cluster user name and password.

6. Click **Save** at the bottom of the page.



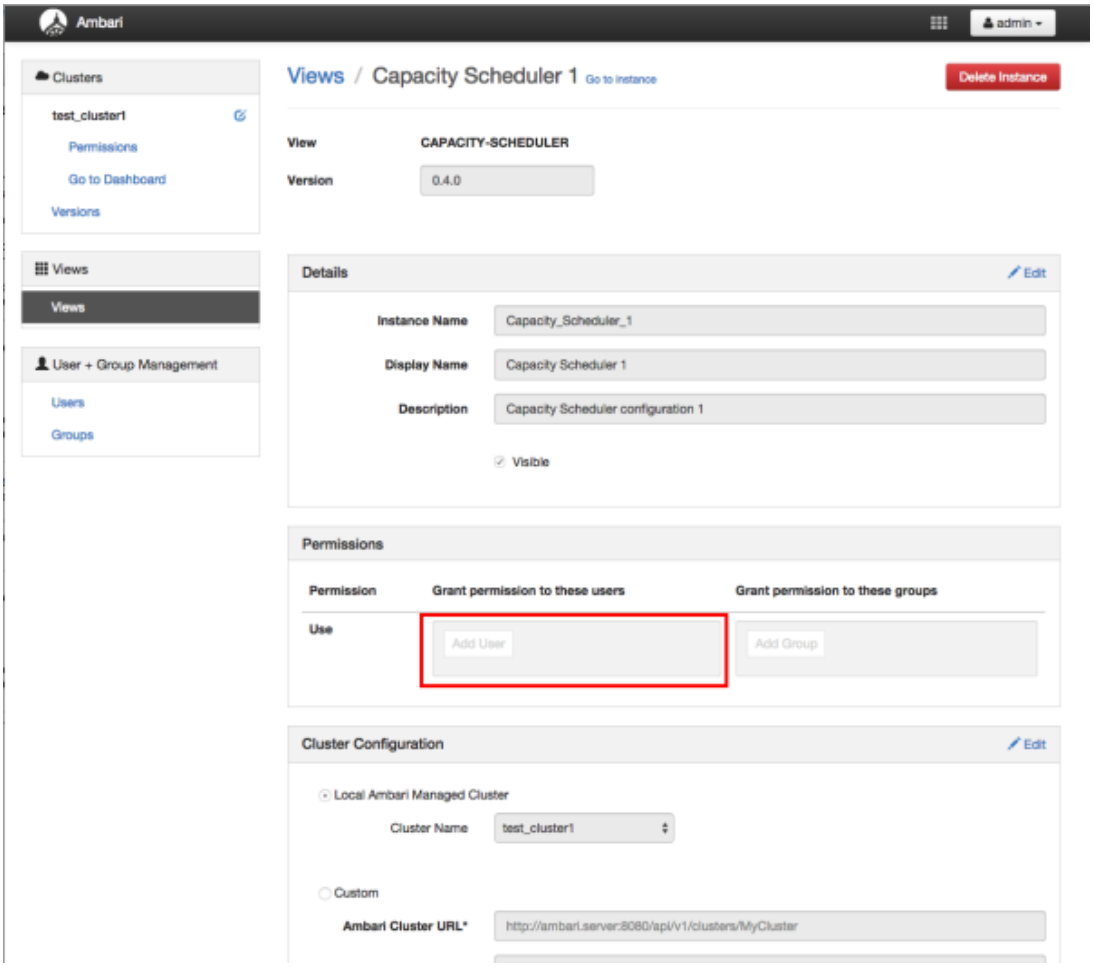
7. The Capacity Scheduler View instance is created, and the configuration page for the instance appears.



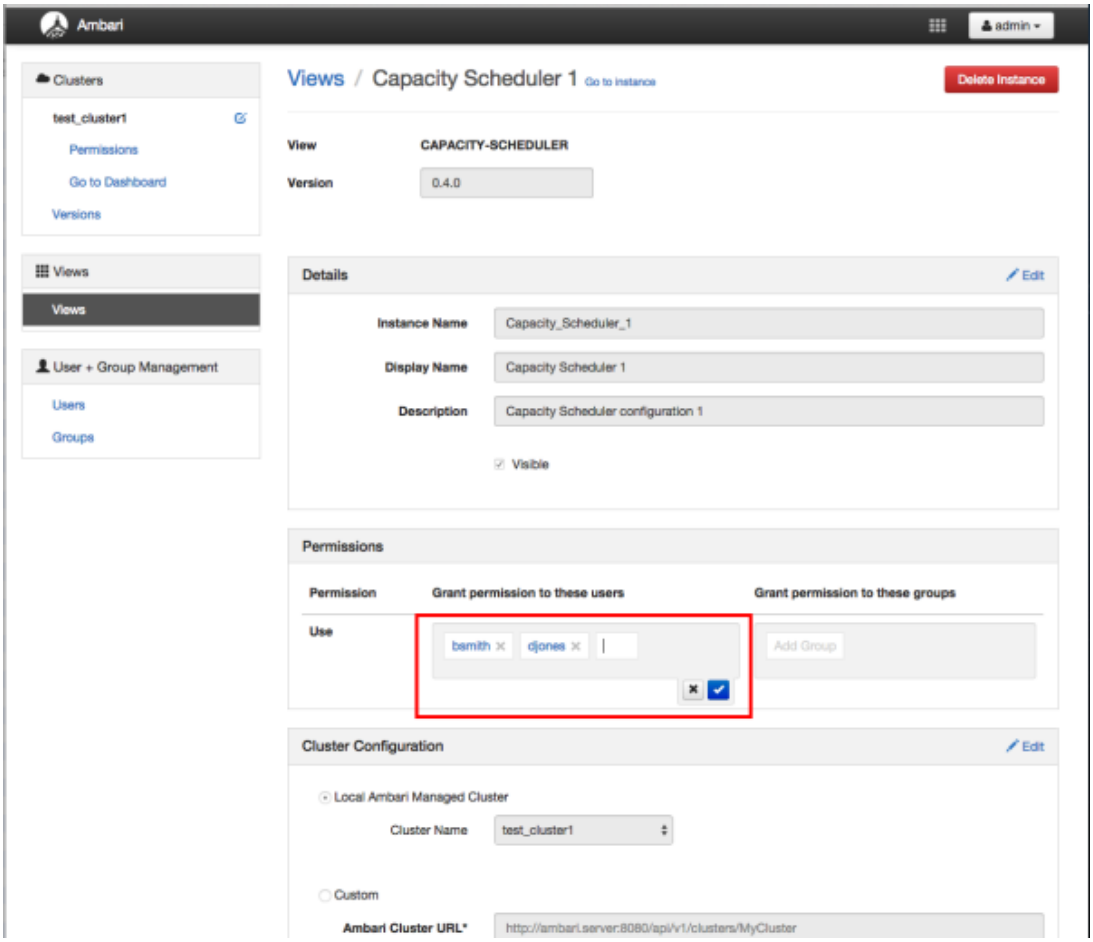
6.2.1. User Permissions for YARN Queue Manager View

Use the following procedure to add users and groups to a YARN Queue Manager view instance.

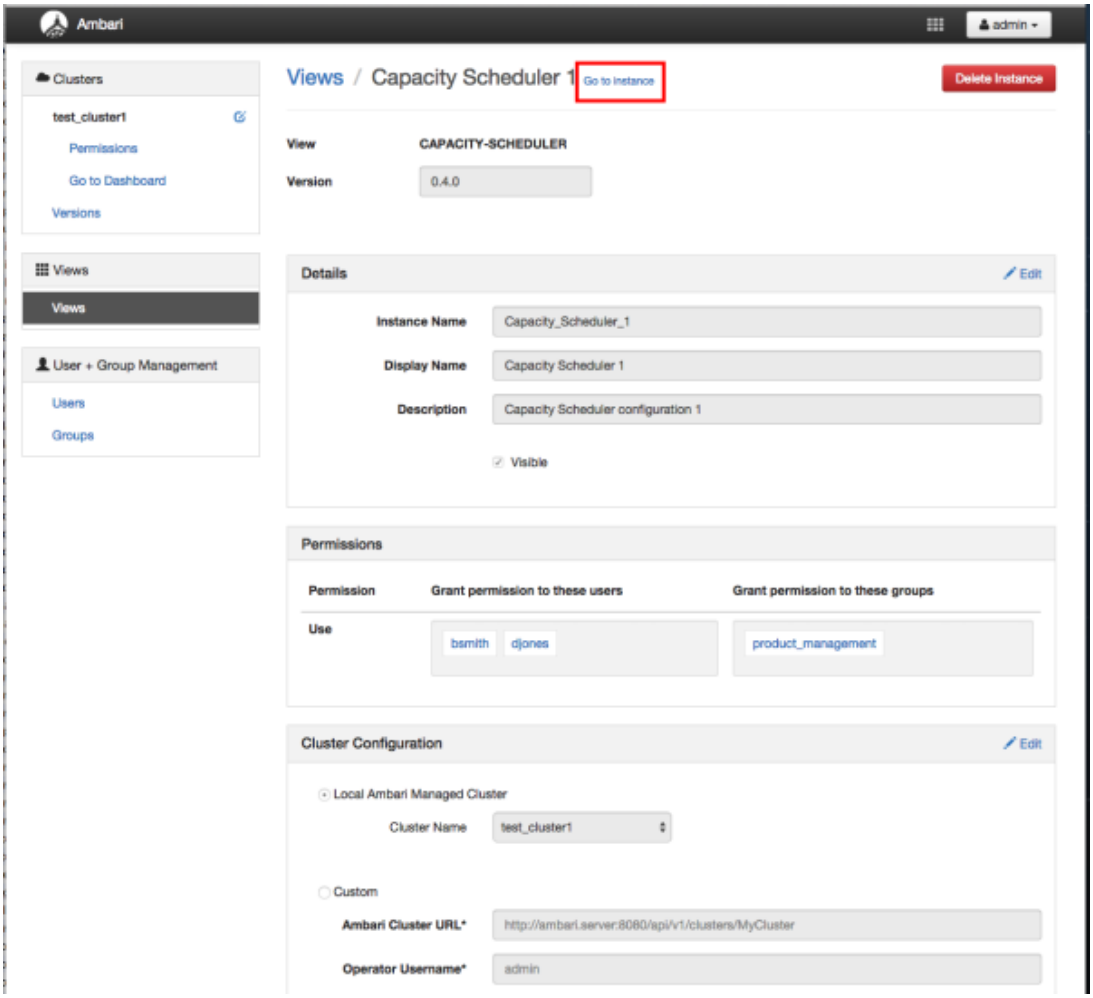
1. On the Capacity Scheduler view instance configuration page, click the box labeled Add User in the Permissions box.



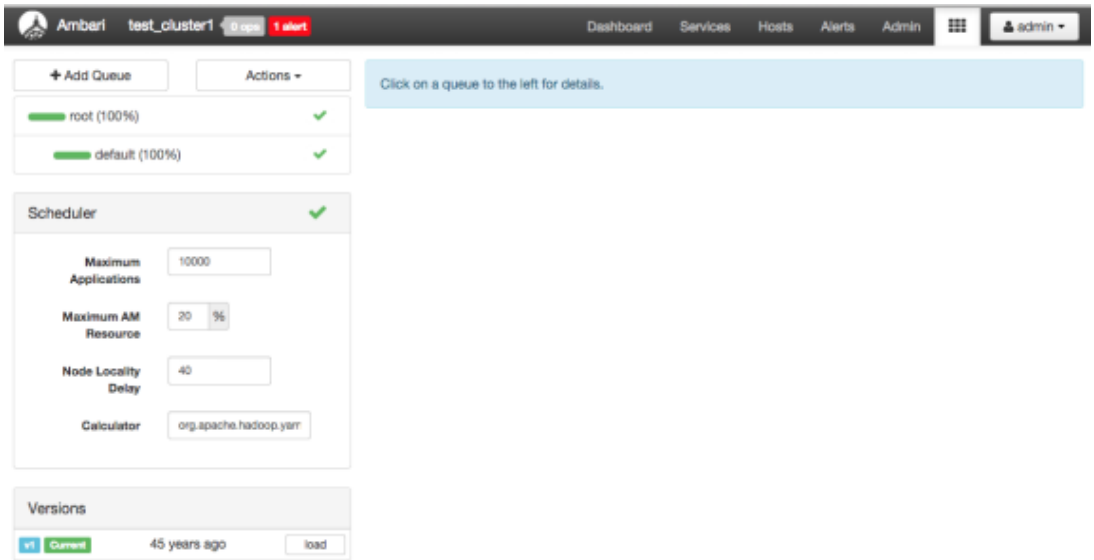
2. Enter user names in the Use box, then click the blue check mark to add the users. You can use the same method to add groups in the Add Group box.



3. After you have finished adding users and groups, click **Go to instance** at the top of the page to open the YARN Queue Manager view instance.



4. The Capacity Scheduler view instance page appears.



6.3. Using the YARN Queue Manager View

The YARN Queue Manager View is designed to help hadoop operators configure workload management policies for YARN. In the YARN Queue Manager View, operators can create hierarchical queues and tune configurations for each queue to define an overall workload management policy for the cluster.

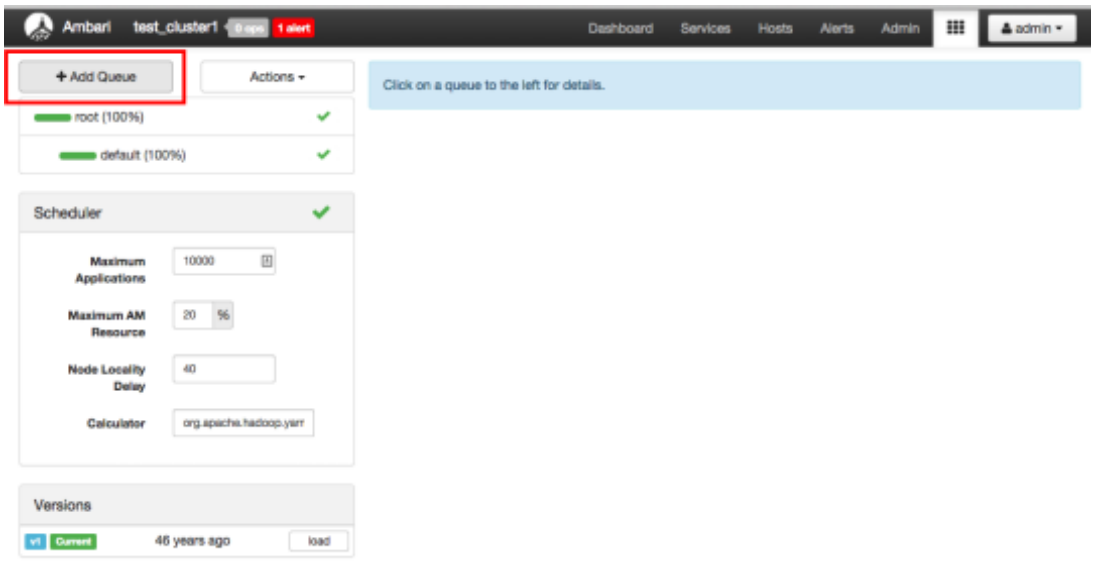
6.3.1. Setting up Queues

Use the following steps to set up Capacity Scheduler queues on a view instance.

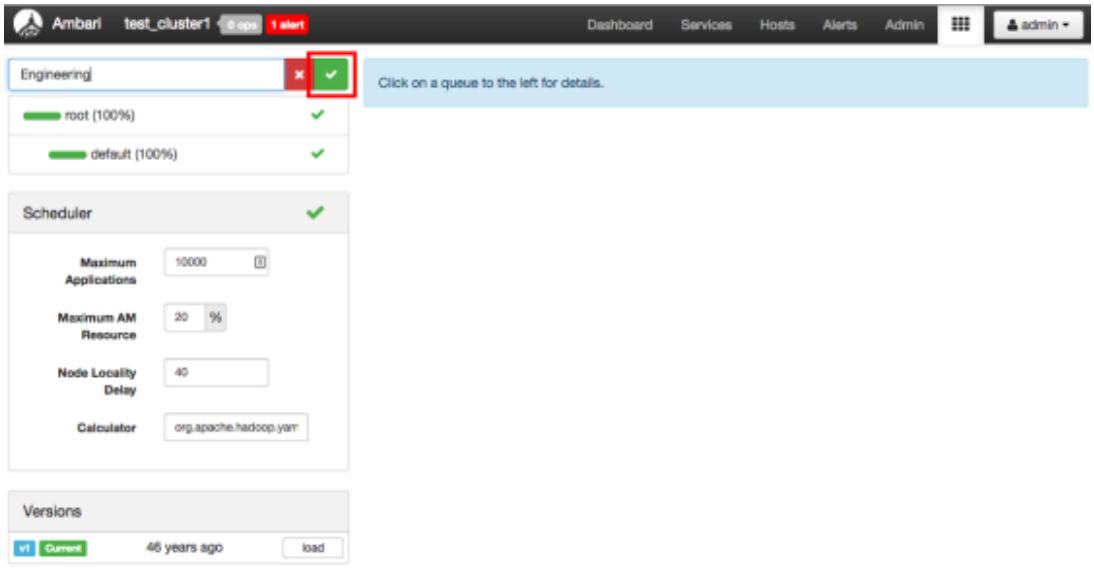
1. On the YARN Queue Manager view instance configuration page, click **Add Queue**. The queue will be added under the top level, or "root" queue. A "default" queue already exists under the root queue.

Note

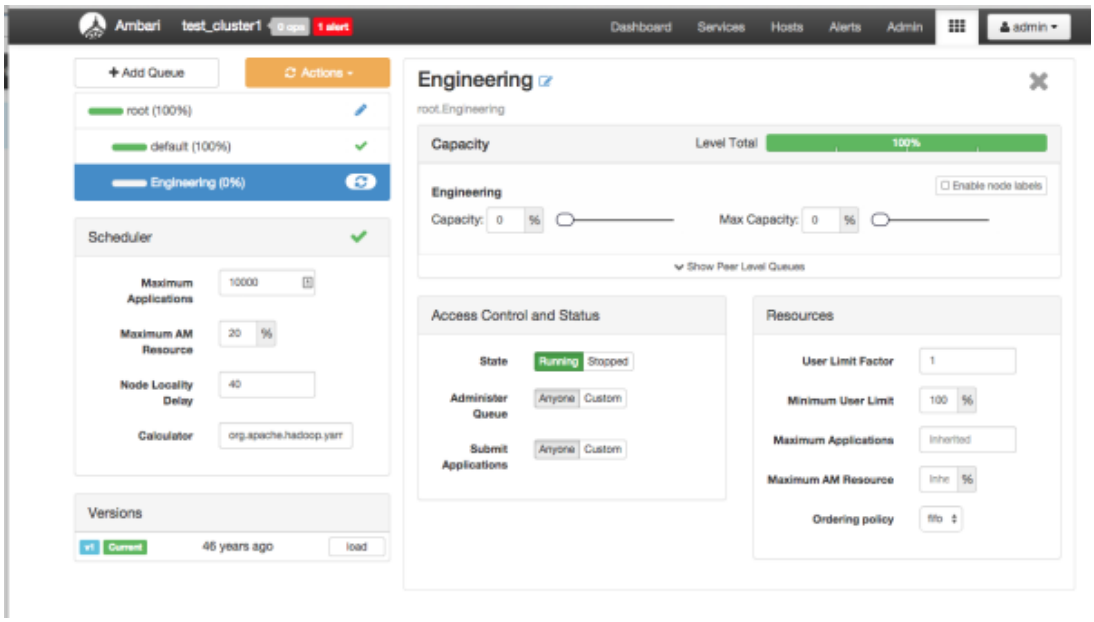
To return to a previously created YARN Queue Manager view instance, click **Views** on the Manage Ambari page, then click **CAPACITY-SCHEDULER**. Click the applicable YARN Queue Manager view instance, then click **Go to instance** at the top of the page.



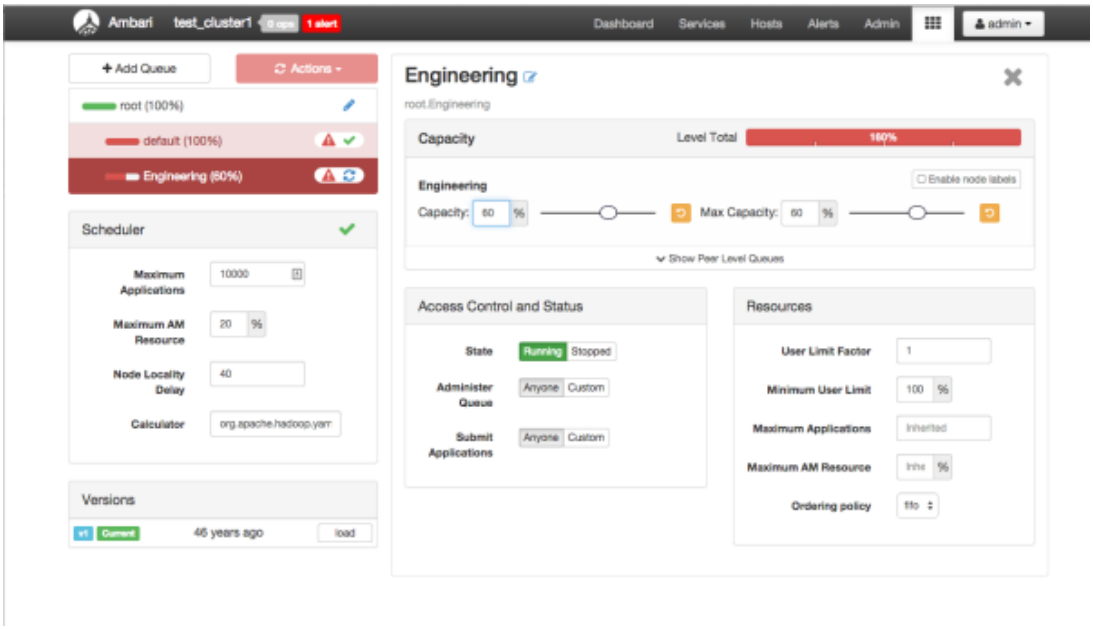
2. Type in a name for the new queue, then click the green check mark to create the queue. In the following example, we're creating the "Engineering" queue.



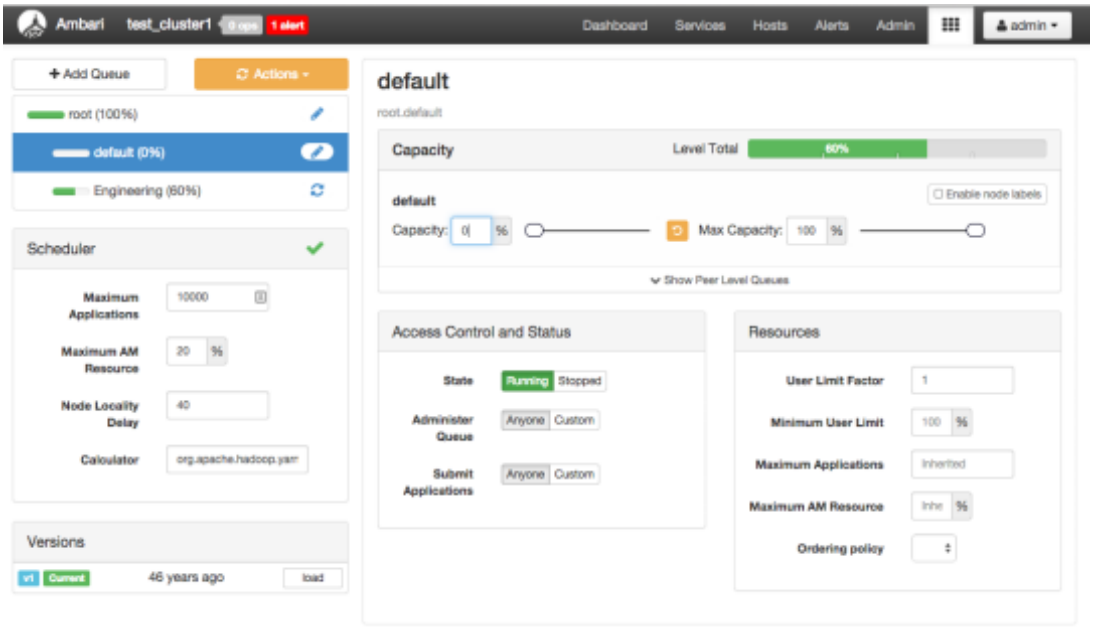
3. The "Engineering" queue is added, and its configuration page appears.



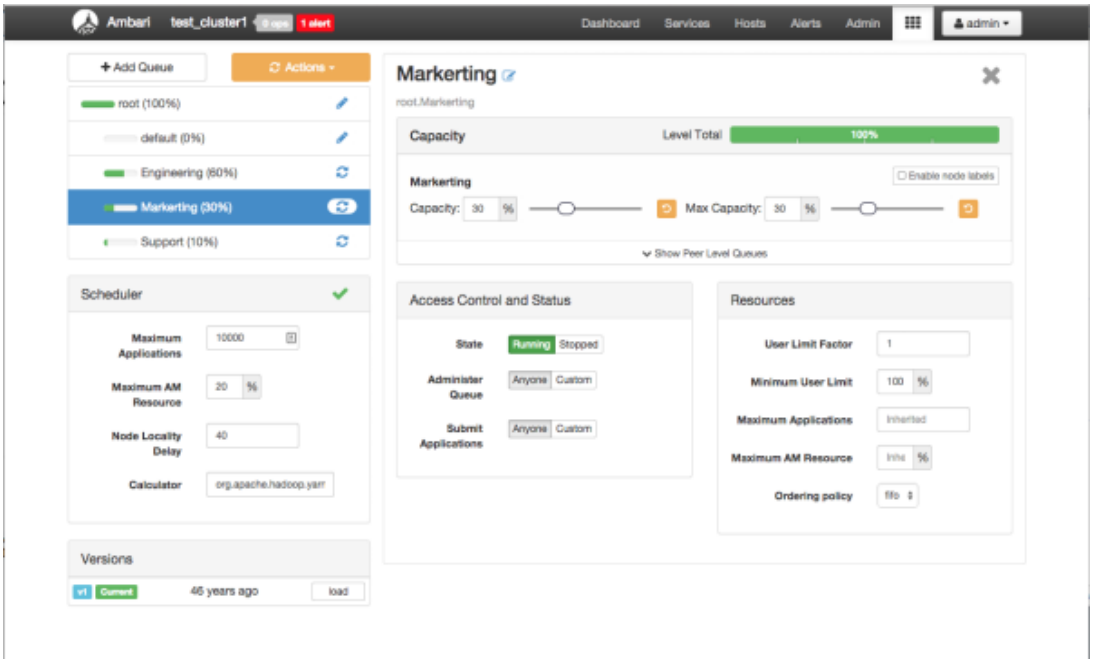
4. The sum of queue capacities at any level in the YARN Queue Manager configuration must total 100%. Here the default queue is already set to 100%. Therefore, if we try to set the "Engineering" queue capacity to 60%, error messages appear warning that the total at this level is 160%.



