# Hortonworks Data Platform

## Apache Ambari Views

(February 8, 2018)

docs.hortonworks.com

# Hortonworks Data Platform: Apache Ambari Views



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.

# Table of Contents

# List of Figures

# List of Tables

# List of Examples

# 1. Understanding Ambari Views

Apache Ambari includes the Ambari Views Framework, which enables developers to create UI components, or Views, that "plug into" the Ambari Web interface. Ambari automatically creates and presents to users some instances of Views, if the service used by that View is added to the cluster. For example, if Apache YARN service is added to the cluster, the YARN Queue Manager View displays to Ambari web users. In other cases, the Ambari Admin user must manually create a view instance.

Developing and using Views enables you to extend and customize the Ambari web to meet your specific needs.

Using Views also extends your Ambari implementation to allow third parties to plug in new resource types, along with APIs, providers, and UIs to support them. Views are deployed on the Ambari Server, which enables Ambari Admins to create View instances and set access privileges for users and groups.

The following sections describe the basics of Views and how to deploy and manage View instances in Ambari:

**More Information**

Ambari Views Framework

Views Terminology [1]

Understanding Views Development, Persona, Versions, and Deployment [2]

## 1.1. Views Terminology

The following are Views terms you should be familiar with:

| | |
|---|---|
| Views framework | The core framework that is used to develop a View: similar to a Java web application. |
| View definition | The View resources and core View properties, such as name, version, and any necessary configuration properties. Ambari reads View definition during deployment. |
| View package | A bundle of View client and server assets (and dependencies) that is ready to deploy to Ambari. |
| View deployment | The process of instantiating a View instance in Ambari, which makes that View available to Ambari Admins for creating instances. |
| View name | The unique identifier for a View. A View can have one or more versions. The name is defined in the View Definition (created by the View Developer) and built into the View Package. |
| View version | The uniquely named version of a View. Multiple versions of a View (uniquely identified by View name) can be deployed to Ambari. |

| | |
|---|---|
| View instance | The instantiation of a specific View version. Instances are created and configured by Ambari Admins and must have a unique View instance name. |
| View instance name | The unique identifier of a specific instance of a View. |
| framework services | View context, instance data, configuration properties, and events |

# 1.2. Understanding Views Development, Persona, Versions, and Deployment

Views are basically web applications that can be "plugged in to" Ambari. Just like a typical web application, a View can include server-side resources and client-side assets. Server-side resources, which are written in Java, can integrate with external systems (such as cluster services) and expose REST end-points that are used by the view. Client-side assets, such as HTML, JavaScript, and CSS, provide the UI for the view that is rendered in the Ambari web interface.

**Development**

Ambari Views Framework Ambari exposes the Views Framework as the basis for View development. The Framework provides the following:

- Method for describing and packaging a View
- Method for deploying a View
- Framework services for a View to integrate with Ambari
- Method for managing View versions, instances, and permissions

The Ambari Views framework is separate from Views themselves. The framework is a core feature of Ambari that you use to create, deploy, integrate, and manage your own, custom views.

You develop and deliver a view by performing the following tasks:

- Develop the View (similar to how you would build a web application)
- Package the View (similar to a WAR)
- Deploy the View to Ambari (using the Ambari Administration interface)
- Create and configure instances of the View (performed by Ambari Admins)

**Persona**

Three user persona interact with Views:

| | |
|---|---|
| View developer | Person who builds the front end and back end of a View and uses the framework services available during development. The developer |

| | |
|---|---|
| | creates the View, resulting in a View package that is delivered to an Ambari Admin. |
| Ambari Admin | Ambari user that has Ambari Admin privilege and uses the Views Management section of the Ambari Administration interface to create and managing instances of Views. Ambari Admin also deploys the View packages delivered by the View developer. |
| View user | Ambari user that has access to one or more Views in the Ambari web (basically, the end user). |

### Versions

Each View must have a unique name, although it can have one or more View versions. Each View name and version combination is a single *View package*. After a View package is deployed, Ambari Admins can create *View instances*, each of which is identified by a unique View instance name. The Ambari Admin can then set access permissions for each View instance.

### Deployment

Views can be deployed and managed in the *operational Ambari Server*, the Ambari Server operating your cluster. Alternatively, 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 the operational Ambari Server. You can run one or more separate standalone Ambari Server instances to scale-out your solution for handling a large number of users.