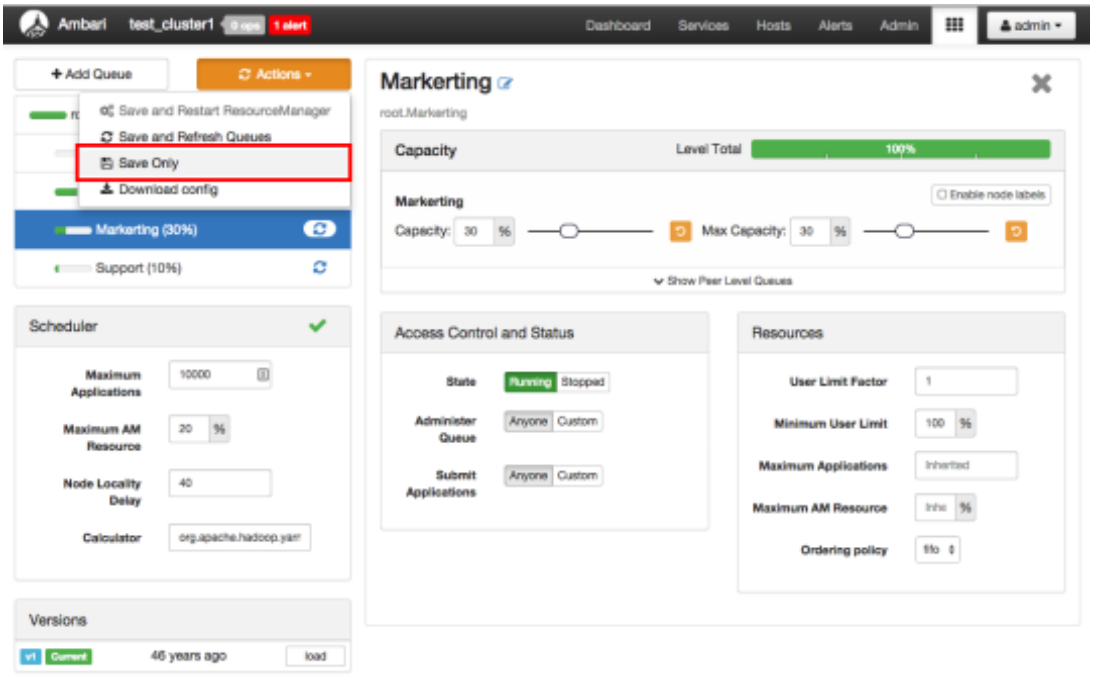
5. If we click the "default" queue and set its capacity to 0%, the error messages no longer appear, and the Level Total bar at the top of the page lists the total queue capacity at this level as 60%.



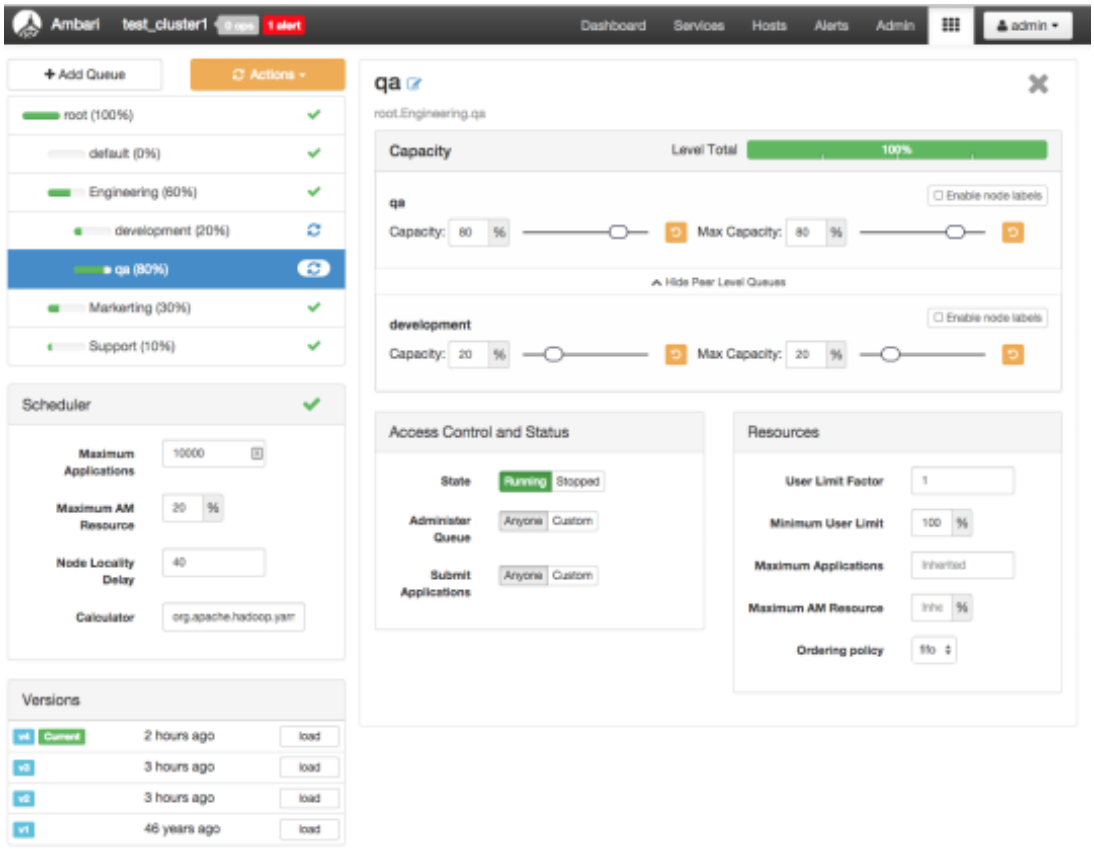
6. To add more queues at the root level, click the **root** queue, then click **Add Queue**. In the following example, we have added a "Support" queue set to 10% of the level capacity, and a "Marketing" queue set to 30%. The root-level queue capacities now total 100%.



7. To save your configuration, click **Actions > Save Only**. On the **Notes** pop-up, enter an optional description of your changes, then click **Save**. Each version is retained and listed in the Versions box.



8. To build a queue hierarchy, click a top level queue, then click Add Queue. In the following example, the "qa" and "development" queues have been added under the "Engineering" queue.



6.3.2. Configuring Queues

To configure a queue, click the queue name, then set the following queue parameters:

Note

Hold the cursor over a parameter name to display a description of the parameter.

Capacity

- Capacity – The percentage of cluster resources available to the queue. For a sub-queue, the percentage of parent queue resources.
- Max Capacity – The maximum percentage of cluster resources available to the queue. Setting this value tends to restrict elasticity, as the queue will be unable to utilize idle cluster resources beyond this setting.
- Enable Node Labels – Select this check box to enable node labels for the queue.

Access Control and Status

- State – Running is the default state. Setting this to Stopped lets you gracefully drain the queue of jobs (for example, before deleting a queue).

- Administer Queue – Click **Custom** to restrict administration of the queue to specific users and groups.

- Submit Applications – Click **Custom** to restrict the ability to run applications in the queue to specific users and groups.

Resources

- User Limit Factor – The default value of "1" means that any single user in the queue can at maximum only occupy the queue's configured capacity. This prevents users in a single queue from monopolizing resources across all queues in a cluster. Setting the value to "2" would restrict the queue's users to twice the queue's configured capacity. Setting it to a value of 0.5 would restrict any user from using resources beyond half of the queue capacity.

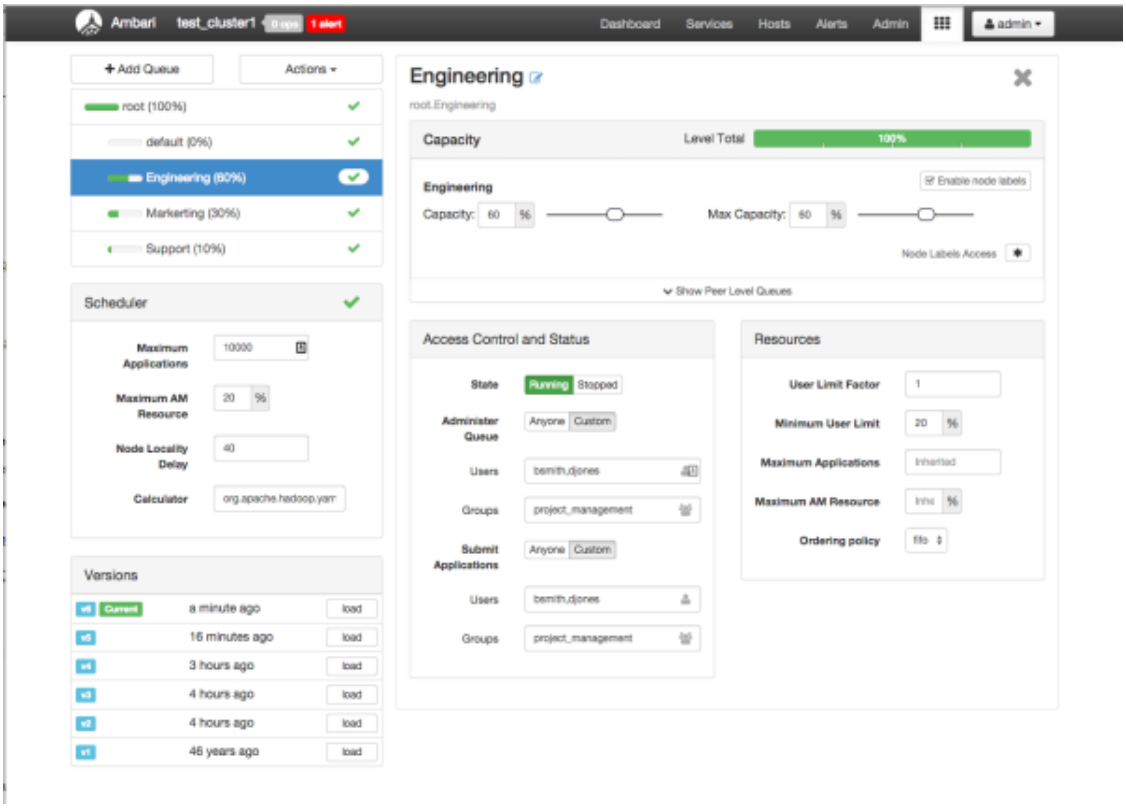
- Minimum User Limit – This property can be used to set the minimum percentage of resources allocated to each queue user. For example, to enable equal sharing of the queue capacity among five users, you would set this property to 20%.

- Maximum Applications – This setting enables you to override the Scheduler Maximum Applications setting (described in Configuring Cluster Scheduler Settings). The default setting is Inherited (no override).

- Maximum AM Resource – This setting enables you to override the Scheduler Maximum AM Resource setting (described in Configuring Cluster Scheduler Settings). The default setting is Inherited (no override).

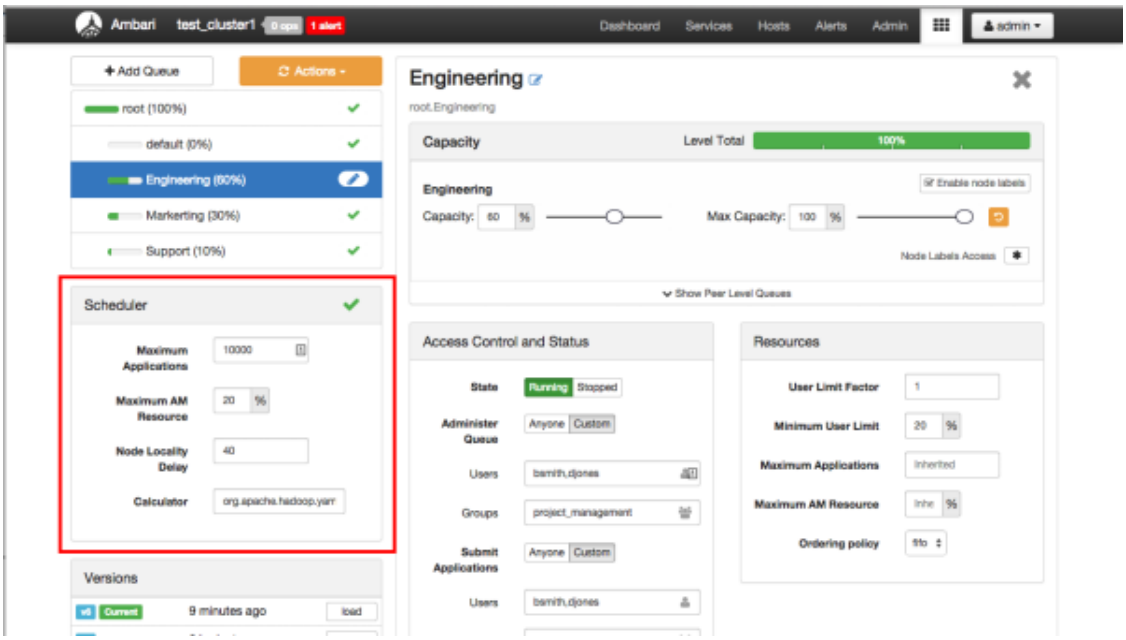
- Ordering Policy – You can specify FIFO (First In, First Out) or fair (Fair Scheduler: applications get a fair share of capacity regardless of the order in which they were submitted).

The following image shows the example "Engineering" queue with these settings specified:



6.3.3. Configuring Cluster Scheduler Settings

You can use the Scheduler box to set global capacity scheduler settings that apply to all queues.



The following Scheduler global parameters are available:

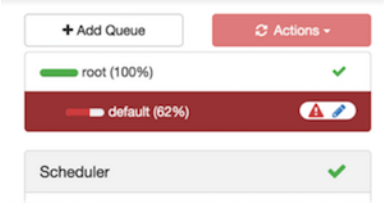
- Maximum Applications – To avoid system-thrash due to an unmanageable load – caused either by malicious users, or accidentally – the Capacity Scheduler enables you to place a static, configurable limit on the total number of concurrently active (both running and pending) applications at any one time. This property is used to set this limit, with a default value of 10,000.
- Maximum AM Resource – The limit for running applications in any specific queue is a fraction of this total limit, proportional to its capacity. This is a hard limit, which means that once this limit is reached for a queue, any new applications submitted to that queue will be rejected, and clients will have to wait and retry later.
- Node Locality Delay – The number of missed scheduling cycles after which the scheduler attempts to schedule rack-local containers.
- Calculator – The method by which the scheduler calculates resource capacity across resource types.

6.3.4. Applying the Configuration Changes

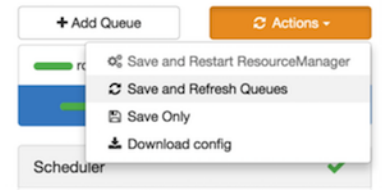
You can use the Actions menu to apply configuration changes made to the queue hierarchy.

Depending on the configuration changes made, the Actions menu will guide you to the options available to apply the changes.

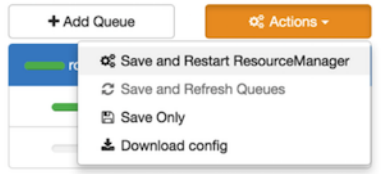
For changes that are not valid and cannot be applied, the **Actions** button will turn red, and the menu will not appear.



For configuration changes that can be applied dynamically (without restarting the YARN ResourceManager), the Actions Menu will guide you to **Save and Refresh Queues**.



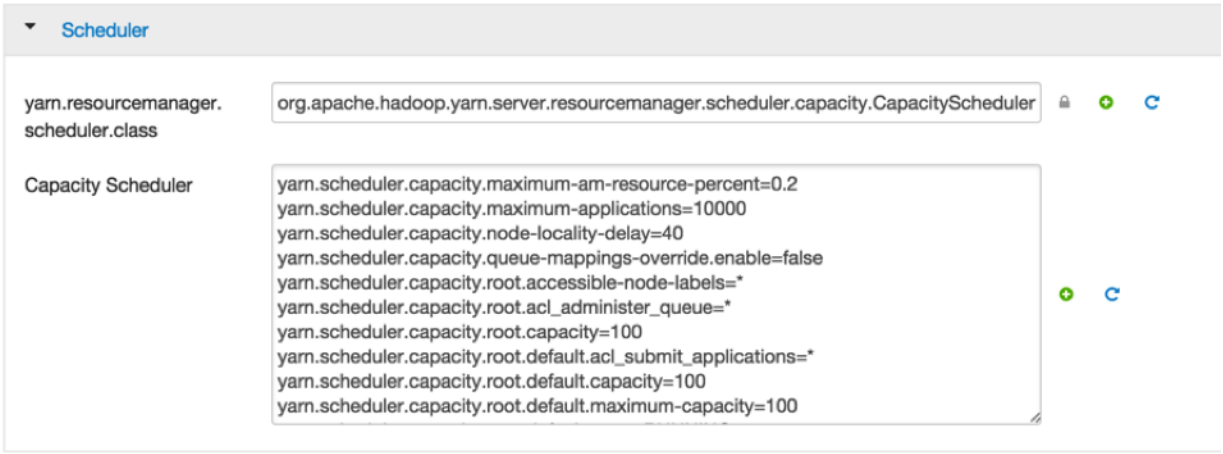
For configuration changes that require a restart of the YARN ResourceManager, the Actions Menu will guide you to **Save and Restart ResourceManager**.



6.4. Troubleshooting

If you encounter an issue where the configurations cannot be applied from the View, you should go to the local Ambari Server instance managing the cluster and directly edit the Capacity Scheduler configuration from the YARN configuration page.

In the local Ambari instance, navigate to **Services > YARN**, then select the **Configs** tab. On the **Advanced** tab, expand the Scheduler section.



Here you will be able to edit the underlying configurations for the YARN Queue Manager and fix any issues you may encounter.

7. Using the Files View

The **Files View** provides a convenient way to access HDFS through a web-based interface. The Files View supports:

- Moving Files/ Folders within HDFS
- Copying Files/Folders within HDFS
- Uploading files from a local system
- Modifying permissions of files and folders

This document provides information on how to configure a Files View instance and use the File browser UI to access HDFS.

- Configuring Your Cluster
- Creating and Configuring a Files View Instance
- Troubleshooting

Important

It is critical that you prepare your Ambari Server for hosting views. It is strongly recommended you increase the amount of memory available to your Ambari Server, and that you run additional "standalone" Ambari Servers to host the views. See Preparing Ambari Server for Views and Running Ambari Server Standalone for more information.

7.1. Configuring Your Cluster

For the Files View to access HDFS, the Ambari Server daemon hosting the view needs to act as the proxy user for HDFS. This allows Ambari to submit requests to HDFS on behalf of the users using the Files View.

Note

If you are running views in an operational Ambari server (one that is operating the cluster) Ambari does this setup by default. You should verify that the setup described in the following subsections has been completed. If you are running views on a standalone server, you must setup proxy user settings manually, using the following instructions.

To set up an HDFS proxy user for the Ambari Server daemon account, you need to configure the proxy user in the HDFS configuration. This configuration is determined by the account name the **ambari-server** daemon is running as. For example, if your ambari-server is running as **root**, you set up an HDFS proxy user for **root** with the following:

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

2. Under the **Advanced** tab, navigate to the **Custom core-site** section.

3. Click **Add Property**... to add the following custom properties:

```
hadoop.proxyuser.root.groups="users"
hadoop.proxyuser.root.hosts=ambari-server.hostname
```

Notice the **ambari-server** daemon account name **root** is part of the property name. Be sure to modify this property name for the account name you are running the ambari-server as. For example, if you were running **ambari-server** daemon under an account name of **ambariusr**, you would use the following properties instead:

```
hadoop.proxyuser.ambariusr.groups="users"
hadoop.proxyuser.ambariusr.hosts=ambari-server.hostname
```

Similarly, if you have configured Ambari Server for Kerberos, be sure to modify this property name for the primary Kerberos principal user. For example, if ambari-server is setup for Kerberos using principal **ambari-server@EXAMPLE.COM**, you would use the following properties instead:

```
hadoop.proxyuser.ambari-server.groups="users"
hadoop.proxyuser.ambari-server.hosts=ambari-server.hostname
```

4. Save the configuration change and restart the required components as indicated by Ambari.

7.2. Creating and Configuring a Files View Instance

1. Browse to the Ambari Administration interface.

2. Click Views, expand the **Files View**, and click **Create Instance**.

3. Enter the following View instance **Details**:

Property	Description	Value
Instance Name	This is the Files view instance name. This value should be unique for all Files view instances you create. This value cannot contain spaces and is required.	FILES_1
Display Name	This is the name of the view link displayed to the user in Ambari Web.	MyFiles
Description	This is the description of the view displayed to the user in Ambari Web.	Browse HDFS files and directories.
Visible	This checkbox determines whether the view is displayed to users in Ambari Web.	Visible or Not Visible

4. The **Settings** and **Cluster Configuration** options depend on a few cluster & deployment factors in your environment:

   - Is your cluster Kerberos-enabled?
   - Is NameNode HA configured?

- Is your **Files View** instance being configured in an **Operational** Ambari Server or a **Standalone** Ambari Server?

  Refer to the following table on the instructions to complete the **Files View** configuration:

Kerberos Enabled	NameNode HA Enabled	Operational Ambari Server see note #1:	Standalone Ambari Server see note #2:
No	No	Settings: defaults Cluster Configuration: Local	Settings: defaults Cluster Configuration: Custom
No	Yes		
Yes	No	Settings : Kerberos Cluster Configuration : Custom	
Yes	Yes	Settings: Kerberos Cluster Configuration: Custom	

Note

#1: The Local Ambari Managed Cluster Configuration option is enabled in the Ambari Administration Interface only if you are managing a cluster in an Operational Ambari Server.

Note

#2: See Running Ambari Standalone for more information.

7.2.1. Kerberos Settings

You must first set up Kerberos for Ambari by configuring the Ambari Server daemon with a Kerberos principal and keytab. Refer to Configuring Views for Kerberos for instructions. After you have set up Kerberos for Ambari, in the Settings section of the Files View, enter the following:

Property	Description	Example Value
WebHDFS Username	This is the username the view will access HDFS as. Leave this default value intact to represent the authenticated view user.	${username}
WebHDFS Authorization	This is the semicolon-separated authentication configuration for WebHDFS access.	auth=KERBEROS;proxyuser=ambari-server **Note** This property is only needed if the view is Custom Configured or Ambari Server is Kerberized before 2.4.0.

Note

With a Kerberos setup, the proxy user setting should be the primary value of the Kerberos principal for Ambari Server. For example, if you configured Ambari Server for Kerberos principal **ambari-server@EXAMPLE.COM**, this value

would be **ambari-server**. Refer to Configuring Views for Kerberos for more information..

7.2.2. Cluster Configuration: Local

The **Local Ambari Managed Cluster Configuration** option is enabled in the Ambari Administration Interface if you are managing a cluster with Ambari. When enabled, you can choose this option and Ambari will automatically configure the view based on how the cluster is configured.

When you configure the view using the Local option, the Files View will communicate with HDFS based on the **fs.defaultFS** property (for example: hdfs://namenode:8020). The View will also determine if NameNode HA is configured and adjust accordingly.

7.2.3. Cluster Configuration: Custom

These properties are required if using Custom configuration.

Required Properties	Description	Example Value
WebHDFS FileSystem URI	The WebHDFS FileSystem URI in the format `webhdfs://<HOST>:<HTTP_PORT>`	webhdfs://namenode:50070

These properties are required if your cluster is configured for NameNode HA.

Property	Description	Example Value
Logical name of the NameNode cluster	Comma-separated list of nameservices.	hdfs-site/dfs.nameservices For example: nameservice
List of NameNodes	Comma-separated list of NameNodes for a given nameservice.	hdfs-site/dfs.ha.namenodes For example: namenode1,namenode2
First NameNode RPC Address	RPC address for first name node.	hdfs-site/dfs.namenode.rpc-address.[nameservice].[namenode1]
Second NameNode RPC Address	RPC address for second NameNode.	hdfs-site/dfs.namenode.rpc-address.[nameservice].[namenode2]
First NameNode HTTP (WebHDFS) Address	WebHDFS address for first NameNode.	hdfs-site/dfs.namenode.http-address.[nameservice].[namenode1]
Second NameNode HTTP (WebHDFS) Address	WebHDFS address for second NameNode.	hdfs-site/dfs.namenode.http-address.[nameservice].[namenode2]
Failover Proxy Provider	The Java class that HDFS clients use to contact the Active NameNode.	hdfs-site/dfs.client.failover.proxy.provider.[nameservice]

7.2.4. Troubleshooting

Error	Solution
500 Usernames not matched: name=root != expected=ambari-server	If your cluster is configured for Kerberos, double-check WebHDFS Authorization setting and confirm the “proxyuser=” part of the string is set to the Ambari Server principal name.

Error	Solution
	For example: auth=KERBEROS;proxyuser=ambari-server Refer to Kerberos Settings.
500 User: ambari-server is not allowed to impersonate admin	HDFS has not been configured for Ambari as a proxy user. Refer to Configuring Your Cluster.
500 SIMPLE authentication is not enabled. Available:[TOKEN, KERBEROS]	If your cluster is configured for Kerberos, you cannot use the Local Cluster Configuration option. You must use the Custom Cluster Configuration option and enter the WebHDFS FileSystem URI. For example: webhdfs://namenode:50070 Refer to Cluster Configuration: Custom

8. Using the Falcon View

Apache Falcon solves enterprise challenges related to Hadoop data replication, business continuity, and lineage tracing by deploying a framework for data management and processing. The Falcon framework can also leverage other HDP components, such as Apache Pig, Apache Hadoop Distributed File System (HDFS), Apache Sqoop, Apache Hive, Apache Spark, and Apache Oozie. Falcon enables this simplified management by providing a framework to define and manage backup, replication, and data transfer.