The following table provides details about the Ambari views currently available to you, including two that have Technical Preview status.

| View | Automatically Created? | Description | HDP Stacks | Required Services |
|---|---|---|---|---|
| Using YARN Queue Manager View [75] | Yes | Provides a visual way to configure YARN capacity scheduler queue capacity. | HDP 2.3 or later | YARN |
| Using Files View [90] | Yes | Allows you to browse the HDFS file system. | HDP 2.2 or later | HDFS |
| Using Hive View 2.0 [92] | Yes | Exposes a way to find, author, execute and debug Hive queries. | HDP 2.3 or later | HDFS, YARN, Hive |
| Migrating Hue Artifacts to Ambari Views [13] | No | Supports migrating Hue artifacts to an Ambari View. | HDP 2.4 or later | Hue |
| Using Pig View [105] | No | Provides a way to author and execute Pig Scripts. | HDP 2.2 or later | HDFS, Hive, Pig |
| Using Slider View [107] | No | A tool to help deploy and manage Slider-based applications.<br><br>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 |

| View | Automatically Created? | Description | HDP Stacks | Required Services |
|---|---|---|---|---|
| Storm | No | Supports monitoring Storm cluster status and topologies.<br><br>This view has been marked deprecated. | HDP 2.5 or later | Storm |
| Using Tez View [115] | 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 |

Subsequent chapters in this guide describe tasks performed by an Ambari Administrator to make Views available to users in their Ambari-managed cluster. This guide does not describe View development and packaging. You can learn more about the Ambari Views Framework from the following resources:

**More Information**

Running Ambari Server Standalone [7]

https://cwiki.apache.org/confluence/display/AMBARI/Views

https://github.com/apache/ambari/tree/trunk/ambari-views/examples

https://github.com/apache/ambari/tree/trunk/contrib/views

# 2. Administering Ambari Views

An Ambari Administrator should review this chapter and use the instructions in each of the following sections to configure an Ambari-managed cluster for using Views.



**Audience**

Read this chapter if you have the Ambari Administrator role. You must have permissions in the Ambari Administrator role to perform tasks described in this chapter.

**More Information**

Role Comparison Chart

# 2.1. Preparing Ambari Server for Views

**Prerequisites**

Before you begin to work with Ambari views:

- Review the amount of memory available to the Ambari server that hosts views for your cluster.
- Review whether your Ambari server is configured for HTTPS.
- **Increase Available Memory to the Ambari Views Server**

  You must increase the amount of memory available to the Ambari server hosting views. This is particularly true if you intend to deploy and use multiple views concurrently.

  To increase the memory available to the Ambari Views server:

**Steps**

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 value:

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

3. Restart the server:

```
ambari-server restart
```

- **Review Number of Expected Concurrent Users**

  Consider the following guidance when planning for Views user capacity.

  An Ambari Views server:

  - On an 8-core box with 16GB of RAM and client.threadpool.size.max = 100 can handle approximately 40 concurrent users
  - On an 16-core box with 32GB of RAM and client.threadpool.size.max = 100 can handle approximately 60 concurrent users

  The recommended best practice is scaling multiple Ambari Views servers horizontally, behind a load balancer.

- **Configure a Trust Store**

  *If your Ambari Server instance is configured for HTTPS*, you must configure a trust store so that the deployed views accept the certificate used by the Ambari Server during API communications.

  To configure such a trust store:

  **Steps**

  1. On the Ambari Server, create a new keystore to contain the server's HTTPS certificate:

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

  2. When prompted, trust the certificate by typing **yes**.

  3. Configure the server to use the 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.
```

4. Restart the server:

```
ambari-server restart
```

- **(Optional) Decrease Timeout Value**

  The views.request.read.timeout.millis property in `/etc/ambari-server/conf/ambari.properties` sets the timeout value for requests made by Ambari views to non-ambari services, such as webHcat, or Hive. By default, views.request.read.timeout.millis is set to 10 seconds.

  The views.ambari.request.read.timeout.millis property in `/etc/ambari-server/conf/ambari.properties` sets the timeout values for requests made by Ambari views to Ambari services. By default, views.ambari.request.read.timeout.millis is set to 5 seconds.

  Usually no action is required. However, if you experience timeouts, or long wait times, you can decrease the values for each of these properties to lower response times.

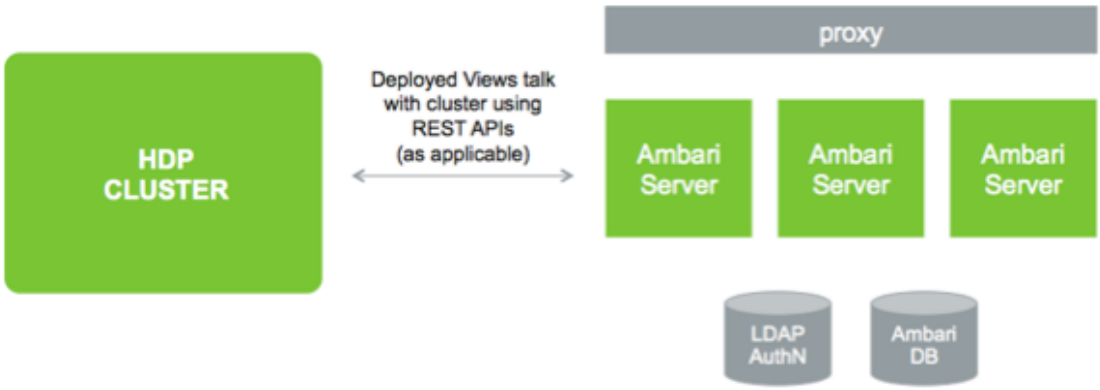
**More Information**

Set Up SSL for Ambari

# 2.2. Running Ambari Server Standalone

A recommended strategy that limits user access to your operational Ambari Server while managing a large number of Views users is to set up one or more standalone Ambari Servers. You can configure your operational Ambari Server as a remote cluster, then use the Remote Cluster option when configuring each view instance. A diagram of this strategy follows:

**Figure 2.1. Configuring Views with your HDP Cluster**




**More Information**

Registering a Remote Cluster

Configuring View Instances

## 2.2.1. Prerequisites For Standalone Ambari Servers

When setting up multiple standalone Ambari Server instances, you must be aware of the following requirements:

- All Ambari Server instances should be the same version.
- All Ambari Server instances should access the same underlying database.

  Ensure that it is *not* the same database that is being used by the Operational Ambari Server that is managing the HDP cluster.
- The Ambari database should be scaled and highly available, independent of Ambari Server.
- For an external authentication source such as LDAP or Active Directory, Ambari Server authentication should be identical for all instances.
- If the cluster that Views users access is Kerberos-enabled, you must configure Ambari and the views for Kerberos.
- You must run each standalone Ambari Server instance behind a reverse proxy.

**More Information**

Running Standalone Ambari Server Instances Behind a Reverse Proxy

## 2.2.2. Setting Up Standalone Ambari Server Compared with Setting Up Operational Ambari Server

Setting up a standalone Ambari Server instance is very similar to setting up an operational Ambari Server instance. Many of the steps are the same, with one key exception: you do not install a cluster using a standalone server instance. A standalone Ambari Server instance

does not manage a cluster and does not deploy or communicate with Ambari Agents; instead, a standalone Ambari Server runs as web server instance, serving views for users.

The following table compares the high-level tasks required to set up an operational Ambari Server and 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)<br>**Important**<br>Do not share the DB with an Operational Ambari Server. |
| 3 | Configure external LDAP authentication | Configure external LDAP authentication |
| 4 | Install cluster | *Do not* install cluster |
| 5 | Deploy views | Deploy views |
| 6 | Create and configure view instances | Create and configure view instances |
| 7 | | (Optional) Repeat for each Ambari Server instance |
| 8 | | (Optional) Set up proxy for Ambari Server instances |
| 9 | | (Optional) Set up SSL for Ambari |

**More Information**

Hortonworks Data Platform Apache Ambari Installation

Configure external LDAP authentication

Set up proxy for Ambari Server instances

Set up SSL for Ambari

### 2.2.3. Running Standalone Ambari Server Instances Behind a Reverse Proxy

If you require many users to access Ambari views, you should install and run multiple standalone Ambari Server instances behind a reverse proxy. In this case, the reverse proxy must honor *session affinity,* meaning that after a session is established, the reverse proxy routes each subsequent request to the same Ambari server instance. Depending on the reverse proxy implementation, you can achieve session affinity in several different ways, including hashing client IP and using the JSESSIONID header.