Hadoop administrators can use the **Falcon View** to centrally define, schedule, and monitor data management policies. **Falcon** uses those definitions to auto-generate workflows in Apache Oozie.

This chapter describes the following:

- Section 8.1, "Configuring Your Cluster" [36]
- Section 8.2, "Installing and Configuring the Falcon View" [37]
- Section 8.3, "Accessing the Falcon Documentation" [41]

8.1. Configuring Your Cluster

For the Falcon View to access HDFS, the Ambari Server daemon hosting the view needs to act as the proxy user for HDFS. This allows Ambari to submit requests to HDFS on behalf of the users using the Falcon View. This is critical since the Falcon View stores metadata about the user Falcon entity definitions. This also means users who access the Falcon View must have a user directory setup in HDFS.

8.1.1. Setup HDFS Proxy User

To set up an HDFS proxy user for the Ambari Server daemon account, you need to configure the proxy user in the HDFS configuration. This configuration is determined by the account name the **ambari-server** daemon is running as. For example, if your ambari-server is running as **root**, you set up an HDFS proxy user for **root** with the following:

1. In Ambari Web, browse to **Services > HDFS > Configs**.
2. Under the **Advanced** tab, navigate to the **Custom core-site** section.
3. Click **Add Property...** to add the following custom properties:

```
hadoop.proxyuser.root.groups="users"
hadoop.proxyuser.root.hosts=ambari-server.hostname
```

Notice the **ambari-server** daemon account name root is part of the property name. Be sure to modify this property name for the account name you are running the ambari-server as. For example, if you were running **ambari-server** daemon under an account name of **ambariusr**, you would use the following properties instead:

```
hadoop.proxyuser.ambariusr.groups="users"
hadoop.proxyuser.ambariusr.hosts=ambari-server.hostname
```

Similarly, if you have configured Ambari Server for Kerberos, be sure to modify this property name for the **primary Kerberos principal** user. For example, if ambari-server is setup for Kerberos using principal **ambari-server@EXAMPLE.COM**, you would use the following properties instead:

```
hadoop.proxyuser.ambari-server.groups="users"
hadoop.proxyuser.ambari-server.hosts=ambari-server.hostname
```

4. Save the configuration change and restart the required components as indicated by Ambari.

8.1.2. Setup HDFS User Directory

The Falcon View stores user metadata in HDFS. By default, the location in HDFS for this metadata is `/user/${username}` where `${username}` is the username of the currently logged in user that is accessing the Falcon View.

Important

Since many users leverage the default Ambari admin user for getting started with Ambari, the `/user/admin` folder needs to be created in HDFS. Therefore, be sure to create the admin user directory in HDFS using these instructions prior to using the view.

To create user directories in HDFS, do the following for each user you plan to have use the Hive View.

1. Connect to a host in the cluster that includes the HDFS client.
2. Switch to the hdfs system account user.

   ```
   su - hdfs
   ```

3. Using the HDFS client, make an HDFS directory for the user. For example, if your username is admin, you would create the following directory.

   ```
   hadoop fs -mkdir /user/admin
   ```

4. Set the ownership on the newly created directory. For example, if your username is admin, you would make that user the directory owner.

   ```
   hadoop fs -chown admin:hadoop /user/admin
   ```

8.2. Installing and Configuring the Falcon View

You must manually copy the `.jar` file for the Falcon View, then configure Ambari to access the View. You can install the Falcon View in a secure or an unsecure cluster. If using a secure cluster, Ambari and Falcon must be properly configured with Kerberos.

Prerequisites

- Apache Falcon must have been installed and configured, and be deployed in Ambari.

  For an Ambari-managed installation, Falcon is included as a default service. To deploy the Falcon service, refer to Adding a Service to your Hadoop cluster.

For manual (non-Ambari) installation and setup of Falcon, refer to Installing Apache Falcon, then Adding a Service to your Hadoop cluster.

- The users and groups for Falcon must exist in Ambari prior to installing the Falcon View.

  Refer to Managing Users and Groups.

- Falcon must have been configured as a proxy super user in the `oozie-site` properties and in the HDFS `core-site` properties.

Steps

1. Copy the Falcon View `falcon-ambari-view.jar` file from the Falcon server `/webapp` directory to the Ambari server `/views` directory.

   - If the Falcon and Ambari servers are on the same host, use the copy command:

     ```
     cp /usr/hdp/current/falcon-server/server/webapp/falcon-ambari-view.jar /
     var/lib/ambari-server/resources/views/
     ```

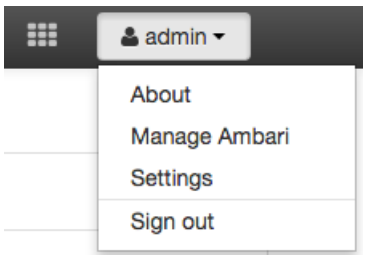
   - If the Falcon server is on a remote host, use the secure copy command for your operating system.

     A key pair might be required. See your operating system documentation for more information about remote copies.

2. Restart the Ambari server.

   ```
   [root@DataMovementDocs-1 ~]# ambari-server restart
   ```

3. In Ambari, navigate to ***user_name*** > **Manage Ambari**.



4. Under Deploy Views, click **Views**, then click **Falcon** > **Create Instance** in the Views list.

5. Provide the required Details information.

   Instance Name: 250 characters, no spaces, no special characters
   Display Name: 250 characters, including spaces; no special characters; can be the same as the Instance Name
   Description: 140 characters max, including spaces; special characters allowed

Note

If you enter more than the allowed number of characters, you might see the error message `Cannot create instance: Server Error`.

6. Select a cluster configuration.

   The Local and Remote fields populate with the names of available clusters. The authentication type for the cluster is automatically recognized.

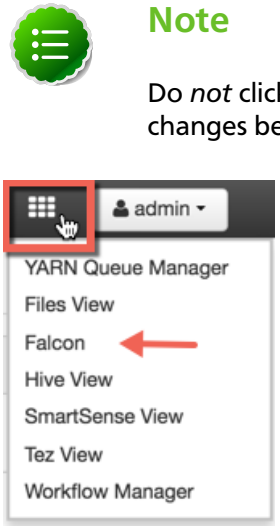
   To use a custom cluster location, enter the Falcon service URI and authentication type of **`simple`** or **`kerberos`**.

7. Click **Save**.

   The Permissions section displays at the bottom of the Views page.

8. (Optional) Set the permissions for access to the view.

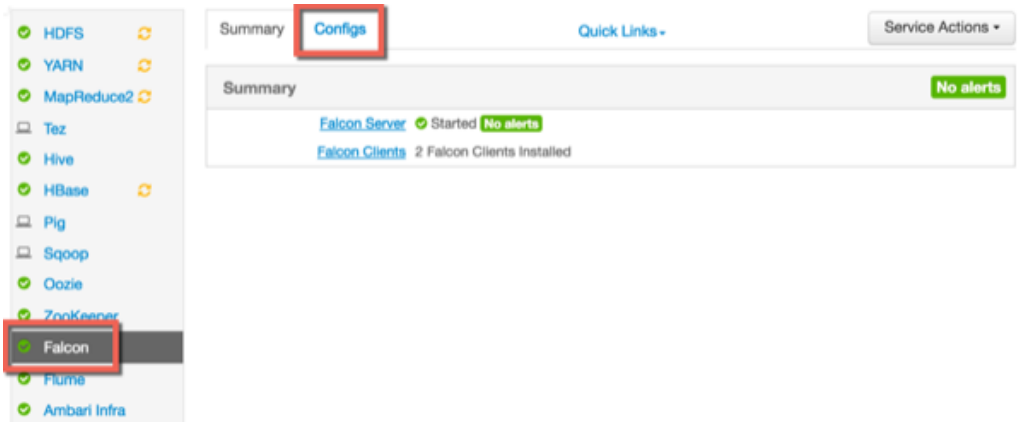
9. Hover over the **Views** icon to verify that your Falcon View is available in the menu.



Note

Do *not* click on the Falcon link yet. You must make additional configuration changes before you can access the Falcon View.

10. Click the Ambari icon to return to the Dashboard window, then click the **Falcon** service and the **Configs** tab.



11. Scroll to the **Falcon startup.properties** section, locate the ***.application.services** field, and enter the following services immediately above the line `org.apache.falcon.metadata.MetadataMappingService`:

```
org.apache.falcon.service.GroupsService,\

org.apache.falcon.service.ProxyUserService,\
```

12. Add the proxy user for hosts and groups in the **Custom falcon-runtime.properties** section.

    The proxy user is the user that the Falcon process runs as, typically *Falcon*.

    a. Click **Add Property**.

    b. Add the following key/value pairs.

    Substitute *#USER#* with the proxy user configured for the Ambari server.

    - Key=***.falcon.service.ProxyUserService.proxyuser.#USER#.hosts**, Value=*****

      These are the hosts from which *#USER#* can impersonate other users.

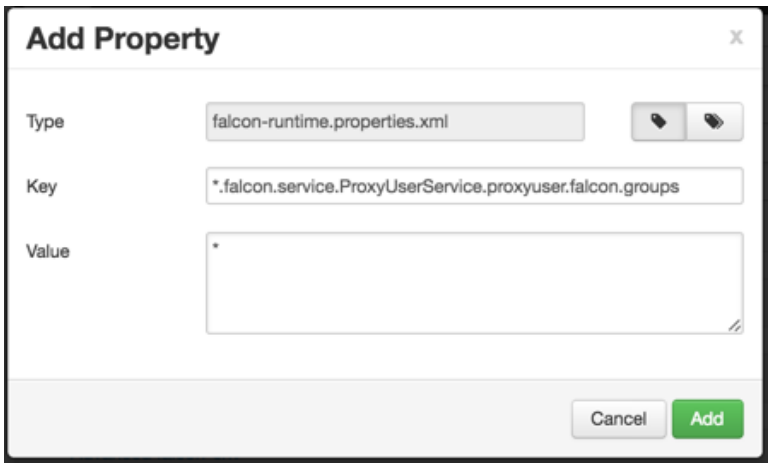
    - Key=***.falcon.service.ProxyUserService.proxyuser.#USER#.groups**, Value=*****

      These are the groups that the users being impersonated must belong to.

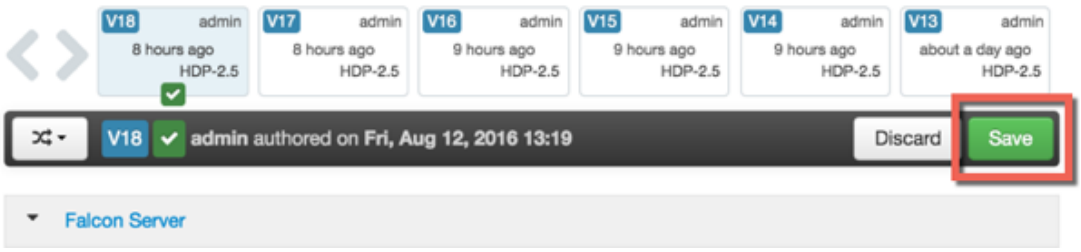
Example 8.1. Substitute *#USER#*

In the key/value pairs above, if the *#USER#* is "*falcon*", enter `*.falcon.service.ProxyUserService.proxyuser.falcon.hosts`.

The wildcard value=* (asterisk) is used to allow impersonation from any host or of any user. If you don't use the wildcard character, enter the appropriate host or group values.



13. Click **Save** on the information bar at the top of the Configs page.



If you try to leave the page without clicking Save, you see a Warning message. Click Save in the Warning dialog box.

A `Restart Required` message displays at the top of the Falcon Configs page.

14. Click **Restart** > **Restart All Affected** to restart the Falcon services.

15. When the restart completes, verify that you can access the Falcon View by clicking **Falcon** in the Views menu.

8.3. Accessing the Falcon Documentation

You can access the Falcon documentation in the Data Movement and Integration guide on the Hortonworks documentation website.

9. Using the Hive View

Hive is a data warehouse infrastructure built on top of Hadoop. It provides tools to enable data ETL, a mechanism to put structures on the data, and the capability to query and analyze large data sets that are stored in Hadoop. The **Hive View** is designed to help you author, execute, understand, and debug Hive queries.

This chapter explains:

- Upgrading Your Hive View
- Configuring Your Cluster
- Creating the Hive View Instance
- Using the Hive View
- Upload Table Tab
- Troubleshooting

Important

The Tez View integrates with the Hive View. Please install the Tez View when you install the Hive View. See Using the Tez View for more information.

Important

It is critical that you prepare your Ambari Server for hosting views. It is strongly recommended you increase the amount of memory available to your Ambari Server, and that you run additional "standalone" Ambari Servers to host the views. See Preparing Ambari Server for Views and Running Ambari Server Standalone for more information.

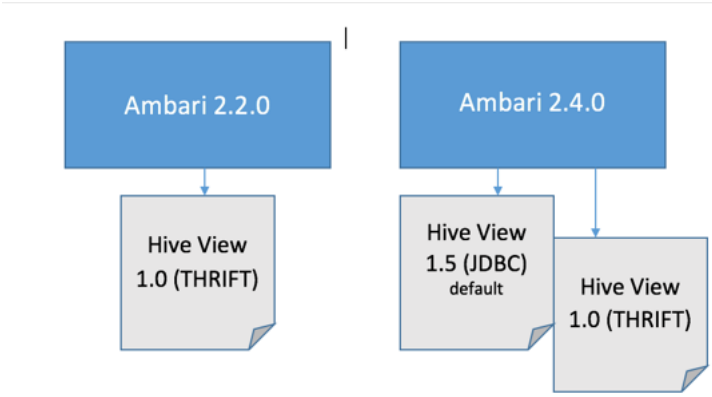
Hive Views

With the release of Apache Ambari 2.4.0, two Hive views install as part of your Hortonworks Data Platform distribution:

Hive View 1.0 - which works with Thift Java API

Hive View 1.5 - which works with the JDBC client

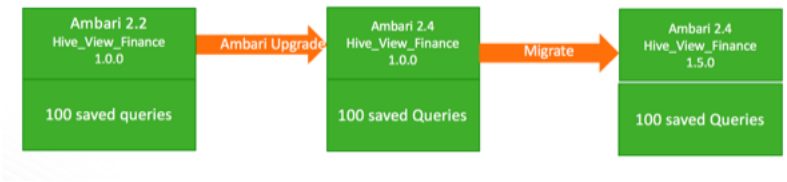
Previously, HDP only installed Hive View 1.0. Hive View 1.5 is now the default when you create a new view.



You can run both views simultaneously, use only one of the views, or upgrade your data from the older view to the newer view. Hortonworks recommends, for enhanced security and because of the future deprecation of Hive View 1.0, that you upgrade and migrate your data from the Hive View 1.0 to the Hive View 1.5.

9.1. Upgrading Your Hive View

If you are upgrading from Apache Ambari 2.2.0 to Apache Ambari 2.4.0 and want to upgrade and migrate the data and queries in your Hive Views, you will need to create a new instance of the Hive View and then migrate your queries.

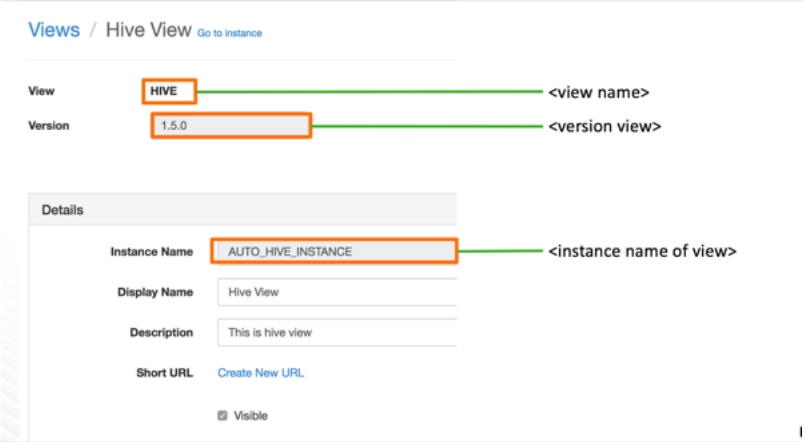


Migrating your queries into the new view

Create a new Hive View 1.5 instance and then you will migrate the saved queries from the Hive View 1.0 instance to the new instance. To do that, run the following curl command.

```
curl -v -u admin:admin -X PUT -H X-Requested-By:1 http://<host/
ip ambari server>:8080/api/v1/views/<view name>/versions/<version
of target view>/instances/<instance name of target view>/
migrate/<version of source view>/<instance name of source view>
```

For information on where to get the specific parameters listed in the curl command, refer to the following figure:



9.2. Configuring Your Cluster

For the Hive View to access HDFS, the Ambari Server daemon hosting the view needs to act as the proxy user for HDFS. This allows Ambari to submit requests to HDFS on behalf of the users using the Hive View. This is critical since the Hive View will store metadata about their user Hive queries in HDFS. This also means users that will access the Hive View must have a user directory setup in HDFS.

Note

If you are running views in an operational Ambari server (one that is operating the cluster) Ambari does this setup by default. You should verify that the setup described in the following subsections has been completed. If you are running views on a standalone server, you must setup proxy user settings manually, using the following instructions.

- Setup HDFS Proxy User
- Setup HDFS User Directory

9.2.1. Setup HDFS Proxy User

To set up an HDFS proxy user for the Ambari Server daemon account, you need to configure the proxy user in the HDFS configuration. This configuration is determined by the account name the **ambari-server** daemon is running as. For example, if your ambari-server is running as **root**, you set up an HDFS proxy user for **root** with the following:

1. In Ambari Web, browse to **Services > HDFS > Configs**.
2. Under the **Advanced** tab, navigate to the **Custom core-site** section.
3. Click **Add Property...** to add the following custom properties:

```
hadoop.proxyuser.root.groups="users"
hadoop.proxyuser.root.hosts=ambari-server.hostname
```

Notice the **ambari-server** daemon account name root is part of the property name. Be sure to modify this property name for the account name you are running the ambari-

server as. For example, if you were running **ambari-server** daemon under an account name of **ambariusr**, you would use the following properties instead:

```
hadoop.proxyuser.ambariusr.groups="users"
hadoop.proxyuser.ambariusr.hosts=ambari-server.hostname
```

Similarly, if you have configured Ambari Server for Kerberos, be sure to modify this property name for the **primary Kerberos principal** user. For example, if ambari-server is setup for Kerberos using principal **ambari-server@EXAMPLE.COM**, you would use the following properties instead:

```
hadoop.proxyuser.ambari-server.groups="users"
hadoop.proxyuser.ambari-server.hosts=ambari-server.hostname
```

4. Save the configuration change and restart the required components as indicated by Ambari.

9.2.2. Setup HDFS User Directory

The Hive View stores user metadata in HDFS. By default, the location in HDFS for this metadata is `/user/${username}` where `${username}` is the username of the currently logged in user that is accessing the Hive View.

Important

Since many users leverage the default Ambari admin user for getting started with Ambari, the `/user/admin` folder needs to be created in HDFS. Therefore, be sure to create the admin user directory in HDFS using these instructions prior to using the view.

To create user directories in HDFS, do the following for each user you plan to have use the Hive View.

1. Connect to a host in the cluster that includes the HDFS client.
2. Switch to the hdfs system account user.

```
su - hdfs
```

3. Using the HDFS client, make an HDFS directory for the user. For example, if your username is admin, you would create the following directory.

```
hadoop fs -mkdir /user/admin
```

4. Set the ownership on the newly created directory. For example, if your username is admin, you would make that user the directory owner.

```
hadoop fs -chown admin:hadoop /user/admin
```

9.3. Creating the Hive View Instance

1. Click **Manage Ambari** to open the Ambari Administration user interface.
2. Click **Views**, expand the **Hive View**, and click **Create Instance**.

3. On the Create Instance page, select the **Version**. If multiple Hive View jars are present, choose one.

4. Enter the following view instance Details:

Table 9.1. Hive View Instance Details

Property	Description	Example Value
Instance Name	This is the Hive view instance name. This value should be unique for all Hive view instances you create. This value cannot contain spaces and is required.	AUTO_HIVE_INSTANCE
Display Name	This is the name of the view link displayed to the user in Ambari Web.	Hive View
Description	This is the description of the view displayed to the user in Ambari Web.	Auto-created when the Hive service is deployed.
Visible	This checkbox determines whether the view is displayed to users in Ambari Web.	Visible or Not Visible

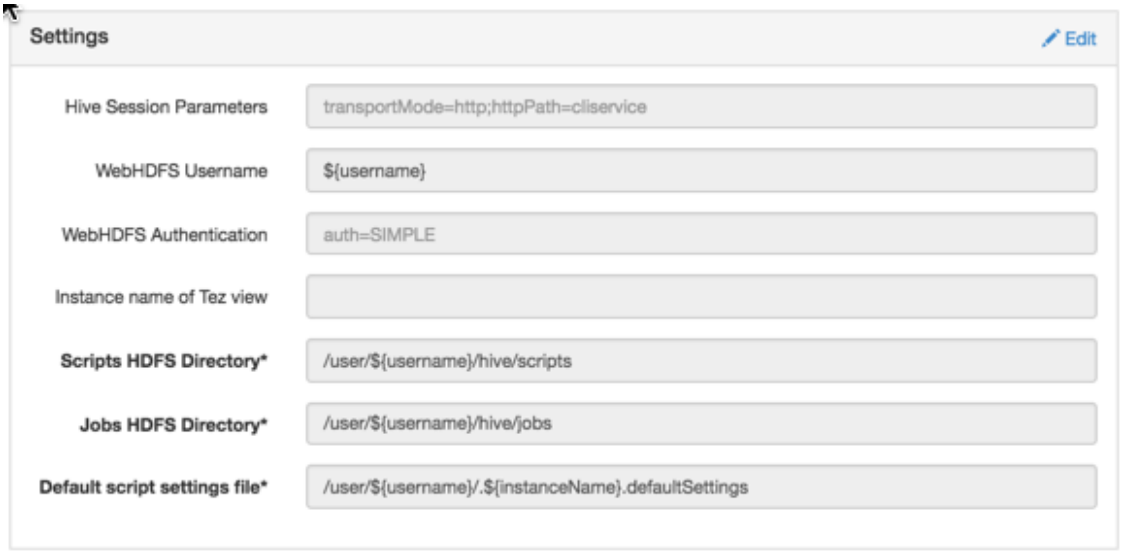
5. The **Settings** and **Cluster Configuration** options depend on a few cluster and deployment factors in your environment. Typically, you can accept the default **Settings** unless you are using the Hive View with a Kerberos-enabled cluster. Refer to Settings and Cluster Configuration for more information.

6. Click **Save**.

9.3.1. Settings and Cluster Configuration

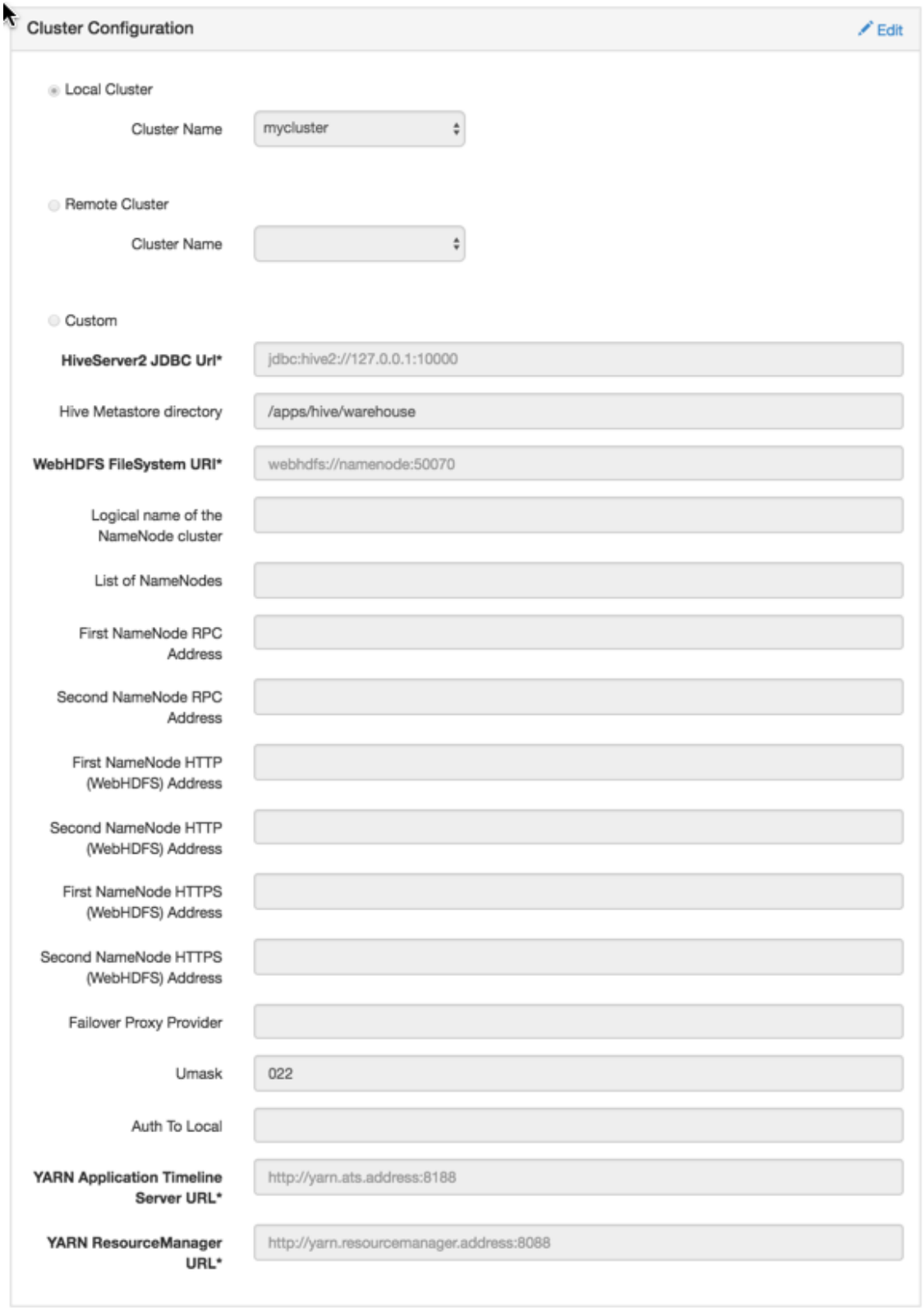
Ambari configures Hive View settings automatically when you choose to add the Hive service. The default Hive View settings are shown in the following figure:

Figure 9.1. Default Hive View Settings



The default Hive View cluster configuration is shown in the following figure:

Figure 9.2. Default Hive View Cluster Configuration



If required for migrating view instances, the following table describes how to locate cluster configuration settings using Ambari.

Table 9.2. Finding Cluster Configuration Values for the Hive View using Ambari

Property	Value
HiveServer2 JDBC URL For secured clusters, see Kerberos Setup for Hive Views	Click **Hive > Summary** to view the URL, diplayed at the bottom of the Summary list. For example: `jdbc:hive2://c6403.ambari.apache.org:2181,c6401.ambari.apache.org:2181,c6402.ambari.`
Hive Metastore directory	Click **Hive > Configs > Advanced > General**. For example, `/apps/hive/warehouse`
WebHDFS FileSystem URI*	Click **HDFS > Configs > Advanced >Advanced hdfs-site** For example `dfs.nameserviceid.http-address`
	For HA: Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.nameservice.id.** When you enter the value in the view definition, pre-pend "webhdfs://" to the value you find in the advanced HDFS configuration settings. For example, `webhdfs://c6401.ambari.apache.org:50070 or webhdfs://nameserviceid`
Logical Name of the NameNode cluster	
List of NameNodes	
First NameNode RPC Address	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.namenode.rpc-address**. See the first address in the list. For example, `c6401.ambari.apache.org`
Second NameNode RPC Address	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.namenode.rpc-address**. See the second address in the list. For example, `c6402.ambari.apache.org`
First NameNode HTTP (WebHDFS) Address	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.namenode.http-address** See the first address in the list. For example, `c6401.ambari.apache.org`
Second NameNode HTTP (WebHDFS) Address	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.namenode.http-address** See the second address in the list. For example, `c6402.ambari.apache.org`
First NameNode HTTPS (WebHDFS) Address	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.namenode.https-address** See the first address in the list. For example, `c6401.ambari.apache.org`
Second NameNode HTTPS (WebHDFS) Address	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.namenode.https-address** See the second address in the list. For example, `c6402.ambari.apache.org`
Failover Proxy Provider	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.client.failover.proxy.provider[nameservice]**
Umask	Click **HDFS > Configs > Advanced > Advanced hdfs-site > fs.permissions.umask-mode** See the second address in the list. For example, `c6402.ambari.apache.org`
Auth To Local	Click **HDFS > Configs > Advanced > Advanced core-site > hadoop.security.auth_to_local**
YARN Application Timeline Server URL*	Click **YARN > Configs > Advanced > Application Timeline Server > yarn.timeline-service.webapp.address**. When you enter the value in the view definition, pre-pend

Property	Value
	"http://" to the value you find in the YARN advanced configuration settings. For example, `http://c6401.ambari.apache.org:8188`
YARN ResourceManager URL*	Click **YARN > Configs > Advanced > Advanced yarn-site > yarn.resourcemanager.webapp.address**. When you enter the value in the view definition, pre-pend "http://" to the value you find in the YARN advanced configuration settings. For example, `http://c6401.ambari.apache.org:8088`

For NameNode High Availability

The following values must be entered for primary and secondary NameNodes:

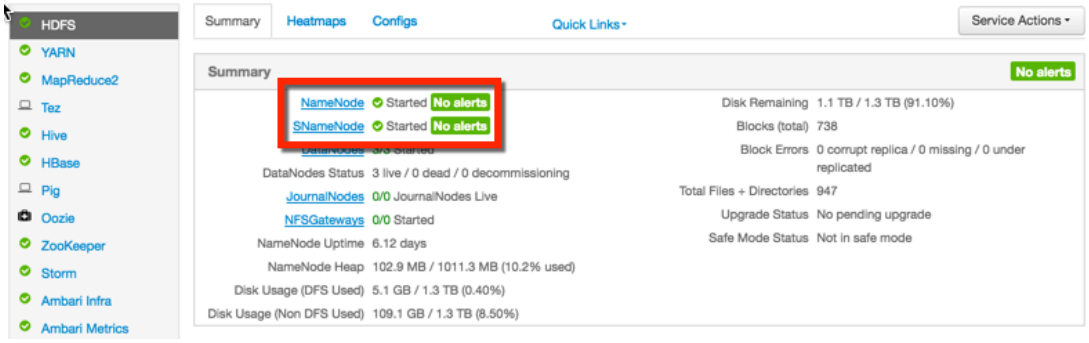
Table 9.3. Hive View Settings for NameNode High Availability

Property	Value
First NameNode RPC Address or **Second NameNode RPC Address**	Select the primary or secondary NameNode to view settings from that host in the cluster. See how to get the NameNode RPC address [49]. When you enter the value in the view definition, pre-pend "http://" to the value you find in the advanced `hdfs-site` settings. For example, `http://c6401.ambari.apache.org:8020`
First NameNode HTTP (WebHDFS) Address or **Second NameNode HTTP (WebHDFS) Address**	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.namenode.http-address**. When you enter the value in the view definition, pre-pend "http://" to the value you find in the advanced `hdfs-site` settings. For example, `http://c6401.ambari.apache.org:50070`

To get First NameNode RPC Address values:

1. Navigate to the HDFS service page in Ambari that contains links to individual NameNodes. Click **NameNode** (primary) or **SNameNode** (secondary) to view the host page:

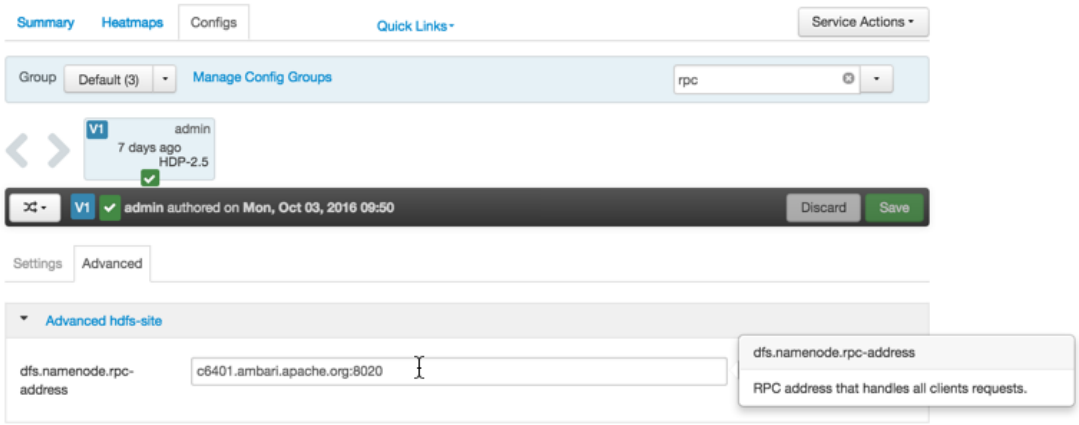
Figure 9.3. HDFS Service Page in Ambari



2. On the host page, click **Configs > Advanced**.

3. Enter "rpc" in the Filter search well at the top right corner of the page or navigate to the **Advanced hdfs-site** settings to find the `dfs.namenode.rpc-address` value that you can enter into the Hive View definition. Here is an example of using the Filter to locate a value:

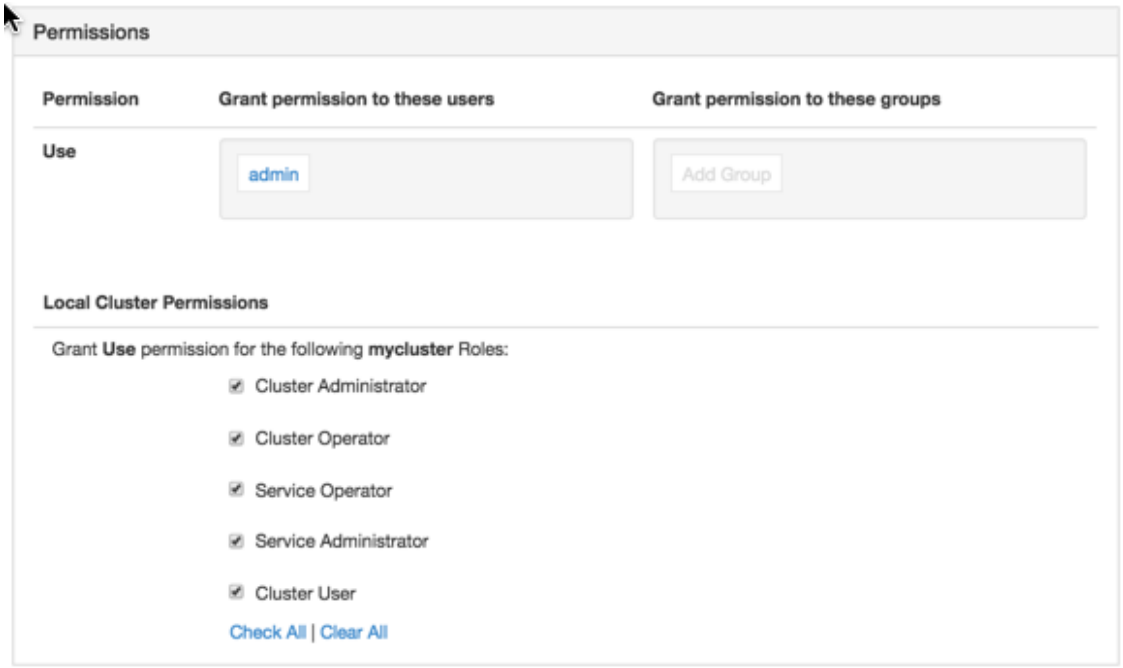
Figure 9.4. Using the Filter to Search Advanced hdfs-site Settings



9.3.2. User Permissions for Hive Views

After saving the Hive View instance definition, grant permission on the view for the set of users who can use the view:

Figure 9.5. Granting User Permissions to Hive Views



9.3.3. Kerberos Setup for Hive Views

To set up basic Kerberos for views, see "Set Up Kerberos for Ambari Server" in Hortonworks Data Platform Apache Ambari Security. After you have set up basic Kerberos for the Hive View, Hive requires the following additional setting:

Property	Value
WebHDFS Authentication	`auth=KERBEROS;proxyuser=<ambari-principal>` **Note** This property is only needed if the view is Custom Configured or Ambari Server is Kerberized

9.4. Using the Hive View

Use the Hive View to:

- Browse databases
- Write and execute queries
- Manage query execution jobs and history

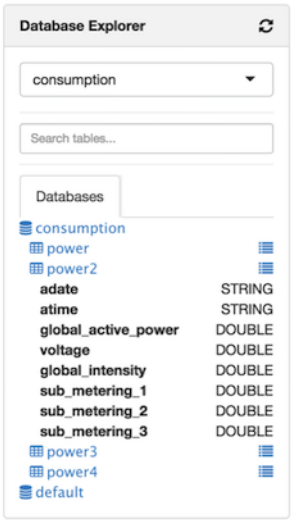
9.4.1. Query Tab

Click the **Query** tab to browse database tables and columns and to build, execute, and debug queries.

Database Explorer

The Database Explorer enables you to view all databases and tables in Hive that you have permissions to view. It is designed to navigate a large number of databases, tables, and columns:

Figure 9.6. Hive View Database Explorer



Features of Database Explorer:

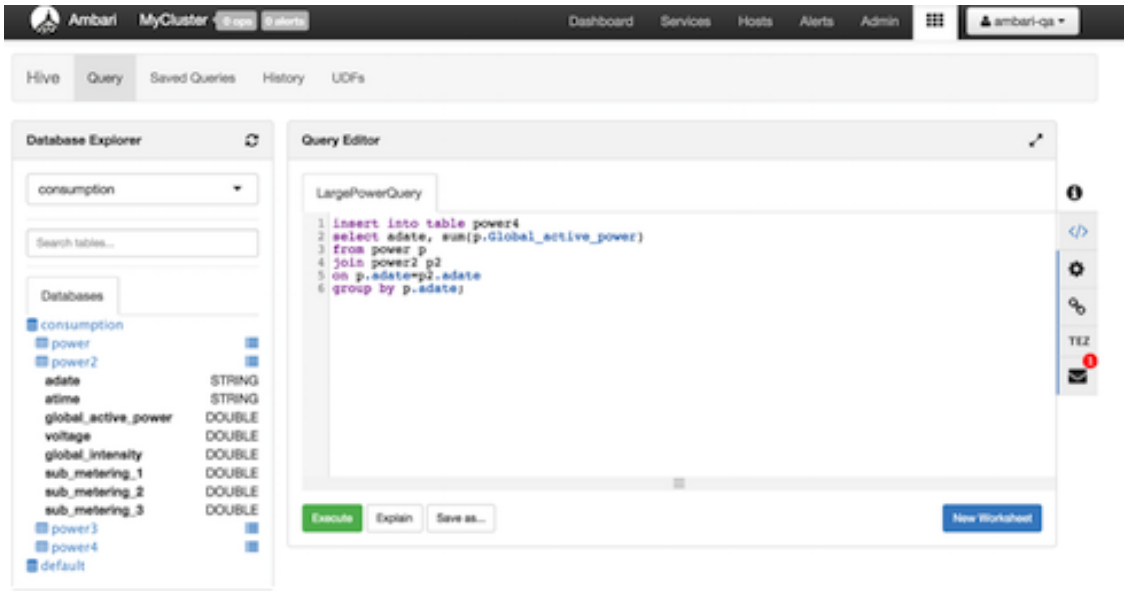
- Click the refresh icon in the top right to view tables that were created since the Hive View session began.

- Select a database from the drop-down list. All queries in the current tab are then run against the selected database. You can also edit the drop-down list to enable substring searches over a large number of databases.
- Use the Search tables and Search columns fields to search when you have a large number of tables and columns.
- Browse the Databases tab to view all of the databases, tables, and columns. This is useful when you are authoring queries. The icon to the right of a table enables you to see sample data within that table.

Query Editor

You can author and execute queries in the Query Editor:

Figure 9.7. Query Editor

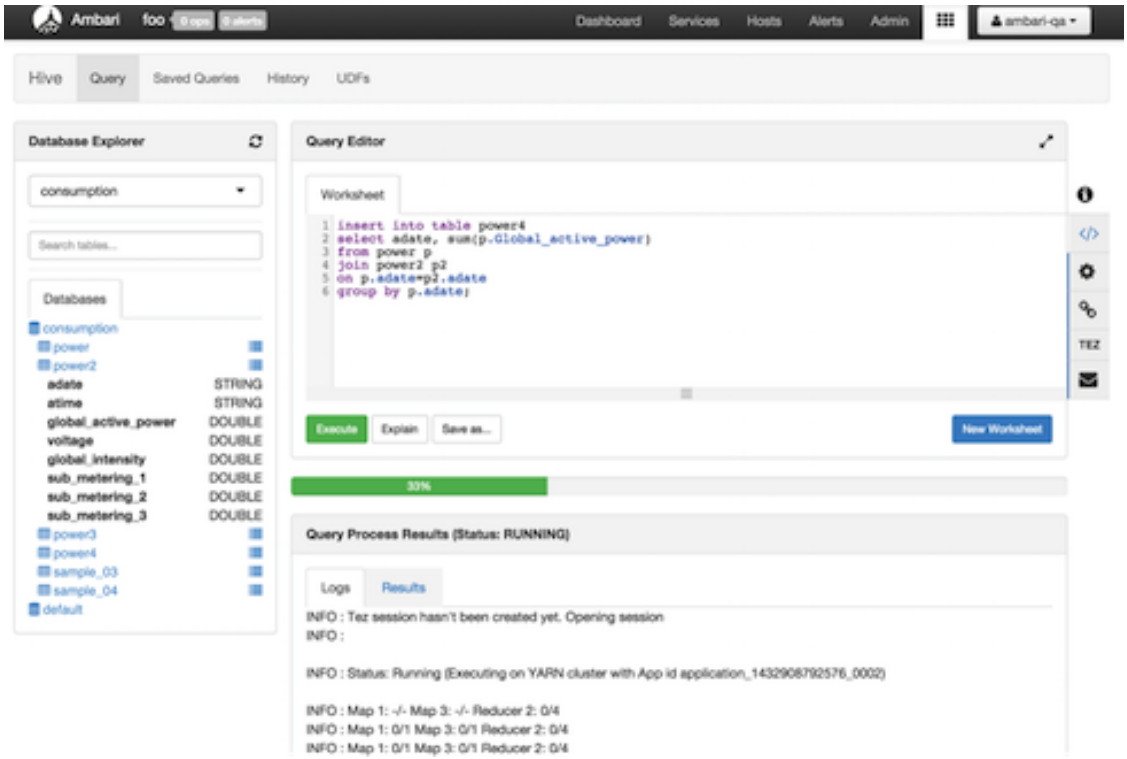


Features and Behavior of Query Editor

- All queries contained in a Worksheet tab execute sequentially, and they run in the same session. Running all queries in one pass requires handling the output of multiple select statements and is not supported in the 1.0 version.
- To run a specific query, highlight it, and click **Execute**.
- When the first query is executed in a Worksheet, a Tez session is opened.
- Click **Save as** to save your query.
- Double-click the **Worksheet** tab to rename the query, click **OK**, and then **Save as** to save the query with the new name.
- Click **New Worksheet** to open a new worksheet tab. Queries executed from the new worksheet tab will execute in a different session. Queries from different worksheets can execute in parallel.

- Press **CTRL + space** to autocomplete query statements.
- Click the double arrow icon in the upper right corner of the Query Editor to expand the Worksheet area and cover Database Explorer. Click the icon again to collapse the Worksheet and make Database Explorer available again.
- Click the icon at the bottom of the Worksheet window and drag it down to expand the authoring space.
- Query results and logs display below the query when it is executed.

Figure 9.8. Query Results and Logs in Hive View Query Editor



Query Editor Settings

Click the gear icon on the right margin of the worksheet to access settings for the Query Editor. Then click **Add**, select a setting parameter from the drop-down list, and then select a value for the parameter. Query Editor settings are configured per worksheet.

To save settings as default settings so they are applied each time that a new worksheet is opened, click **Save Default Settings** in the upper right corner of the settings window.

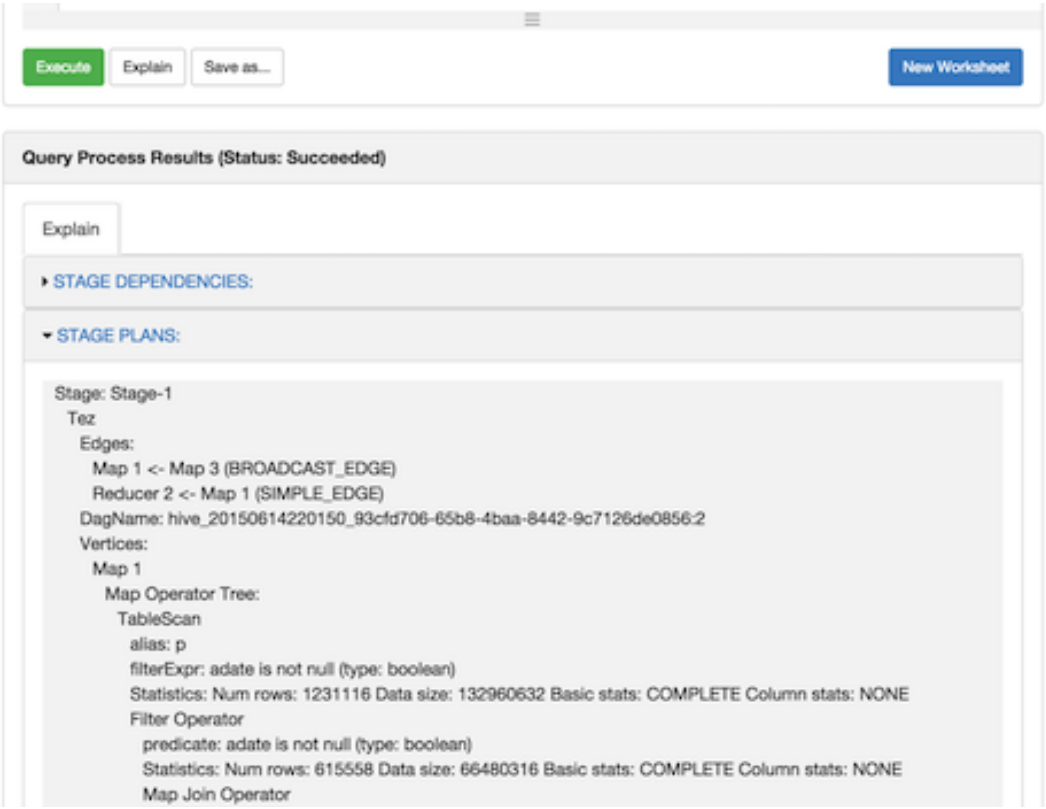
Click **SQL** to the right of the Worksheet window to exit settings and return to the Query Editor authoring pane.

Text Explain and Visual Explain

There are two options that help you understand how your queries are executed. One is a textual explanation of your query and the other form explains the query visually as a diagram. In future releases, column lineage will be added.

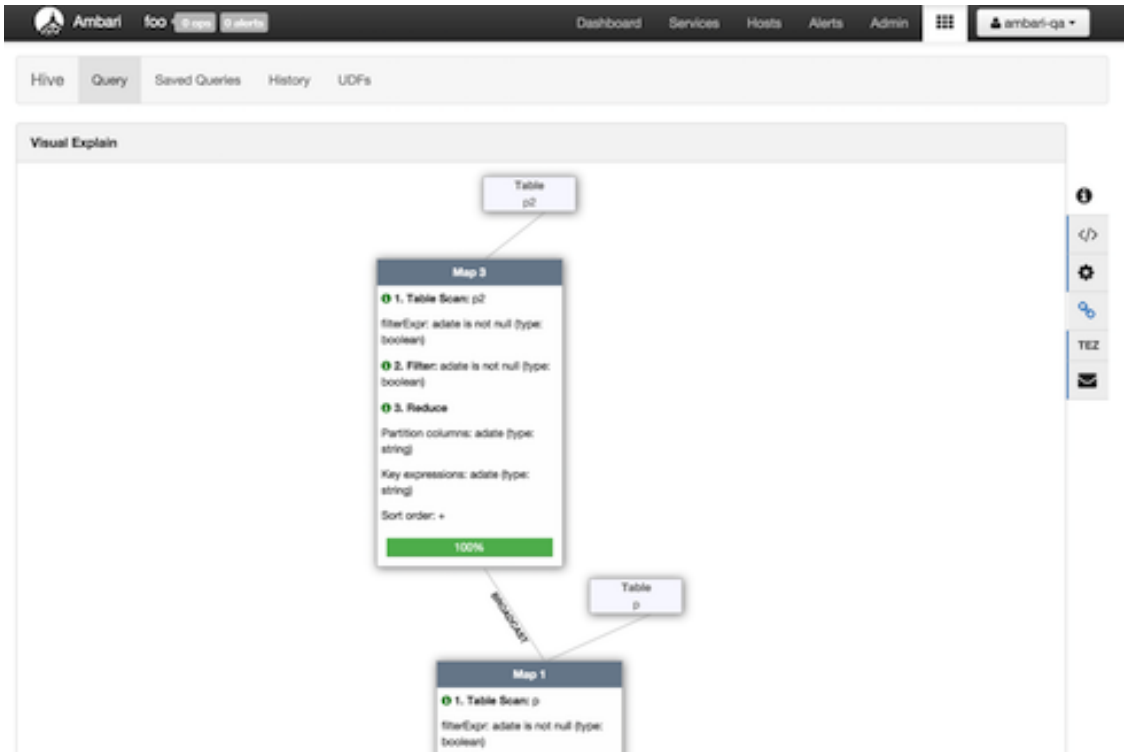
The **Explain** button in the lower left corner of the Worksheet window launches a textual explanation:

Figure 9.9. Query Editor Textual Explain Feature



To launch the Visual Explain diagram, click the link icon to the right of the Worksheet window. If the query is running, Visual Explain shows the query execution progress per vertex:

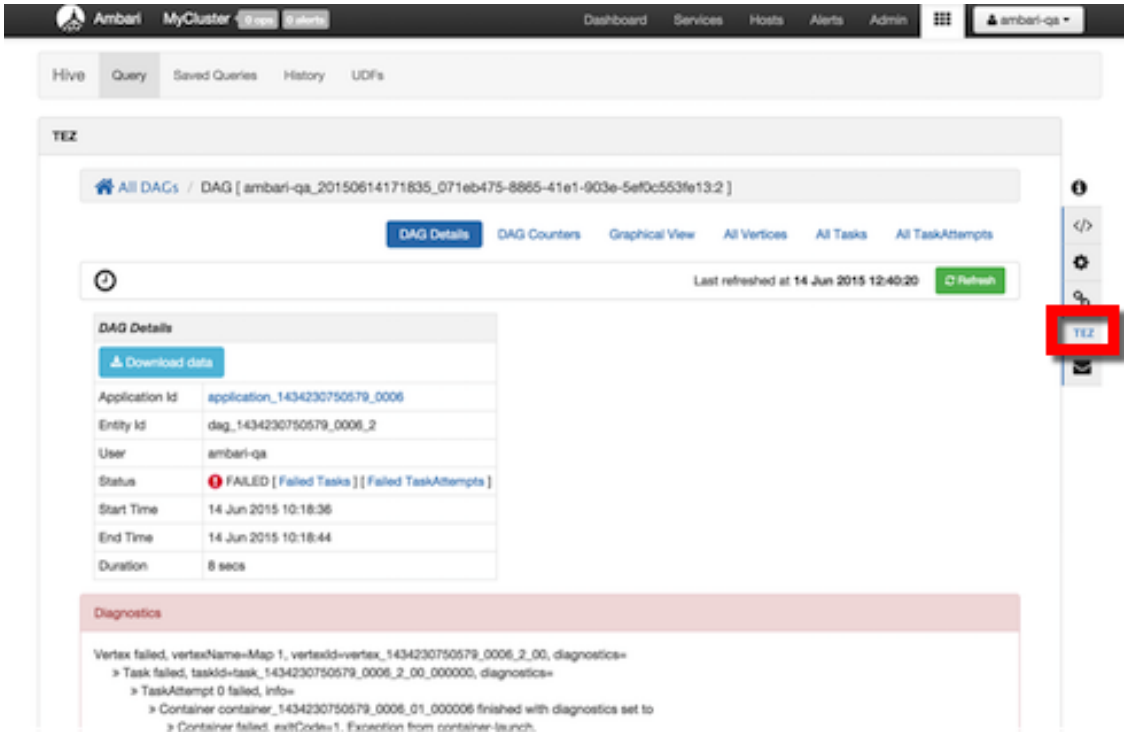
Figure 9.10. Query Editor Visual Explain Feature



Debugging Hive Query Execution Using the Tez View

Query execution can be debugged using the embedded Tez view. To access the Tez view, click **TEZ** in the toolbar on the right of the Worksheet window:

Figure 9.11. Tez View Query Debugging Option

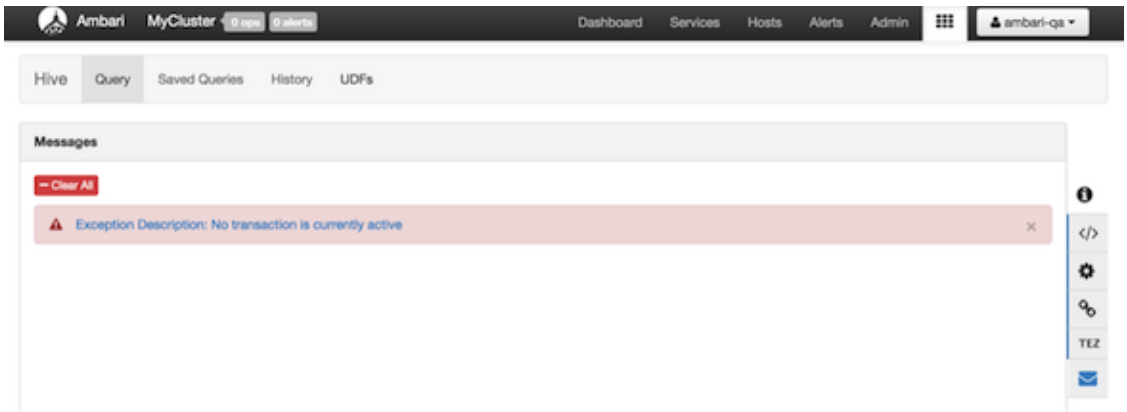


When a query fails, the Status field displays **FAILED** and there is a link to Failed Tasks and the error displays on the first page. Click **Download data** to get the data for the task. For further details on debugging, see Using the Tez View.

Errors and Alerts

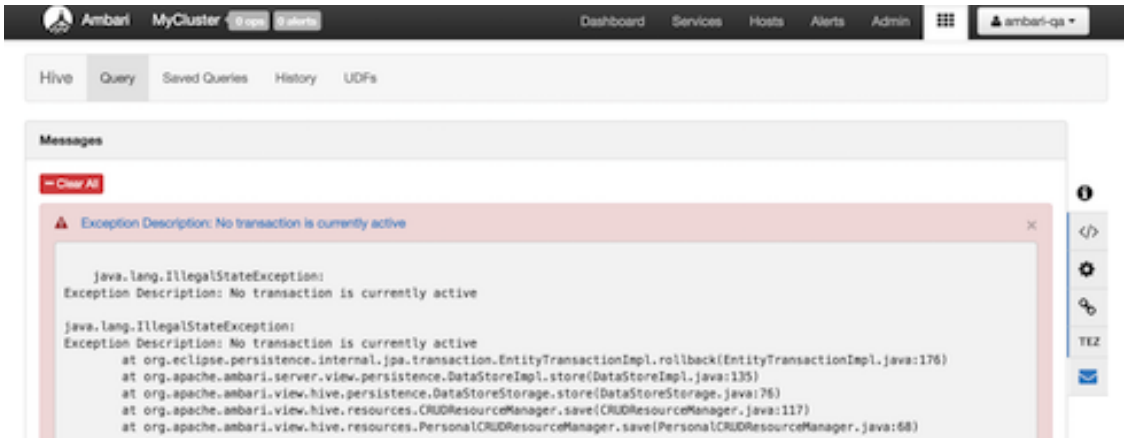
Errors and alerts can be viewed by clicking the envelope icon in the toolbar to the right of the Worksheet window. When the icon is clicked, all the messages are shown with a one-line summary per message:

Figure 9.12. Query Editor Error Message Summary Window



If you want to view details of the errors, expand the summary by clicking it. The details text can be copied into a bug report:

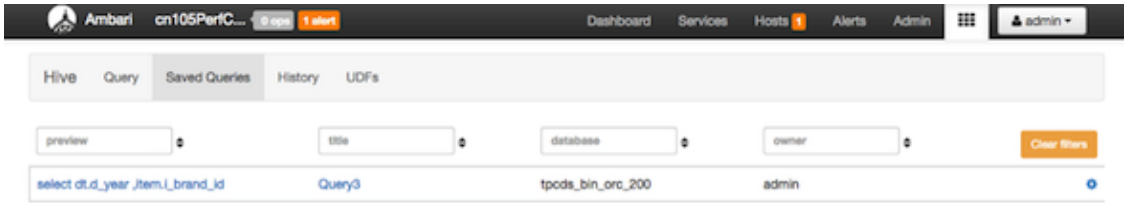
Figure 9.13. Query Editor Error Message Details Window



9.4.2. Saved Queries Tab

The Saved Queries tab shows all the queries that have been saved by the current user. Click the gear icon to the right of the query list to view the history of a query or to delete it:

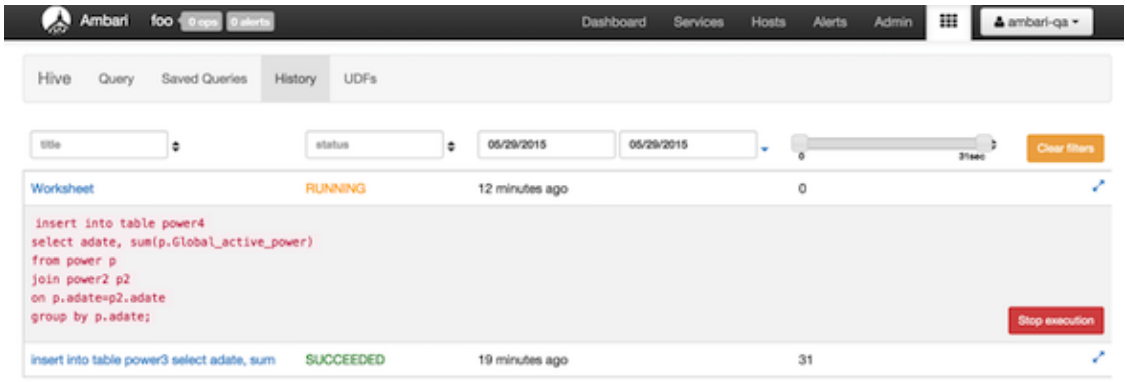
Figure 9.14. Saved Queries Tab



9.4.3. History Tab

You can view the history of all jobs run by the current user in the History tab. It pulls history from the Application Timeline Server database. All queries for which logs are present in that database are displayed here. This means that regardless of the source of the query, (CLI, JDBC/ODBC, Hive View) it will appear here on the History tab. Queries that have not been assigned a name, such as those created in the Hive View, appear as query text. For example, see the insert statement that was submitted by CLI in the following image:

Figure 9.15. History Tab

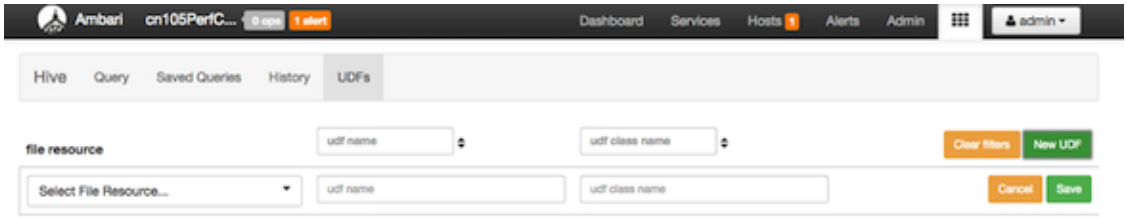


For queries that are submitted from the Hive View, a Stop Execution button is available to enable you to end a currently running query. When you select a query by clicking the title in the first column, that query appears on a new sub-tab in the Query tab where it can be analyzed and debugged.

9.4.4. UDF Tab

User-defined functions (UDFs) can be added to queries by pointing to a JAR file on HDFS, which contains the UDF definition. After the UDF is added here, an Insert UDF button appears in the Query Editor that enables you to add the UDF to your query:

Figure 9.16. UDF Tab



9.5. Upload Table tab

In the Upload Table tab, you can upload files which contain the rows of the Apache Hive table. The Upload Table command supports various input file formats. On uploading, it creates a new Hive table with the data.

Input File Formats:

CSV, XML, and JSON files are supported for input.

CSV

Supported types are:

- CSV with custom field delimiter (default is comma ,)

- Quote character (default is double quote “) Escape character (default is backslash \ )

The row delimiter must be \n or \r or \r\n. If Is first row header? is selected then first row of the file is treated as column names. During preview this can be changed by clearing this field but other delimiters should not be changed during the preview. The number of columns in the table and their order is defined by the first line of the file, irrespective of whether it represents column names nor not. If there are extra columns in line 2 onwards, they are ignored. If there are lesser columns in line 2 onwards then the rest of the columns are assumed null values.

XML

The format of the contents in the XML file should be as shown below:

```
<table>
        <row>
        <col name="col1Name">row1-col1-Data</col>
        <col name="col2Name">row1-col2-Data</col>
        <col name="col3Name">row1-col3-Data</col>
        <col name="col4Name">row1-col4-Data</col>
  </row>
  <row>
        <col name="col1Name">row2-col1-Data</col>
        <col name="col2Name">row2-col2-Data</col>
        <col name="col3Name">row2-col3-Data</col>
        <col name="col4Name">row2-col4-Data</col>
  </row>
</table>
```

The root tag must be <table>. Inside <table> there can be any number of <row> tags representing one row of the table. Inside each <row> tag there can be any number of <col> tags representing columns of the row. Each <col> tag must have a “name” attribute, which will be treated as the name of column. Column values should be within the <col> tag. The names, number and order of columns are decided by the first <row> entry. The names of column and datatypes can be changed during the Preview.

JSON

The following json format is supported: [{ "col1Name" : "value-1-1", "col2Name" : "value-1-2"}, { "col1Name" : "value-2-1", "col2Name" : "value-2-2"}]

The file should contain a valid json array containing any number of json objects. Each json object should contain column names as property and column values as property values. The names, number and order of columns in the table are decided from the first object of the json file. The names and datatype of column can be edited during the preview step. If some json objects have extra properties then they are ignored. If they do not have some of the properties then null values are assumed. Note that extension of files cannot be “.json”

To import a file into the Hive view:

1. Select **Upload from Local** or **Upload from HDFS**.
2. Select the input file format File type by specifying CSV, XML, or JSON.
3. If the File Type is CSV, you can select the **Field Delimiter**, the **Escape Character**, the **Quote Character** and **Is first row header?** values for CSV by clicking on the gear icon.
4. If you selected **Upload from Local**, you can choose the file from your local machine. Otherwise, enter the full HDFS path and click **Preview**. The file is partially read from client's browser or HDFS and the preview is generated with a suggested table name, column names, column data types and 10 rows from the data file.
5. You can select **Database** from the drop-down list, change the suggested table name, column names, column types, precision and scale and hive storage type **Stored as** as required.
6. If Stored as is TEXTFILE, then a gear next to it is enabled and you can click it to select **Fields Terminated By**, and **Escape By** to be used in creation of the Hive table.
7. If Stored as is NOT TEXTFILE, another option **Contains endlines?** is enabled. If the column values in your file contain endline characters, ("\n" newline, ASCII 10 or "\r" carriage return, ASCII 13) then you must check this field for proper handling otherwise unexpected results might occur. Endline characters are not supported in TEXTFILE format.
8. Click **Upload table**. The actual table and temporary table (Stored as TEXTFILE) is created. After this the data rows from the file are inserted into the temporary table followed by Insert from temporary table to actual table.
9. On success the temporary table is deleted and workflow completes.

In case of failure, an error is reported and the temporary table and actual tables are deleted. You can see the error message by clicking the message icon at the top right. Clicking again on the message icon brings back the Upload Table page. You can perform any changes required and click **Upload** again to upload the same file or restart the process by selecting a different file.

9.6. Troubleshooting

Table 9.4. Troubleshooting Hive Views Errors

Error	Solution
User: root is not allowed to impersonate admin	HDFS has not been configured for Ambari as a proxy user. Refer to Setup HDFS Proxy User.
E090 HDFS020 Could not write file /user/admin/hive/jobs/hive-job-1-2015-10-30_02-12/query.hql [HdfsApiException]	The user does not have a user directory in HDFS for the view to store metadata about the view. Refer to Setup HDFS User Directory.

10. Using the Pig View

Apache Pig is a scripting platform for processing and analyzing large data sets. Pig was designed to perform extract-transform-load (ETL) operations, raw data research, and iterative data processing. The **Pig View** provides a web-based interface to compose, edit, and submit Pig scripts, download results, and view logs and the history of job submissions.

This chapter explains:

- Configuring Your Cluster
- Creating the Pig View Instance
- Using the Pig View

10.1. Configuring Your Cluster

For the Pig View to access HDFS, the Ambari Server daemon hosting the view needs to act as the proxy user for HDFS. This allows Ambari to submit requests to HDFS on behalf of the users using the Pig View. This is critical since the Pig View will store metadata about the user Pig scripts. This also means users that will access the Pig View must have a user directory setup in HDFS. In addition, the Pig View uses WebHCat to submit Pig scripts so the View needs a proxy user for WebHCat.

Note

If you are running views in an operational Ambari server (one that is operating the cluster) Ambari does this setup by default. You should verify that the setup described in the following subsections has been completed. If you are running views on a standalone server, you must setup proxy user settings manually, using the following instructions.

- Setup HDFS Proxy User
- Setup WebHCat Proxy User
- Setup HDFS User Directory

10.1.1. Setup HDFS Proxy User

To set up an HDFS proxy user for the Ambari Server daemon account, you need to configure the proxy user in the HDFS configuration. This configuration is determined by the account name the **ambari-server** daemon is running as. For example, if your ambari-server is running as **root**, you set up an HDFS proxy user for **root** with the following:

1. In Ambari Web, browse to **Services > HDFS > Configs**.
2. Under the **Advanced** tab, navigate to the **Custom core-site** section.
3. Click **Add Property...** to add the following custom properties:

```
hadoop.proxyuser.root.groups="users"
hadoop.proxyuser.root.hosts=ambari-server.hostname
```

Notice the **ambari-server** daemon account name root is part of the property name. Be sure to modify this property name for the account name you are running the ambari-server as. For example, if you were running **ambari-server** daemon under an account name of **ambariusr**, you would use the following properties instead:

```
hadoop.proxyuser.ambariusr.groups="users"
hadoop.proxyuser.ambariusr.hosts=ambari-server.hostname
```

Similarly, if you have configured Ambari Server for Kerberos, be sure to modify this property name for the **primary Kerberos principal** user. For example, if ambari-server is setup for Kerberos using principal **ambari-server@EXAMPLE.COM**, you would use the following properties instead:

```
hadoop.proxyuser.ambari-server.groups="users"
hadoop.proxyuser.ambari-server.hosts=ambari-server.hostname
```

4. Save the configuration change and restart the required components as indicated by Ambari.

10.1.2. Setup WebHCat Proxy User

You must set up an HDFS proxy user for WebHCat and a WebHCat proxy user for the Ambari Server daemon account.

To setup the HDFS proxy user for WebHCat :

1. In Ambari Web, browse to **Services > HDFS > Configs**.
2. Under the **Advanced** tab, navigate to the **Custom core-site** section.
3. Click **Add Property...** to add the following custom properties:

```
hadoop.proxyuser.hcat.groups=*
hadoop.proxyuser.hcat.hosts=*
```

4. Save the configuration change and restart the required components as indicated by Ambari.

To setup a WebHCat proxy user for the Ambari Server daemon account, you need to configure the proxy user in the WebHCat configuration. This configuration is determined by the account name the **ambari -server** daemon is running as. For example, if your ambari -server is running as **root**, you set up an WebHCat proxy user for **root** with the following:

1. In Ambari Web, browse to **Services > Hive > Configs**.
2. Under the **Advanced** tab, navigate to the **Custom webhcat-site** section.
3. Click **Add Property...** to add the following custom properties:

```
webhcat.proxyuser.root.groups=*
webhcat.proxyuser.root.hosts=*
```

Notice the **ambari-server** daemon account name root is part of the property name. Be sure to modify this property name for the account name you are running the ambari-

server as. For example, if you were running **ambari-server** daemon under an account name of **ambariusr**, you would use the following properties instead:

```
webhcat.proxyuser.ambariusr.groups=*
webhcat.proxyuser.ambariusr.hosts=*
```

Similarly, if you have configured Ambari Server for Kerberos, be sure to modify this property name for the **primary Kerberos principal** user. For example, if ambari-server is setup for Kerberos using principal **ambari-server@EXAMPLE.COM**, you would use the following properties instead:

```
webhcat.proxyuser.ambari-server.groups=*
webhcat.proxyuser.ambari-server.hosts=*
```

4. Save the configuration change and restart the required components as indicated by Ambari.

10.1.3. Setup HDFS User Directory

The Hive View stores user metadata in HDFS. By default, the location in HDFS for this metadata is `/user/${username}` where `${username}` is the username of the currently logged in user that is accessing the Hive View.

Important

Since many users leverage the default Ambari admin user for getting started with Ambari, the `/user/admin` folder needs to be created in HDFS. Therefore, be sure to create the admin user directory in HDFS using these instructions prior to using the view.

To create user directories in HDFS, do the following for each user you plan to have use the Hive View.

1. Connect to a host in the cluster that includes the HDFS client.
2. Switch to the hdfs system account user.

```
su - hdfs
```

3. Using the HDFS client, make an HDFS directory for the user. For example, if your username is admin, you would create the following directory.

```
hadoop fs -mkdir /user/admin
```

4. Set the ownership on the newly created directory. For example, if your username is admin, you would make that user the directory owner.

```
hadoop fs -chown admin:hadoop /user/admin
```

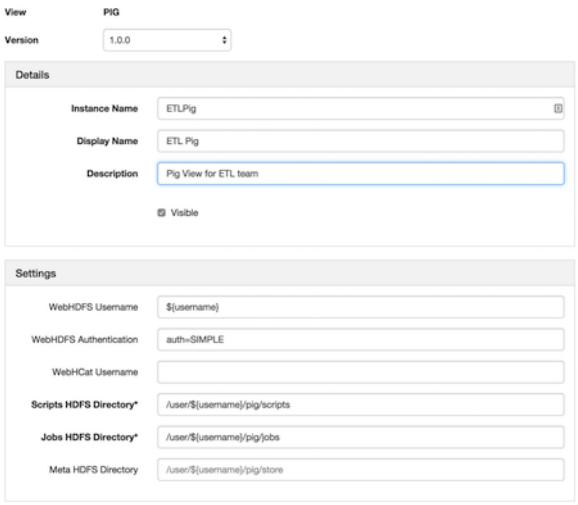
10.2. Creating the Pig View Instance

1. Browse to the Ambari Administration interface.
2. Click **Views**, expand the **Pig View**, and click **Create Instance**.

3. On the Create Instance page, select **Version**. If multiple Pig View jars are present, choose one.

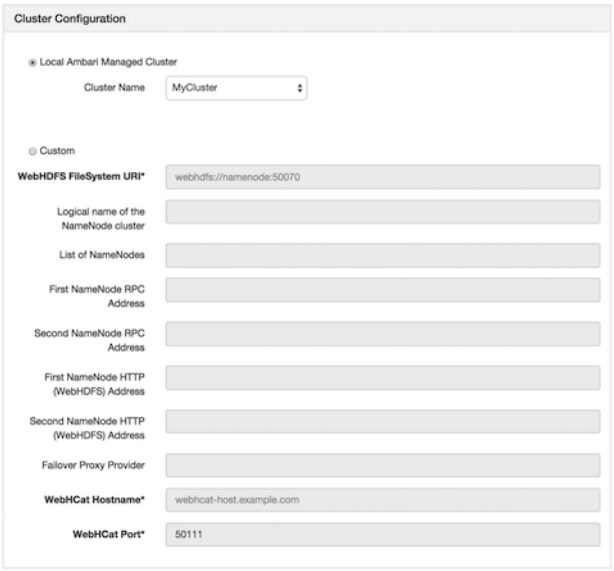
4. Enter the Details and Settings. The Instance Name appears in the URI, the Display Name appears in the Views drop-down list, and the Description helps multiple users identify the view:

Figure 10.1. Pig View Details and Settings



5. Scroll down, and enter the Cluster Configuration information, which tells the Pig View how to access resources in the cluster. For a cluster that is deployed and managed by Ambari, select **Local Ambari Managed Cluster**:

Figure 10.2. Pig View Cluster Configuration



6. Click **Save**, give Permissions to the appropriate users and groups, and click **Go to instance** at the top of the page to go to the view instance.

10.2.1. Getting Correct Configuration Values for Manually-Deployed Clusters

If you have manually deployed your cluster, you must enter cluster configuration values in the Pig View Create Instance page. The following table explains where you can find cluster configuration settings in Ambari.

Table 10.1. Finding Cluster Configuration Values for the Pig View in Ambari

Property	Value
Scripts HDFS Directory*	`/user/${username}/pig/scripts`
Jobs HDFS Directory*	`/user/${username}/pig/jobs`
WebHDFS FileSystem URI*	Click **HDFS > Configs > Advanced hdfs-site > dfs.namenode.http-address**. When you enter the value in the view definition, pre-pend "webhdfs://" to the value you find in the advanced HDFS configuration settings. For example, `webhdfs://c6401.ambari.apache.org:50070`
WebHCat Hostname*	Click **Hive > Configs > Advanced > WebHCat Server > WebHCat Server host** to view the hostname. For example, `c6402.ambari.apache.org`
WebHCat Port*	Click **Hive > Configs > Advanced > Advanced webhcat-site > templeton.port** to view the port number. For example, `50111`

For NameNode High Availability

The following values must be entered for primary and secondary NameNodes:

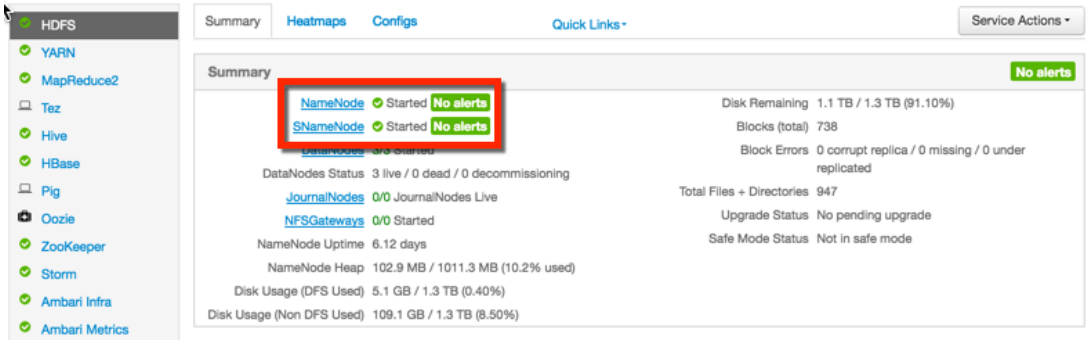
Table 10.2. Pig View Settings for NameNode High Availability

Property	Value
First NameNode RPC Address or **Second NameNode RPC Address**	Select the primary or secondary NameNode to view settings from that host in the cluster. See how to get the NameNode RPC address [65]. When you enter the value in the view definition, pre-pend "http://" to the value you find in the advanced `hdfs-site` settings. For example, `http://c6401.ambari.apache.org:8020`
First NameNode HTTP (WebHDFS) Address or **Second NameNode HTTP (WebHDFS) Address**	Click **HDFS > Configs > Advanced > Advanced hdfs-site > dfs.namenode.http-address**. When you enter the value in the view definition, pre-pend "http://" to the value you find in the advanced `hdfs-site` settings. For example, `http://c6401.ambari.apache.org:50070`

To get First NameNode RPC Address values:

1. Navigate to the HDFS service page in Ambari that contains links to individual NameNodes. Click **NameNode** (primary) or **SNameNode** (secondary) to view the host page:

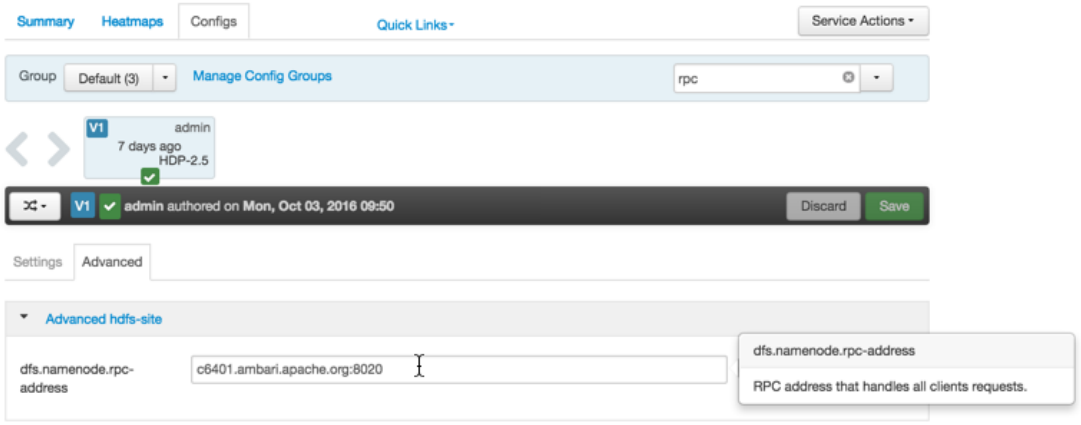
Figure 10.3. HDFS Service Page in Ambari



2. On the host page, click **Configs > Advanced**.

3. Enter "rpc" in the Filter search well at the top right corner of the page or navigate to the **Advanced hdfs-site** settings to find the `dfs.namenode.rpc-address` value that you can enter into the Pig View definition. Here is an example of using the Filter to locate a value:

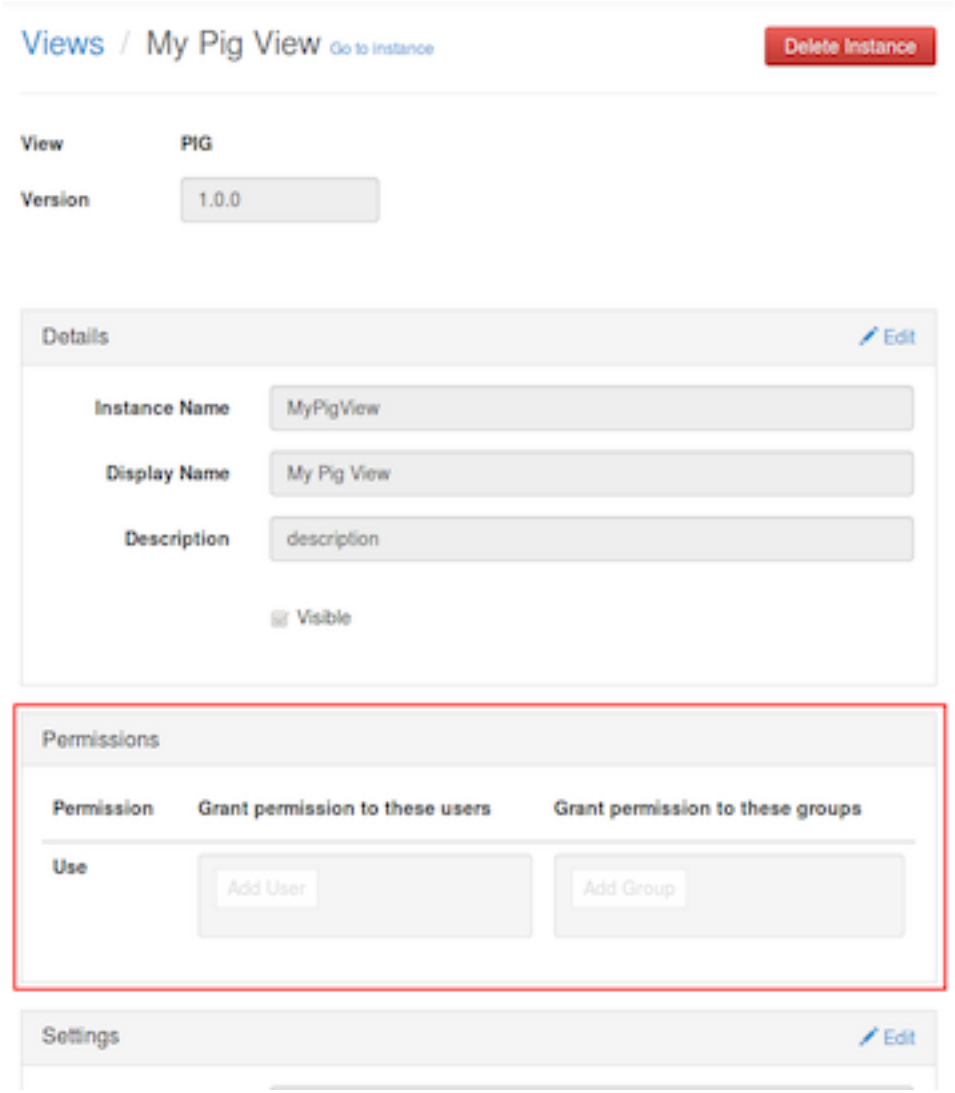
Figure 10.4. Using the Filter to Search Advanced hdfs-site Settings



10.2.2. User Permissions for Pig Views

After saving the Pig View instance definition, grant permission on the view for the set of users who can use the view:

Figure 10.5. Granting User Permissions to Pig Views



10.2.3. Kerberos Setup for Pig Views

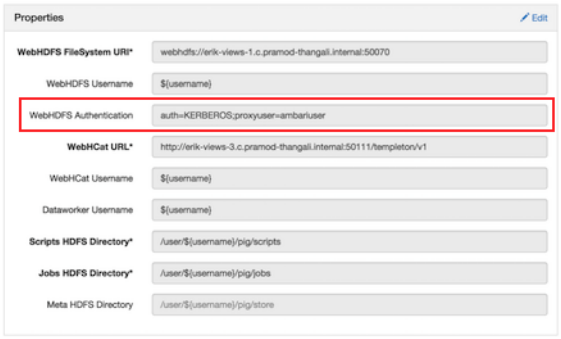
To set up basic Kerberos for views, see "Set Up Kerberos for Ambari Server" in the Ambari Security Guide. After you have set up basic Kerberos for the Pig View, Pig requires that **WebHDFS Authentication** be set to `auth=KERBEROS;proxyuser=<ambari-user-principal>`.

Note

This property is only needed if the view is Custom Configured or Ambari Server is Kerberized before 2.4.0.

For example, see the following figure:

Figure 10.6. Kerberos Settings for Pig Views



10.3. Using the Pig View

Use the Pig View to:

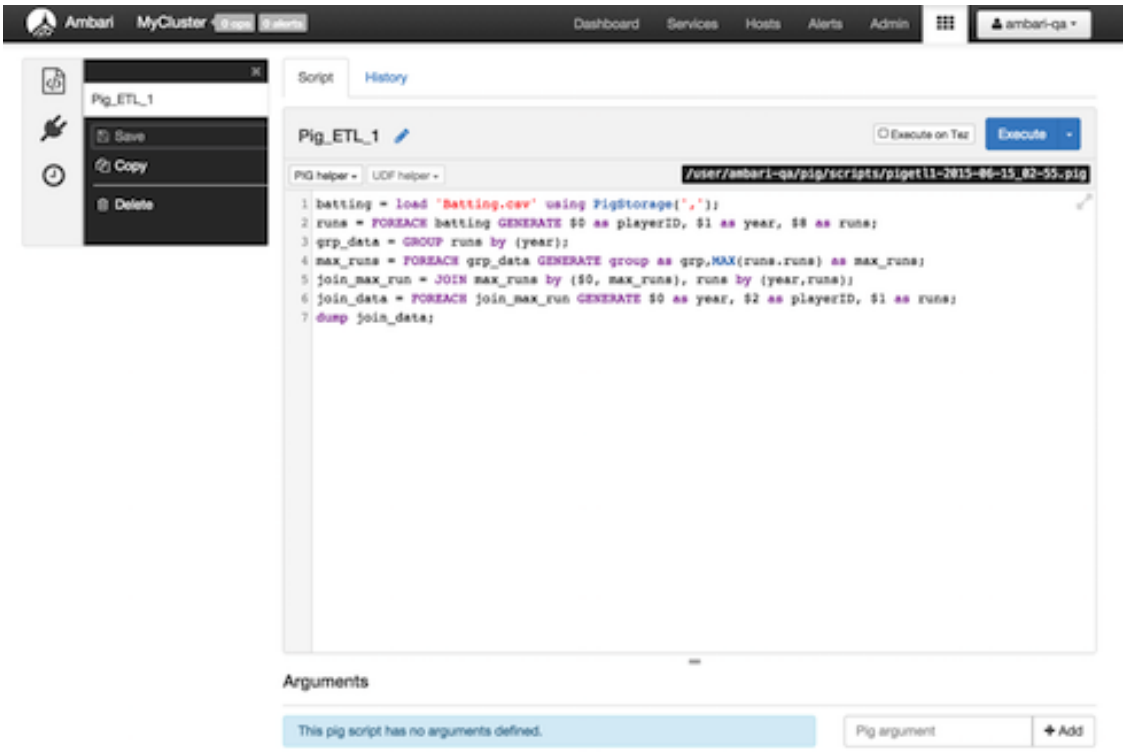
- Write Pig scripts
- Execute Pig scripts
- Add user-defined functions (UDFs) to Pig scripts
- View the history of all Pig scripts run by the current user

10.3.1. Writing Pig Scripts

Navigate to the Pig View instance Scripts page, and click **New Script** in the upper right corner of the window. Name the script in the New Script dialog box, click **Create**, and enter your script into the editor. After you have written the script, you can use the execute button on the upper right to run it. Check the box that is adjacent to the execute button to use Tez instead of the default MapReduce engine.

The following figure shows a running Pig script:

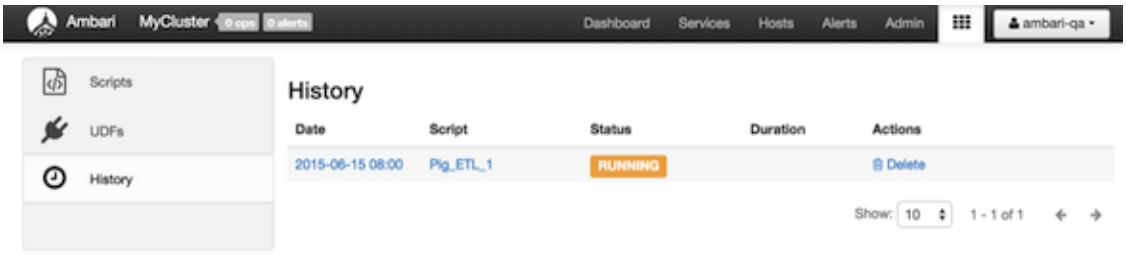
Figure 10.7. Pig Script Running in the Pig View



10.3.2. Viewing Pig Script Execution History

The History tab shows the history of Pig scripts run by the current user. A particular script in history can be clicked to open it in a new Script tab to view its details:

Figure 10.8. Pig View Script History Tab



10.3.3. User-Defined Functions (UDFs) Tab

UDFs can be added to Pig scripts by clicking **Create UDF** in the upper right corner of the UDFs window. In the Create UDF dialog box, point to a UDF in the system by specifying the name and path:

Figure 10.9. Pig View UDFs Tab

11. Using the Slider View

Slider is a framework for deploying and managing long-running applications on YARN. When applications are packaged using Slider for YARN, the **Slider View** can be used to help deploy and manage those applications from Ambari.

Important

This view has been marked deprecated.

11.1. Deploying the Slider View

Refer to the Ambari Administration guide for general information about Managing Views.

1. From the Ambari Administration interface, browse to the Views section.
2. Click to expand the **Slider** view and click **Create Instance**.
3. Enter the instance name, the display name and description.
4. Enter the configuration properties for your cluster.

Property	Description	Example
Ambari Server URL (required)	The Ambari REST URL to the cluster resource.	http://ambari.server:8080/api/v1/clusters/MyCluster
Ambari Server Username (required)	The username to connect to Ambari. Must be an Ambari Admin user.	admin
Ambari Server Password (required)	The password for the Ambari user.	password
Slider User	The user to deploy slider applications as. By default, the applications will be deployed as the "yarn" service account user. To use the current logged-in Ambari user, enter `${username}`.	joe.user or ${username}
Kerberos Principal	The Kerberos principal for Ambari views. This principal identifies the process in which the view runs. Only required if your cluster is configured for Kerberos. Be sure to configure the view principal as a proxy user in core-site.	view-principal@EXAMPLE.CO
Kerberos Keytab	The Kerberos keytab for Ambari views. Only required if your cluster is configured for Kerberos.	/path/to/keytab/view-principal.headless.keytab

5. Save the view.

12. Using the SmartSense View

The SmartSense View allows Hortonworks support subscription customers to capture diagnostic data for two purposes:

- To receive recommendations on performance, security, and operational changes based on your server hardware, HDP services deployed, and your use cases.
- To quickly capture diagnostic information about services and hosts when working with support to troubleshoot a support case.

This chapter explains:

- Configuring Your Cluster
- Creating the SmartSense View Instance
- Using the SmartSense View

12.1. Configuring Your Cluster

When you deploy a cluster with Ambari, a SmartSense View instance is automatically created as long as an Ambari Agent is deployed on the host running the Ambari Server.

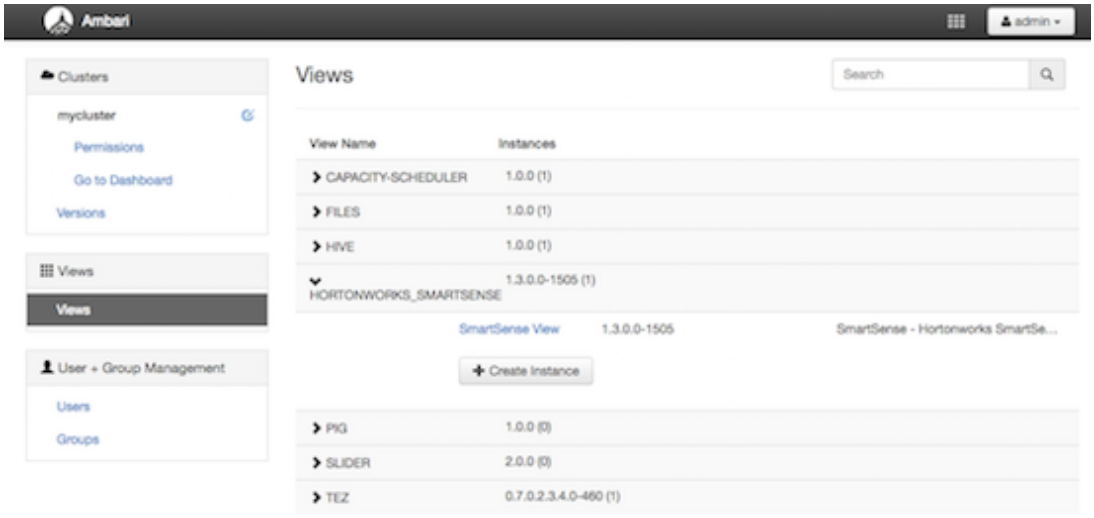
Important

If an Ambari Agent is not installed on the Ambari Server host, the view will not be automatically created, and you will have to add it manually using the instructions in Creating the SmartSense View Instance.

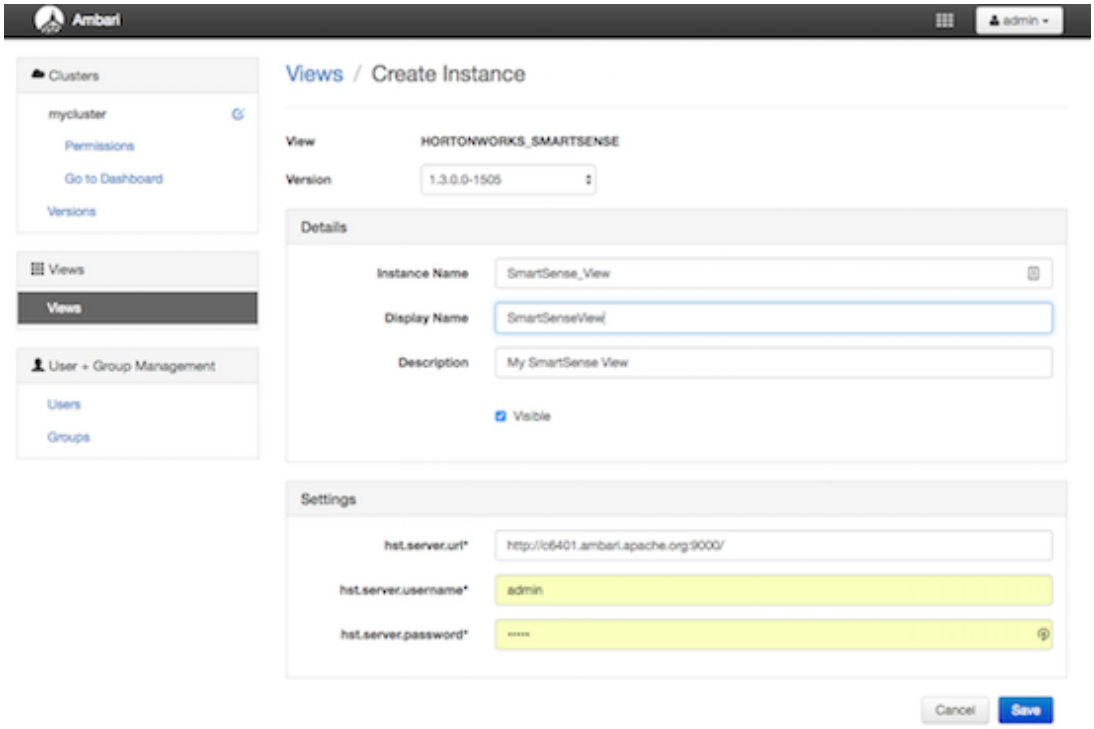
Before accessing the SmartSense View, you should enter your SmartSense user ID, account name (both are available in the Hortonworks support portal in the **Tools** tab), and email address in the SmartSense service configuration properties.

12.2. Creating the SmartSense View Instance

1. Browse to the Ambari Administration interface.
2. Click **Views**, expand the **HORTONWORKS_SMARTSENSE** menu, and click **Create Instance**:



3. On the Create Instance page, select the **Version**. If multiple SmartSense View jars are present, choose one.



4. Enter the following view instance details:

Table 12.1. SmartSense View Instance Details

Property	Description
Instance Name	This is the SmartSense view instance name. This value should be unique for all SmartSense view instances you create. This value cannot contain spaces and is required.

Property	Description
Display Name	This is the name of the view link displayed to the user in Ambari Web.
Description	This is the description of the view displayed to the user in Ambari Web.

5. Enter the following view instance settings:

Table 12.2. SmartSense View Instance Settings

Property	Description
hst.server.url	This is the HST server URL. This should be `http://<HST_host>:9000/`.
hst.server.username	The default username is 'admin'.
hst.server.password	Unless changed after installation, the default password is 'admin'.

6. Click **Save**.

12.3. Using the SmartSense View

Use the SmartSense View to:

- Capture a bundle
- Set a bundle capture schedule
- View and download captured bundles

13. Using the Storm View

Storm provides a real-time, scalable, and distributed solution for data streamed from real-time sources such as machine sensors, supporting data ingestion, processing, and real-time response. Typical use cases include automated systems that respond to sensor data by notifying support staff, or an application that places a proximity-based advertisement on a consumer's smart phone.

This chapter explains:

- Configuring Your Cluster
- Creating the Storm View Instance
- Using the Storm View

Important

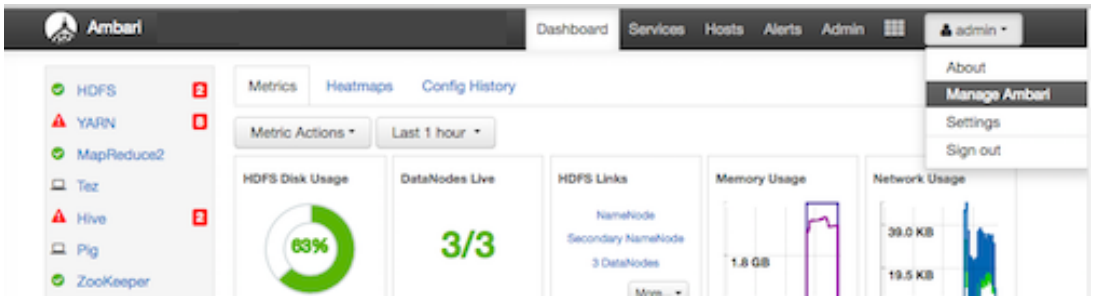
Before creating the Storm view instance, prepare your Ambari Server for hosting views. It is strongly recommended you increase the amount of memory available to your Ambari Server, and that you run additional "standalone" Ambari Servers to host the views. See Preparing Ambari Server for Views and Running Ambari Server Standalone for more information.

13.1. Configuring Your Cluster

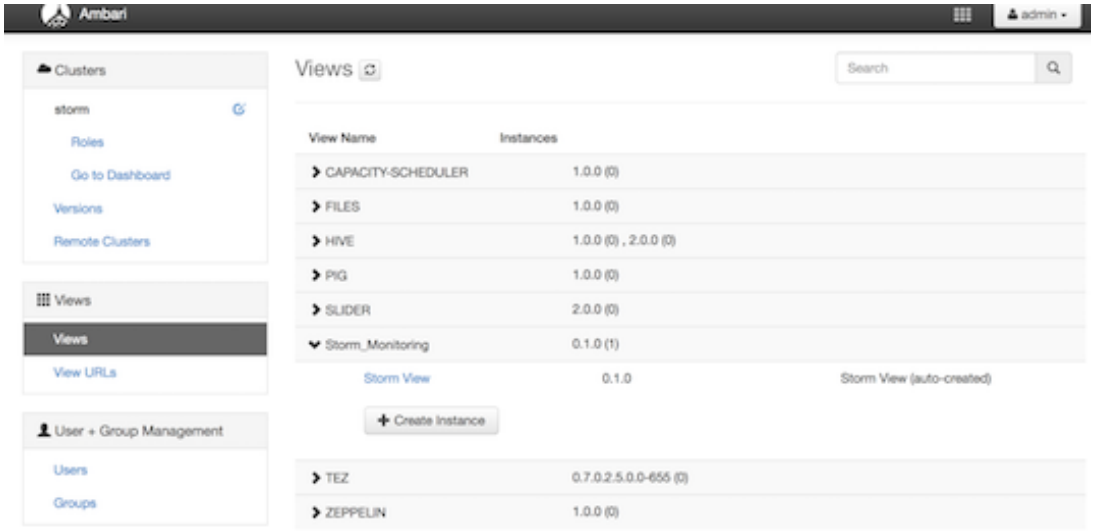
The Storm View requires that the cluster is managed by Ambari; the view utilizes the Ambari Server API.

13.2. Creating the Storm View Instance

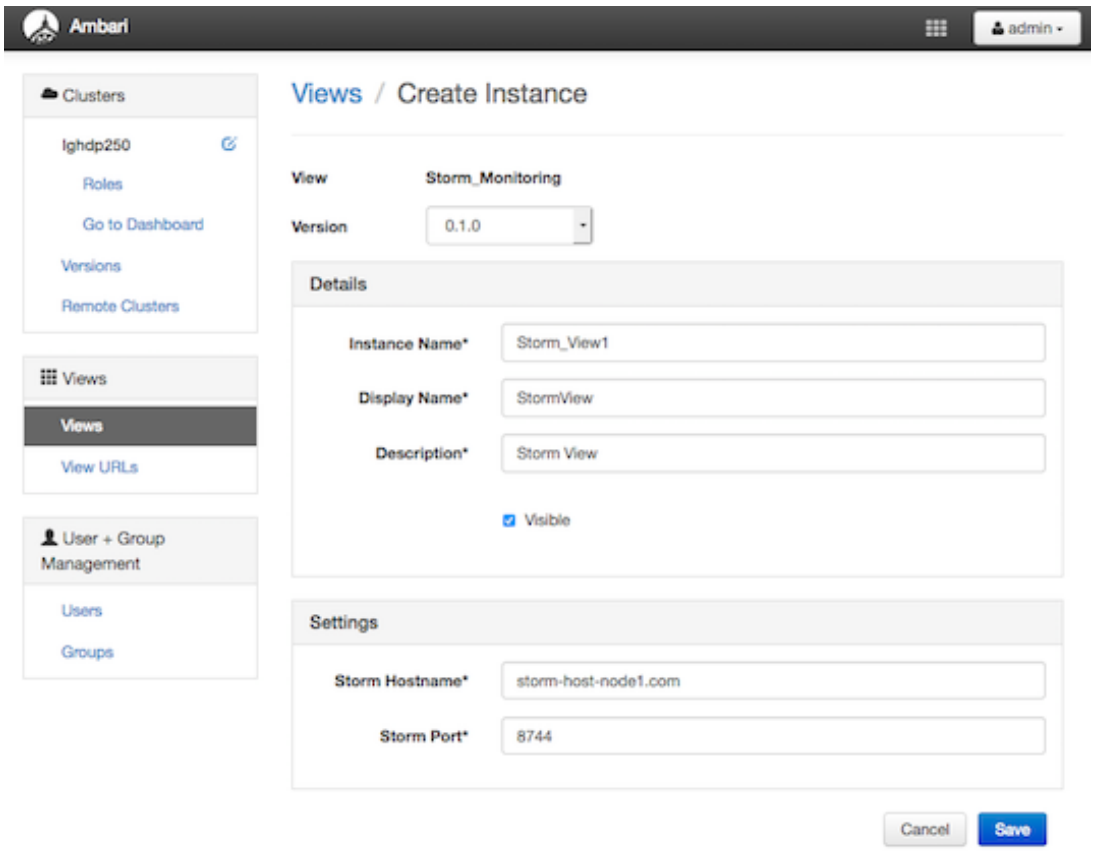
1. Browse to the Ambari Administration interface: from the dashboard, open the administrator account menu and click **Manage Ambari**:



2. Click **Views**, expand the **Storm_Monitoring** menu, and click **Create Instance**:



3. On the Create Instance page, select the **Version**. If multiple Storm View versions are present, choose one.



4. Enter the following view instance details:

Table 13.1. Storm View Instance Details

Property	Description	Example Value
Instance Name	This is the Storm view instance name. This value should be unique for all Storm view instances you create. This value cannot contain spaces, and it is a required setting.	`Storm_View1`
Display Name	This is the name of the view link displayed to the user in Ambari Web.	`StormView`
Description	This is the description of the view displayed to the user in Ambari Web.	`StormView`

5. Enter the following view instance settings:

Table 13.2. Storm View Instance Settings

Property	Description	Example Value
Storm Hostname	This is the hostname where the Storm UI Server is running.	`storm-host-node1.com`
Storm Port	This is the port where the Storm UI Server is listening.	`8744`

   Settings depend on cluster and deployment factors in your environment. You can typically leave the default settings unless you are using the Storm View with a Kerberos-enabled cluster. For more information, refer to the Ambari Security Guide and Configuring Views for Kerberos.

6. Click **Save**.

13.3. Using the Storm View

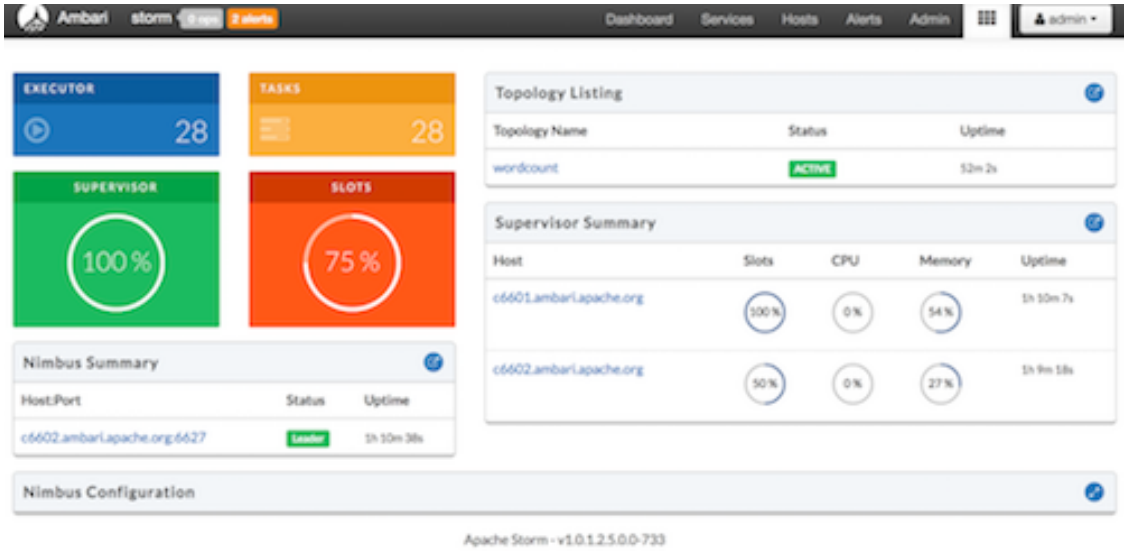
Use the Storm View for the following types of operations:

- Monitor Storm cluster status and review configuration settings.
- Monitor Storm topologies, review configuration settings, perform topology actions such as Activate, Deactivate, and Kill, and perform topology rebalancing to increase worker JVMs and component parallelism.
- Access component metrics, debug logs, and jstack outputs; debug and profile worker JVMs.

13.3.1. Monitoring Storm Cluster Status: the Cluster Summary Page

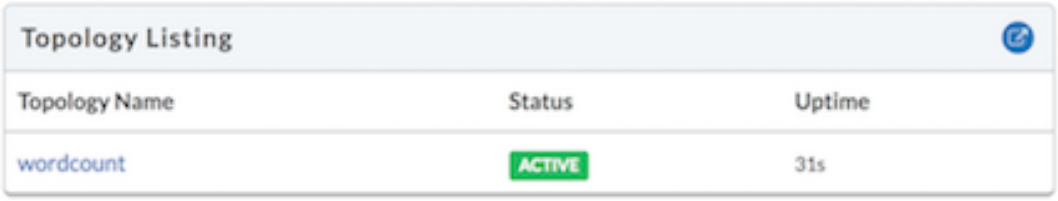
The landing page for the Storm view shows current cluster status and nimbus configuration.

It shows the available nimbus host(s), and for a nimbus HA, denotes which host is a leader. It also shows all available supervisor hosts and currently deployed topologies. Here is an example landing page:

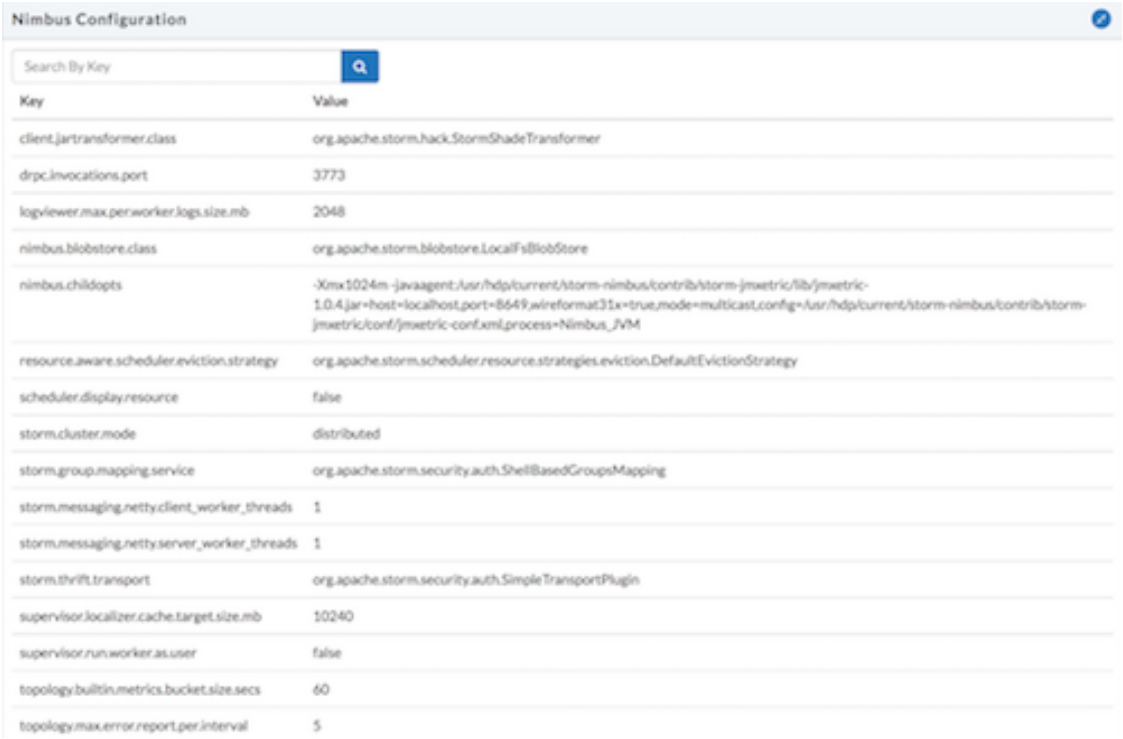


The lower left section of the summary page shows resource utilization of supervisors:

The upper right section shows the current status of the deployed topology:



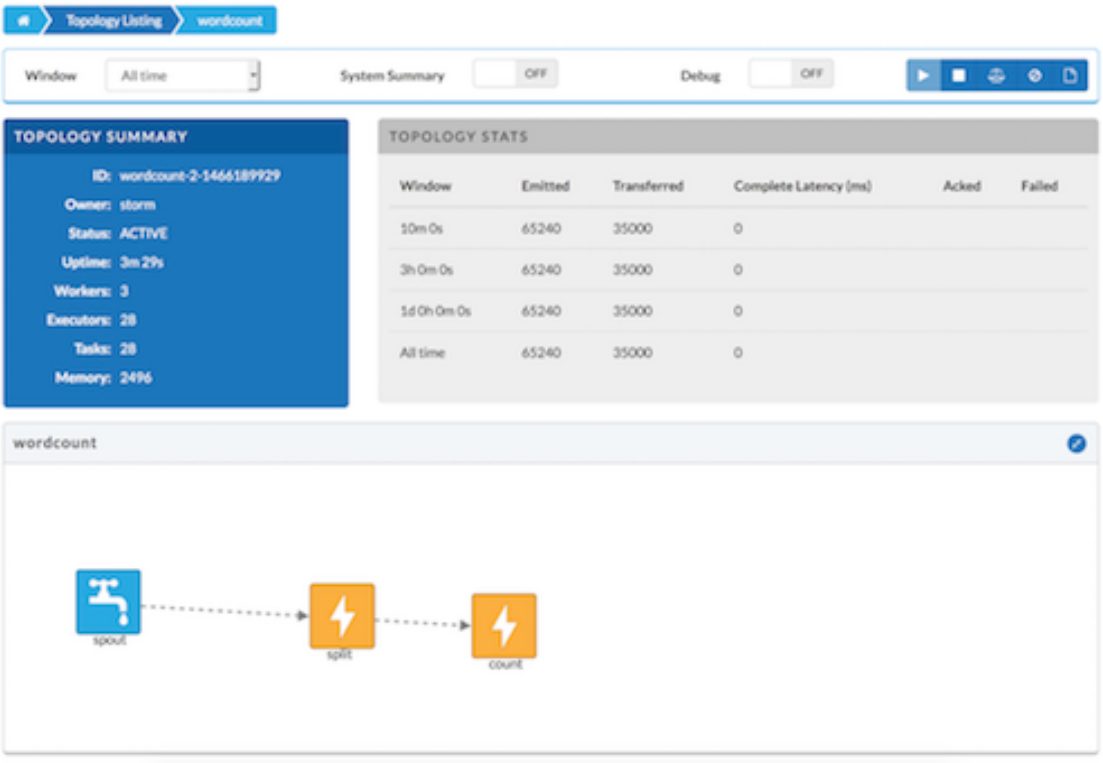
Click on the "Nimbus Configuration" section to list Storm configuration settings:



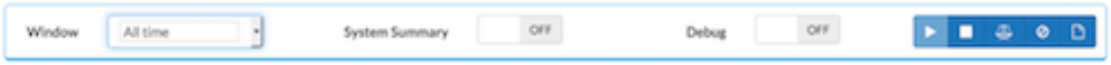
13.3.2. Monitoring Topology Status: the Topology Summary Page

The topology summary page contains metrics and directed acyclic graphs (DAG) that show deployed topology components and topology debugging features.

You can select which window for which to review metrics. By default the view will show metrics for "All Time."

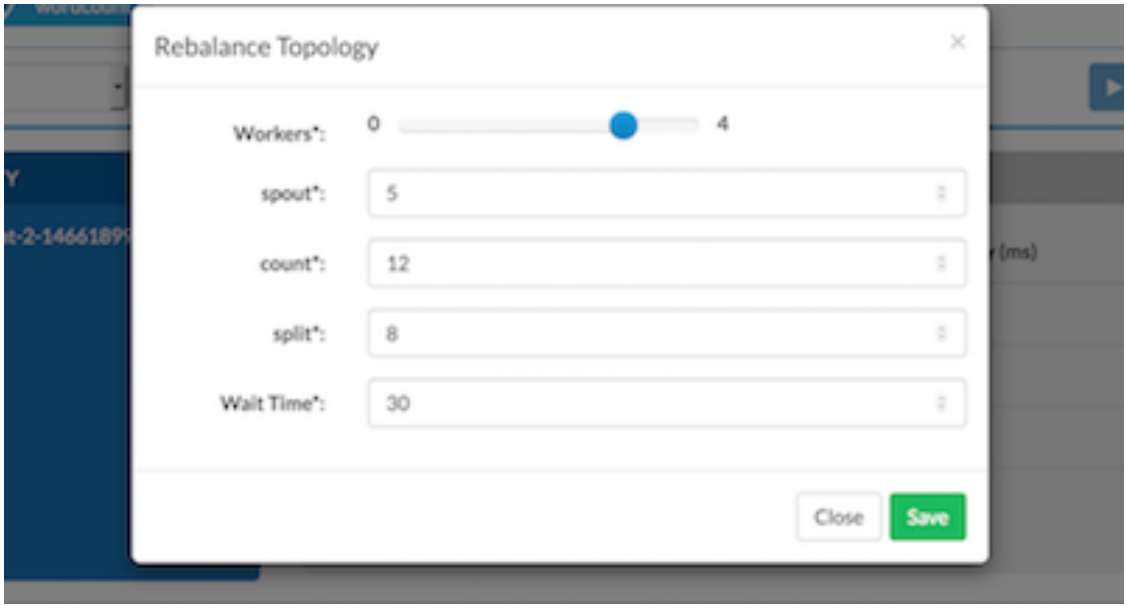


On the right side above panel, there are several topology actions buttons. These buttons allow you to perform several actions: Activate (highlights when topology status is deactivated), Deactivate, Rebalance, Kill, and Changing log level.



Rebalancing a Topology

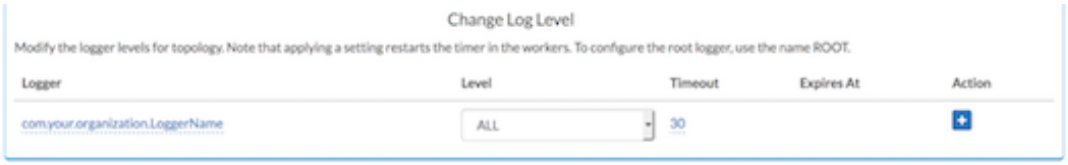
To adjust the number of workers for the topology and the parallelism of each component in the topology, use the rebalance button.



Changing the Logging Level of a Running Topology

This feature facilitates topology debugging, by allowing you to temporarily enable debug log level and see any issues in a topology.

To use this feature, edit the Logger to update the class name for which you would like to add a log level.

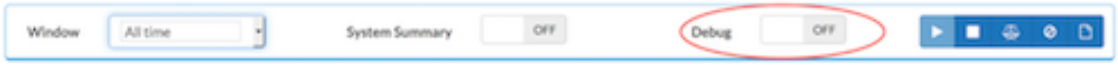


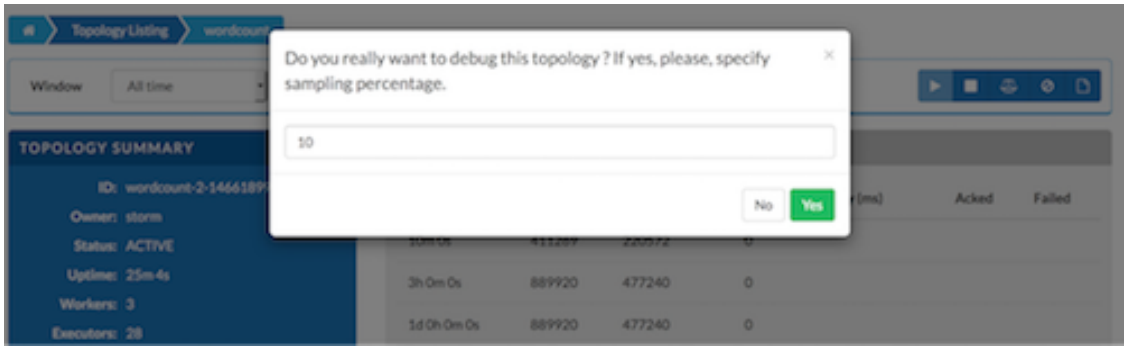
For example, if you would like to see debug logs in the count bolt of the sample word count topology supplied with Storm, add the classname as `org.apache.storm.starter.WordCount`.

Sampling Events in a Running Topology

This feature allows users to debug and see the events that are flowing through the topology, essentially sampling events from a running topology and storing them in a log file.

To use this feature, turn the Debug switch to "On":

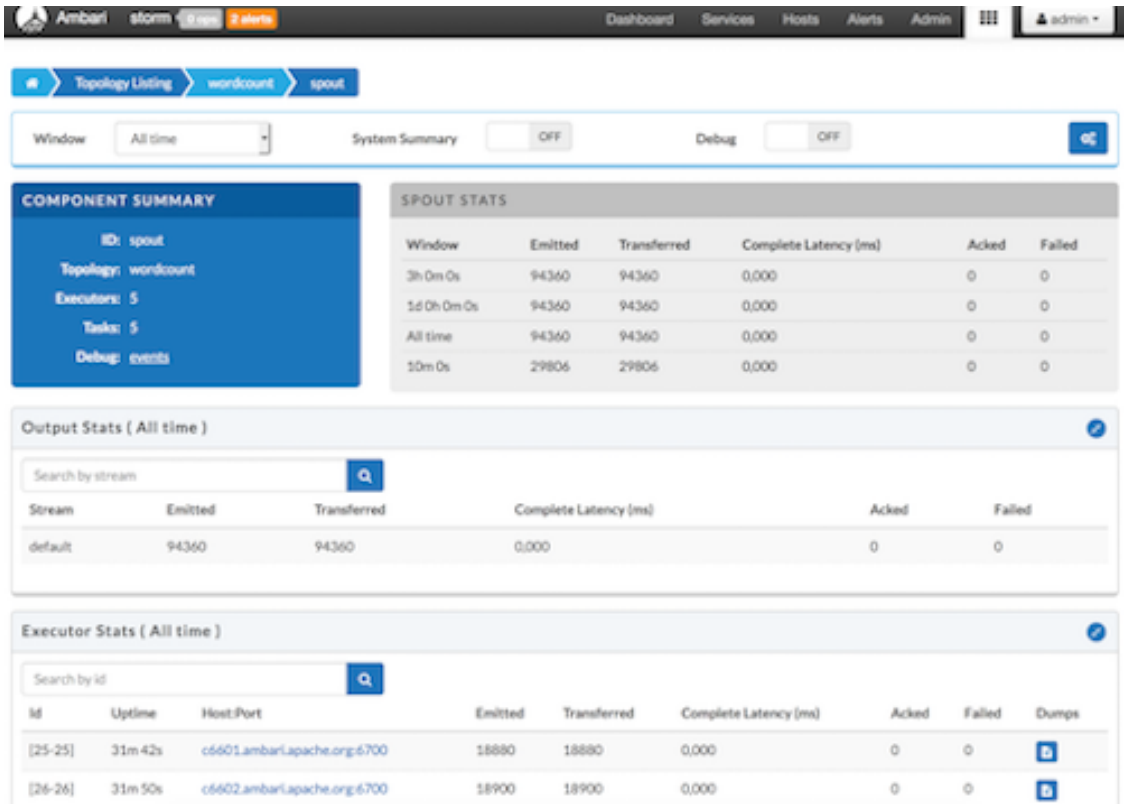




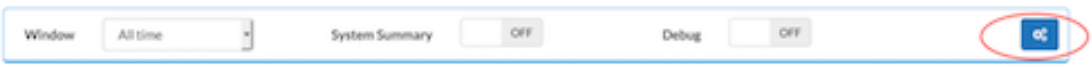
The event logger will sample the given percentage of incoming tuples and write them to the log for users to see the incoming tuples at each stage of topology. We recommend that you not set this to a higher percentage, because it can fill up the logs on disk very quickly.

13.3.3. Looking Up Configuration Values: the Component Summary Page

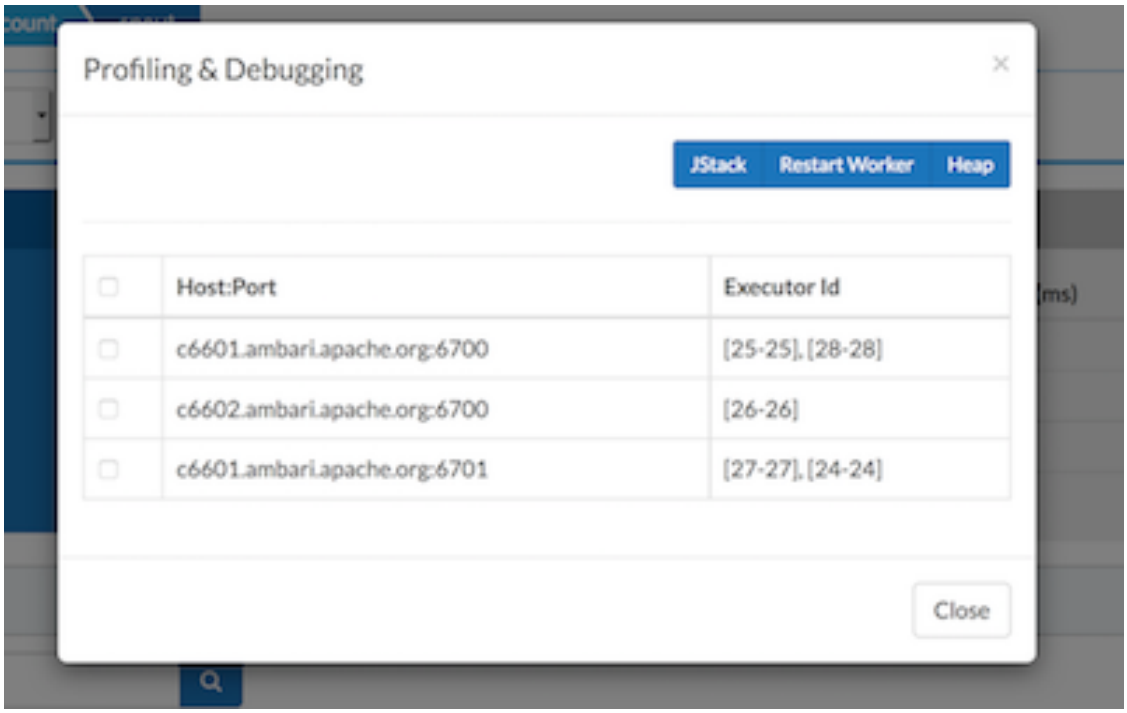
On the Component Summary page, you can drill down to a individual component in a topology to see relevant stats for the component and access debug logs and jstack outputs.



You can also debug and profile a worker JVM, by choosing the rightmost button on the Component Summary Page:



The popup window shows all worker processes running the particular spout. You can select the worker processes to take the jstack output or Heap dump, and selectively restart a worker JVM.



14. Using the Tez View

Tez is an framework for building high performance batch and interactive data processing applications. Apache Hive and Pig use the Tez framework. When you run a job such as a Hive query or Pig script using Tez, you can use the Tez View to track and debug the execution of that job. Topics in this chapter describe how to configure, deploy and use the Tez View to execute jobs in your cluster:

- Configuring Your Cluster for Tez View
- Creating the Tez View Instance
- Using the Tez View

14.1. Configuring Your Cluster for Tez View

When you deploy a cluster with Ambari, a Tez View instance is automatically created. However, you must verify that the configurations listed in the following table have been correctly set.

If you have manually deployed your cluster, you must set the properties listed in the following table to configure your cluster before you create the Tez View on your standalone Ambari server.

To configure your cluster for the Tez View:

1. Confirm the following configurations are set:

Table 14.1. Cluster Configurations for Tez View

Component	Configuration	Property	Comments
YARN	yarn-site.xml	yarn.resourcemanager.system-metrics-publisher.enabled	Enable the generic history service in the Timeline Server. Verify that this property is set to `true`.
YARN	yarn-site.xml	yarn.timeline-service.enabled	Enable the Timeline Server for logging details. Verify that this property is set to `true`.
YARN	yarn-site.xml	yarn.timeline-service.webapp.address	Value must be the `IP:PORT` on which the Timeline Server is running.

2. If you changed any settings, you must restart the YARN ResourceManager and the Timeline Server for your changes to take effect.

Important

If you do not need to reconfigure the Ambari-created Tez View, see Using the Tez View.

14.2. Creating or Editing the Tez View Instance

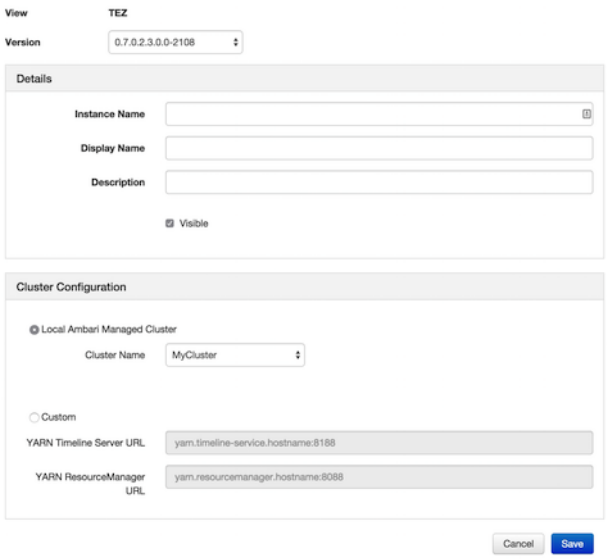
Depending on whether you must create a new Tez View instance for a manually deployed cluster or modify an Ambari-created Tez View, see one of the following sections:

- Modifying a Tez View Instance on an Ambari-Managed Cluster [85]
- Creating a New Tez View Instance on a Manually-Deployed Cluster [85]

To modify a Tez View instance on an Ambari-managed cluster:

1. Navigate to the Ambari Administration interface.
2. Click **Views** and expand the **Tez View**.
3. On the Create Instance page, change the appropriate configuration parameters.
4. Select **Local Ambari-Managed Cluster**:

Figure 14.1. Tez View Create Instance Page



Important

Secure clusters that use wire encryption (SSL/TSL) cannot use the **Local Ambari Managed Cluster** option. Instead you must configure the view as described in the instructions for manually-deployed clusters [85].

5. Click **Save**, grant Permissions on the view (see User Permissions for Tez Views), and click **Go to instance** to use the view. See Using the Tez View.

To create a new Tez View instance for a manually-deployed cluster:

1. Navigate to the Ambari Administration interface.
2. Click **Views**, expand the **Tez View**, and click **Create Instance**.

3. On the Create Instance page, select the **Version**.

4. Enter the Details (required). The Instance Name appears in the URI, the Display Name appears in the Views drop-down list, and the Description helps multiple users identify the view.

5. Scroll down to the Cluster Configuration, verify that **Custom** is checked and enter the following values, which tell the Tez View how to access resources in the cluster:

Table 14.2. Cluster Configuration Values for the Tez View in Ambari

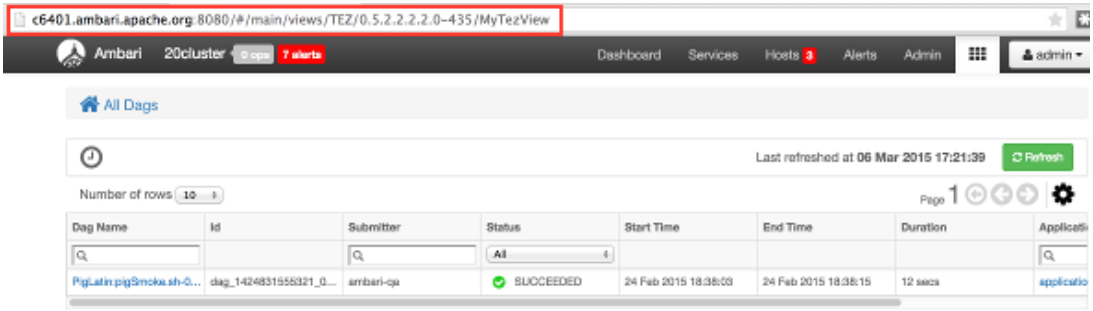
Property	Value
YARN Timeline Server URL (required)	The URL to the YARN Application Timeline Server, used to provide Tez information. Typically, this is the `yarn.timeline-service.webapp.address` property that is specified in the `etc/hadoop/conf/yarn-site.xml`. When you enter the value in the view definition, prepend "http://" to the value you find in the yarn-site.xml file. For example, `http://<timeline server host>:8188` **For wire encryption-enabled clusters:** Set this based on the value of `yarn.timeline-service.webapp.https.address` in `yarn-site.xml` When you enter the value in the view definition, prepend "https://" to the value. For example, `https://<timeline server host>:8190`
YARN ResourceManager URL (required)	The URL to the YARN ResourceManager, used to provide YARN Application data. Typically, this is the `yarn.resourcemanager.webapp.address` property that is specified in the `etc/hadoop/conf/yarn-site.xml`. When you enter the value in the view definition, prepend "http://" to the value you find in the yarn-site.xml file. For example, `http://<resourcemanager host>:8088` **Important:** If YARN ResourceManager HA is enabled, provide a comma-separated list of URLs for all the Resource Managers. **For wire encryption-enabled clusters:** Set this based on the value of `yarn.resourcemanager.webapp.https.address` in `yarn-site.xml` When you enter the value in the view definition, prepend "https://" to the value. For example, `https://<resourcemanager host>:8090`

6. Click **Save** and grant Permissions on the view (see User Permissions for Tez Views).

7. At the top of the view instance configuration page, click **Go to instance**.

8. When your browser is at the view instance page, copy the URL for the Tez View from your browser address bar:

Figure 14.2. Tez View Instance Page



9. In `tez-site.xml`, specify the URL that you copied in Step 8 as the value for the `tez.tez-ui.history-url.base` property, and save the file.

10. Restart the HiveServer2 daemon to make sure that your changes to `tez-site.xml` take effect.

To use the view, see Using the Tez View.

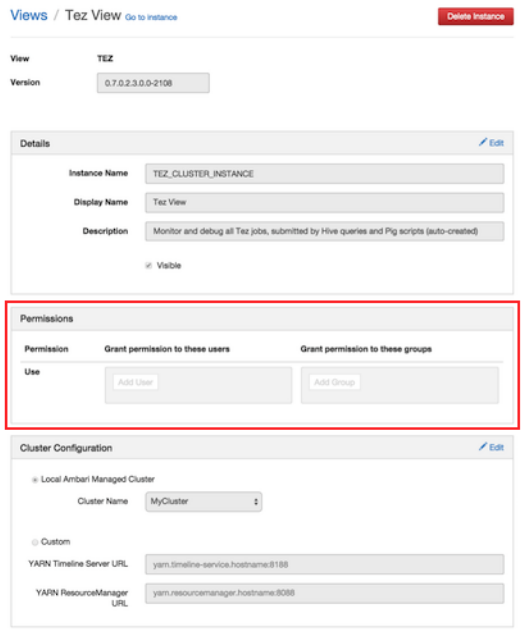
Important

If your cluster is configured for Kerberos, you must set up Ambari Server for Kerberos for the Tez View to access the ATS component. See Kerberos Setup for Tez Views.

14.2.1. User Permissions for Tez Views

After saving the Tez View instance definition, grant permission on the view for the set of users who can use the view:

Figure 14.3. Granting User Permissions to Tez Views



Note

To grant access to all Hive and Pig users, create a group that contains these users, and then grant permission to use the Tez View to that group. See also the Managing Users and Groups in Hortonworks Data Platform Apache Ambari Administration.

14.2.2. Kerberos Setup for Tez Views

To set up basic Kerberos for views, see Set Up Kerberos for Ambari Server in Hortonworks Data Platform Apache Ambari Security.

After you have set up basic Kerberos for the Tez View, you must set the following configuration properties:

1. **On the timeline server host**, set the following values for properties in the YARN configuration for Ambari-managed clusters or the `yarn-site.xml` for manually deployed clusters:

Table 14.3. Kerberos Settings for Tez Views

Property	Value
`yarn.timeline-service.http-authentication.proxyuser.${ambari principal name}.hosts`	*
`yarn.timeline-service.http-authentication.proxyuser.${ambari principal name}.users`	*
`yarn.timeline-service.http-authentication.proxyuser.${ambari principal name}.groups`	*

For example, if the Kerberos principal used for the Ambari server is `ambari-service@EXAMPLE.COM`, replace `${ambari principal name}` with `ambari-service`.

2. Restart the Timeline Server so your configuration changes take effect.

14.3. Using the Tez View

Tez provides a framework that enables human-interactive response times with Apache Hive queries and Apache Pig data transformations. The Tez View enables you to understand and debug submitted Tez jobs, such as Hive queries or Pig scripts, that are executed using the Tez execution engine.

The following sections discuss using the Tez Views to manage Hive and Pig tasks:

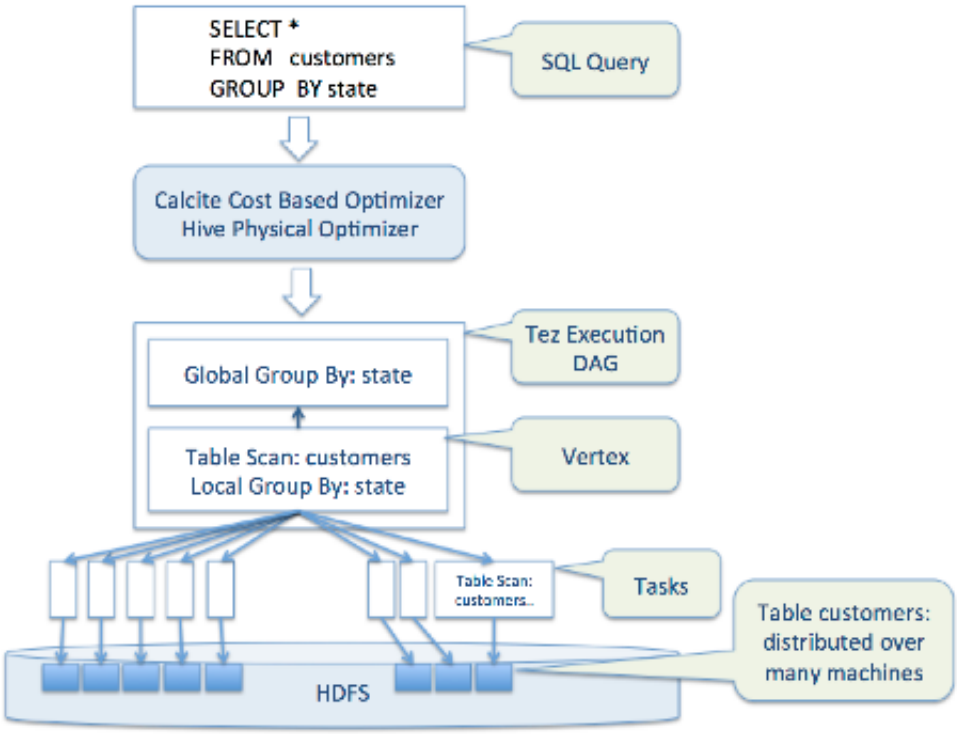
- Understanding DAGs, Vertices, and Tasks
- Identifying the Tez DAG for Your Job
- Understanding How Your Tez Job Is Executed
- Identifying Causes of Failed Jobs
- Viewing All Failed Tasks
- Using Counters to Identify the Cause of Slow-Performing Jobs

14.3.1. Understanding Directed Acyclic Graphs (DAGs), Vertices, and Tasks

To explain DAGs, vertices, and tasks, consider how Hive SQL queries are compiled and converted into a Tez execution graph also known as a DAG. A *DAG* is a collection of vertices where each vertex executes a fragment of the query or script. Directed connections between vertices determine the order in which they are executed. For example, the vertex to read a table must be run before a filter can be applied to the rows of that table.

As another example, consider when a vertex reads a user table. This table can be very large and distributed across multiple computers and multiple racks. Reading the table is achieved by running many tasks in parallel. The following figure shows the execution of a SQL query in Hive:

Figure 14.4. SQL Query Execution in Hive



14.3.2. Identifying the Tez DAG for Your Job

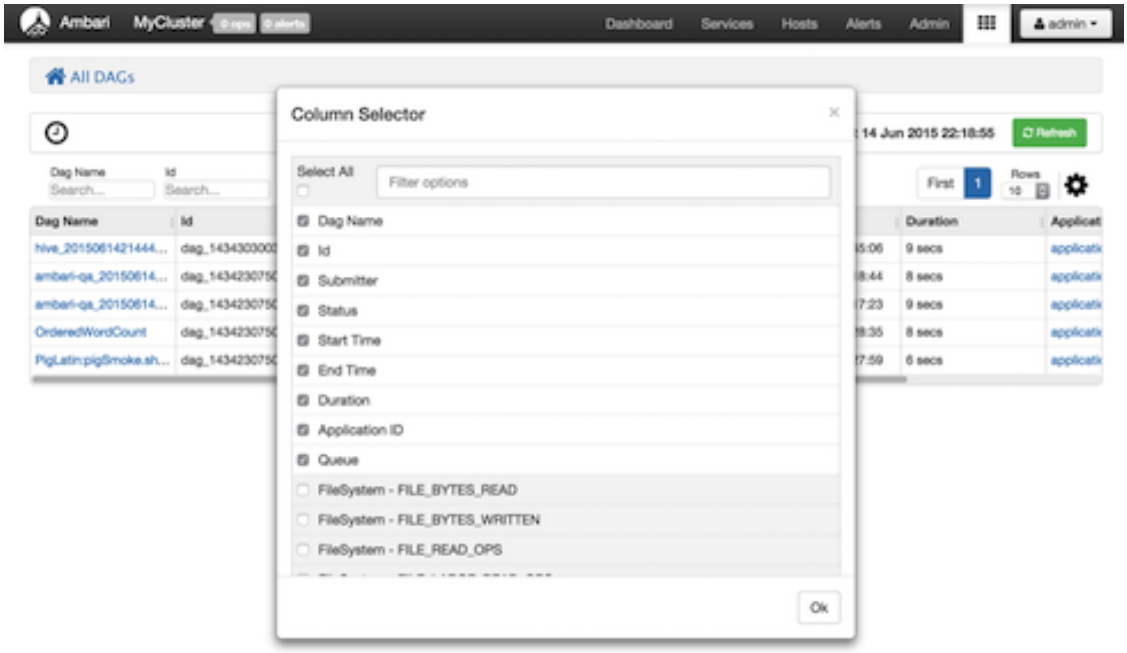
To identify the Tez DAG for your job:

1. Navigate to the Tez View instance by clicking **Go to instance** on the Tez View page in Ambari. The Tez View instance page displays a list of jobs sorted by time, listing the latest jobs first. You can search a job using the following fields:
   - **Dag Name** (DAG name for the job)
   - **Id** (DAG identifier)
   - **Submitter** (user who submitted the job)
   - **Status** (job status)
   - **Application ID**
2. When you have entered your search criteria, press **Enter**, and search results matching your criteria are returned below.

Selecting the Columns That Appear in Search Results

To select which columns are included in the Tez View search results, click the gear icon to the right of the search tool bar. A Column Selector dialog box appears where you can select which columns appear in the search results. Select the columns, and click **Ok** to return to the Tez View:

Figure 14.5. Tez View Column Selector Dialog Box



Note

To search for columns, use the search well at the top of the Column Selector dialog box. Check **Select All** to include all columns in your search results and uncheck it to clear all of your column selections.

Understanding Tez View Job Status

The following table explains the job status field that is returned for all search results returned in the Tez View:

Table 14.4. Tez Job Status Descriptions

Status	Description
Submitted	The DAG is submitted to Tez but is not running.
Running	The DAG is currently running.
Succeeded	The DAG completed successfully.
Failed	The DAG failed to complete successfully.
Killed	The DAG was stopped manually.
Error	An internal error occurred when executing the DAG.

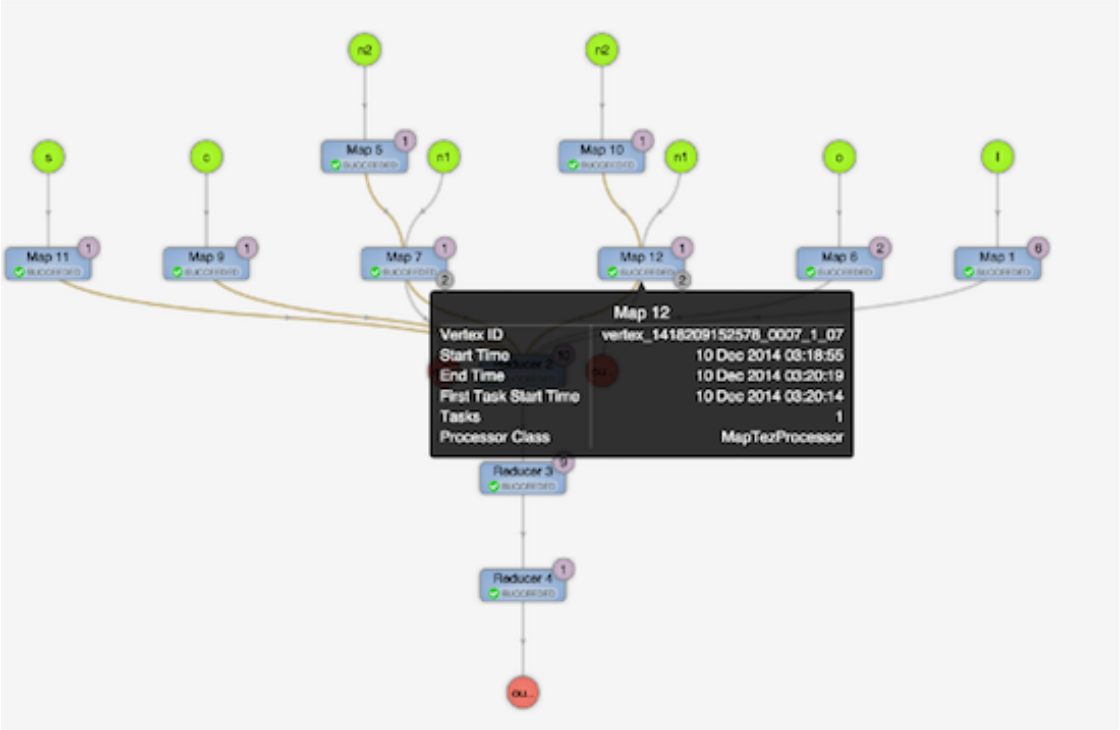
14.3.3. Understanding How Your Tez Job Is Executed

The Tez View enables you to gain insight into the complexity and the progress of executing jobs.

The View tab shows the following:

- DAG graphical view
- All vertices
- Tasks per vertex on top right of the vertex
- Failed vertices display in red, successful vertices display in green
- Mouse over vertices to view timeline details

Figure 14.6. View Tab in Tez View



The View Tab enables you to investigate the vertices that have failures or are taking a long time.

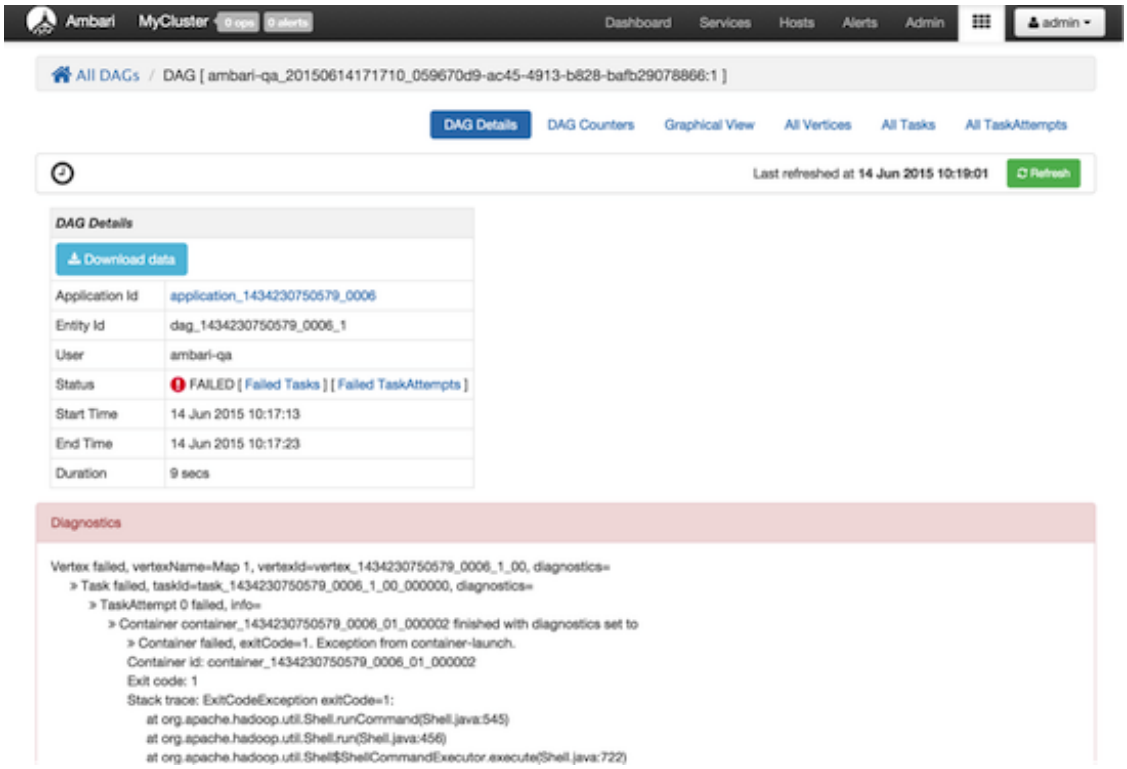
14.3.4. Identifying Causes of Failed Jobs

The Tez View enables you to quickly find and report errors. When a Tez task fails, you must:

- Identify why the task failed
- Capture the reason for task failure

When a Tez task fails, the DAG Details tab explains the failure:

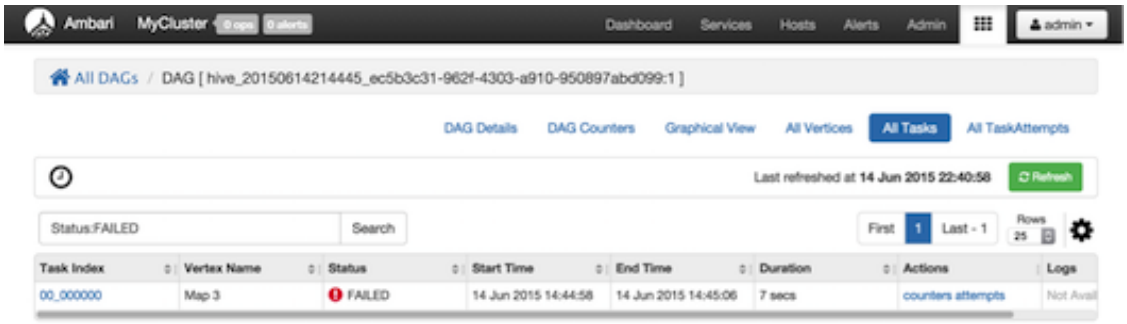
Figure 14.7. DAG Details Window



14.3.5. Viewing All Failed Tasks

Multiple task failures may occur. The Tez View All Tasks tab enables you to view all tasks that failed and examine the reason and logs for each failure. Logs for failed tasks, but not for aborted tasks are available to download from this tab:

Figure 14.8. Tez View All Tasks Tab



14.3.6. Using Counters to Identify the Cause of Slow-Performing Jobs

The Tez View shows counters so you can understand why a task performs more slowly than expected. Counters help you better understand the task size and enable you to locate anomalies. Elapsed time is one of the primary counters to look for.

Counters are available at the DAG, vertex, and task levels:

Figure 14.9. Tez View DAG-Level Counters Tab

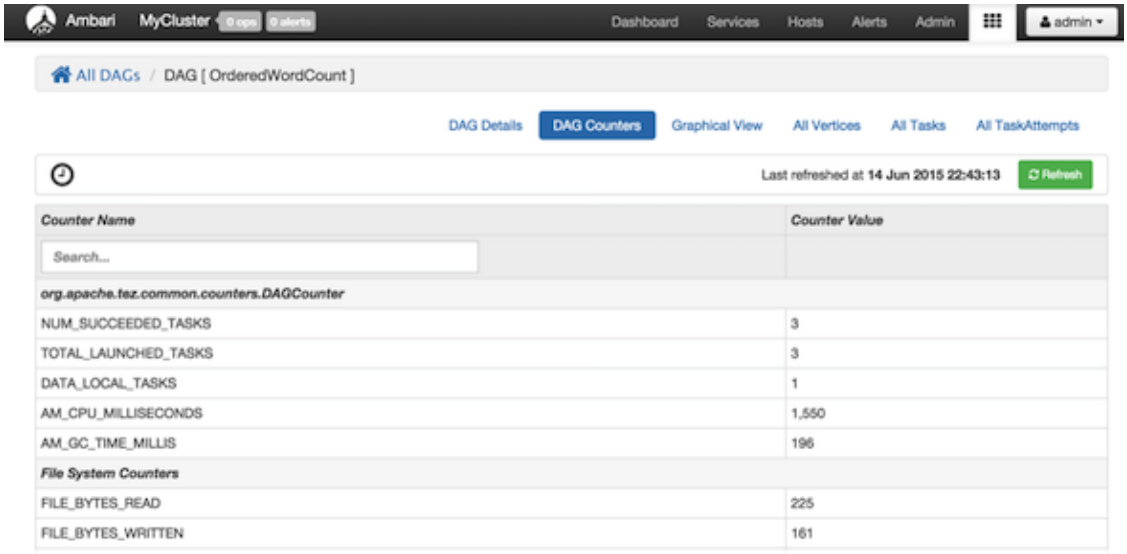


Figure 14.10. Tez View Vertex-Level Counters Tab

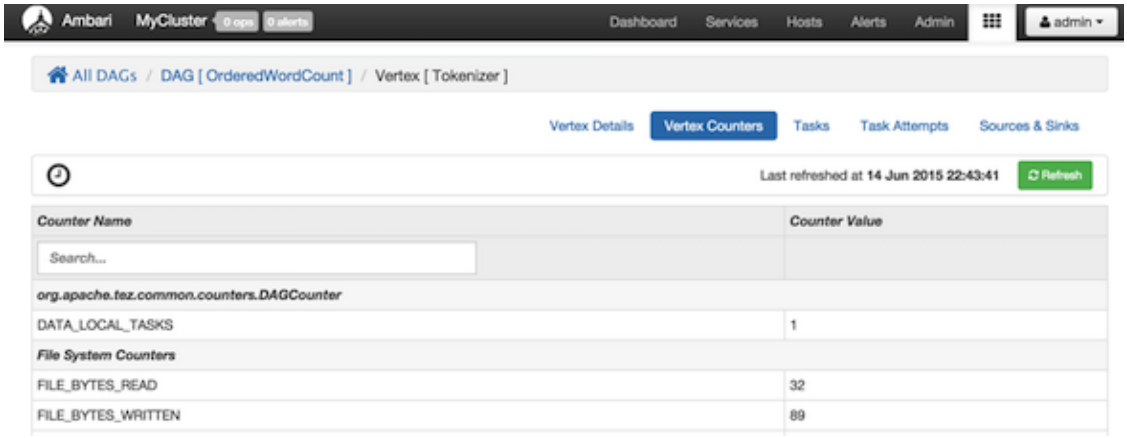
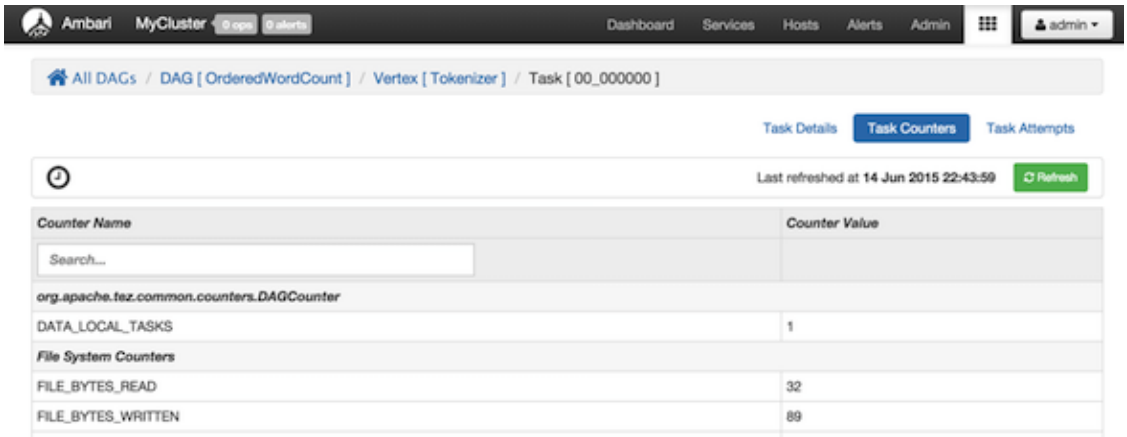


Figure 14.11. Tez View Task-Level Counters Tab



Monitoring Task Progress for Jobs

The Tez View shows task progress by increasing the count of completed tasks and total tasks. This enables you to identify the tasks that might be "hung" and to understand more about long-running tasks.

15. Using Workflow Designer View -Tech Preview

Ambari includes the Workflow Designer View, which supports monitoring and scheduling jobs on the cluster.

Important

Workflow Designer View is a Tech Preview feature. Expect Workflow Designer View documentation in a future Ambari release.

16. Using Zeppelin View - Tech Preview

Ambari includes the Zeppelin View, which supports creating and editing scripts that run data streaming jobs.

Important

Zeppelin View is a Tech Preview feature. Expect Zeppelin View documentation in a future Ambari release.