**Important**

Using a reverse proxy is supported only for standalone Ambari Server instances.

Using multiple, operational Ambari Sever instances behind a reverse proxy *is not* supported.

## 2.3. Configuring View Instances

When you create a View instance, you specify some basic configuration information about the view and you configure the view to communicate with a cluster. Based on the resources

managed by your Ambari Server, choose one of three options when completing the Cluster Configuration section; Local Cluster, Remote Cluster, or Custom. Use the following descriptions to guide your choice.

Local Cluster — If you are configuring a view instance in an Ambari Server that is also managing a cluster, you can select **Local Cluster**. When you select this option, Ambari automatically determines the cluster configuration properties required.

Remote Cluster — If your Ambari Server is not managing a cluster, then you must select either **Remote Cluster** or **Custom**.

If you plan to configure a view to work with a cluster that is remote from an Ambari Server *and* that cluster is being managed by Ambari, you should select **Remote Cluster**.

Registering a Remote Cluster enables the Remote Cluster option. When you select the **Remote Cluster** option for a view instance, Ambari automatically determines the cluster configuration properties required for the view instance. Be sure the Remote Cluster includes all services required for the view you are configuring.

Custom — If your cluster is remote from and not being managed by the Ambari Server running the view, you must select **Custom** and then manually configure the view to work with the cluster.

You can use the following table to help determine which options are available for view configuration:

| If you are working in this scenario... | Choose this option... |
|---|---|
| Your cluster is managed by a local Ambari Server that is also running the view | Local Cluster |
| Your cluster is managed by Ambari and your cluster<br><br>is remote from the standalone Ambari Server running the view | Remote Cluster |
| Your cluster is remote from the standalone Ambari Server running the view and your cluster is not managed by Ambari. | Custom |

**More Information**

Registering Remote Clusters

# 2.4. Creating View Instances

To create a View instance:

**Steps**

1. On the **Ambari Admin** page, browse to a View and expand it.
2. Click **Create Instance**.
3. Provide the following information:

| Item | Required? | Description |
|---|---|---|
| View version | Yes | Exact version to instantiate |
| Instance name | Yes | Name unique to the selected View |
| Display label | Yes | Readable display name of the View instance in Ambari Web |
| Description | Yes | Readable description of the View instance in Ambari Web |
| Visible | No | Whether the View is visible or not visible to the end-user in Ambari Web<br><br>Use this property to temporarily hide a view from users. |
| Settings | Maybe | Depending on the View, a group of settings that can be customized<br><br>If a setting is required, you are prompted to provide the required information. |
| Cluster configuration | Maybe | Depending on the View, you can choose a local or remote cluster, or manually configure a custom View. |

If Ambari has a cluster configured that will work with the View instance, then the choice of **Local Cluster** will be available. If you have registered one or more Remote Clusters, then the choice of **Remote Cluster** will also be available. If neither Local or Remote clusters are available, you will have to enter the **Custom** configuration manually.

**More Information**

Registering Remote Clusters

Configuring View Instances [9]

# 2.5. Migrating View Instance Data

If you have more than one instance of the same Ambari View, you can migrate view data (for example, entity data, instance data, and View use permissions) from one instance to another. This is useful when a new view version is released and you want to migrate view data from the previous version to the newer version.

### Important

Migrating view data between instances is supported *only for* Hive, Pig and Tez views.

For example, consider a case to migrate from view `INSTANCE A` (the source view instance) to view `INSTANCE B` (the target view instance). In this case, you run the following command:

```
curl -v -u admin:admin -X PUT -H "X-Requested-By:1" http://
AMBARI_SERVER_HOST:8080/api/v1/views/VIEW_NAME/versions/TARGET_VIEW_VERSION/
instances/INSTANCEB/migrate/SOURCE_VIEW_VERSION/INSTANCEA
```

The command values are as follows:

- `AMBARI_SERVER_HOST` is the Ambari Server host name or IP address.
- `VIEW_NAME` is the name of the view.

- `TARGET_VIEW_VERSION` is the version of the target view.
- `SOURCE_VIEW_VERSION` is the version of the source view.

  For example, if you are migrating from version 1.0.0 to 1.0.1, your `SOURCE_VIEW_VERSION` is 1.0.0 and `TARGET_VIEW_VERSION` is 1.0.1.

# 2.6. Creating View URLs

After creating a View instance, you should create a URL by which to access it, based on the view name, version, and instance name. You can also create a short URL of your choosing. You can copy and embed View URLs to provide user access to specific view instances.

To create a View URL:

**Steps**

1. In the **Ambari Admin** page, browse to the **View URLs** section.
2. Click **Create New URL**.
3. Enter a URL name, select the view, select the instance, and (optionally) type a short URL.

   Short URLs must include only lowercase, alphanumeric characters.
4. Click **Save**.

# 2.7. Setting View Permissions

By default, a new View instance has no permissions set. An Ambari Admin must specify which users and groups can use the View or, on a local cluster, specify permissions based on cluster roles. An Admin can also set permissions other than those required to use a View.

To set permissions for users and groups to use a view:

**Steps**

1. Browse to a view and expand it.
2. Click the name of the view instance you want to modify.
3. In the **Permissions** section, click the **Users or Groups** control.
4. Modify the user and group lists, as appropriate.
5. Click the check mark to save changes.

The Views Framework provides a way for view developers to specify custom permissions, beyond just the default Use permission. If custom permissions are specified, they will show up in the Ambari Admin interface and the Ambari Admin can set users and groups on these permissions.

View permissions can also be inherited from Cluster roles. If you are using a Local Cluster for view configuration, you can optionally choose to provide view Use permission based on cluster roles.

**More Information**

Understanding View Development, Persona, Versions, and Deployment

# 2.8. Configuring Views for Kerberos

If the cluster that your views communicate with is Kerberos-enabled, you must:

- Configure all Ambari Server instances for Kerberos.
- Configure each view for Kerberos.
- 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
    ```

- If a view requires HDFS or WebHCat to be configured for a proxy user, you must use the primary Kerberos principal as that user, instead of the ambari-server daemon user.

  For example, if you configure Ambari Server for Kerberos principal ambari-server@EXAMPLE.COM, this value would be ambari-server.

Follow specific instructions to configure each view for Kerberos, and the cluster for Kerberos access from the view.

**More Information**

Configuring Ambari and Hadoop for Kerberos

Configuring Specific Views

# 2.9. Migrating Hue Artifacts to Ambari Views

The Ambari Views Framework provides Ambari users a rich, GUI experience to utilize HDP components. Alternative to Ambari Views, HDP users have leveraged the open source web interface Hue to utilize HDP components.

Ambari 2.5 includes the Hue-to-Views Migration tool, which is specifically designed to migrate existing Apache Hue artifacts to Ambari views.

The following sections describe requirements and how to migrate your existing Hue artifacts to Ambari views:

- Requirements for Hue-to-Views Migration [14]
- Creating a HueToAmbari View Instance [15]
- Migrate Hue Artifacts to an Ambari View [19]

## 2.9.1. 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
- Install a standalone Ambari Server v2.5. Ambari Server must be enabled as a views server
- Hue database types supported:
  - mysql
  - oracle
  - postgresql

  Access rights must be granted to the Hue back-end database.
- In Ambari, create all users in Ambari equivalent to Hue users for each view.

  Grant permission for each view user to use their respective views. Each view user must log in to that view, before migrating artifacts.

**Supported Artifacts, Versions, and Expectations**

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

**Hive**

- Saved Queries
- Query History
- User-defined function (UDF) and JAR artifacts

**Pig**

- Saved scripts
- Pig Jobs
- User-defined function (UDF) and JAR artifacts

**Versions**

Ambari 2.5.x supports migrating Hue version 2.6.1 artifacts into Ambari 2.5.x

**Limitations**

- The Hue-to-Views Migration tool does not validate scripts.

  Scripts maintain the same status in the Ambari Views Framework as they had in Hue.

- The Hue-to-Views migration tool does not support high availability (HA).

  You must provide the current, active namenode for the target cluster regarding the Webhdfs URI for Ambari.

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

- Hue-to-Views migrations must be done on a single cluster. Both Hue and Views must point to the same cluster.

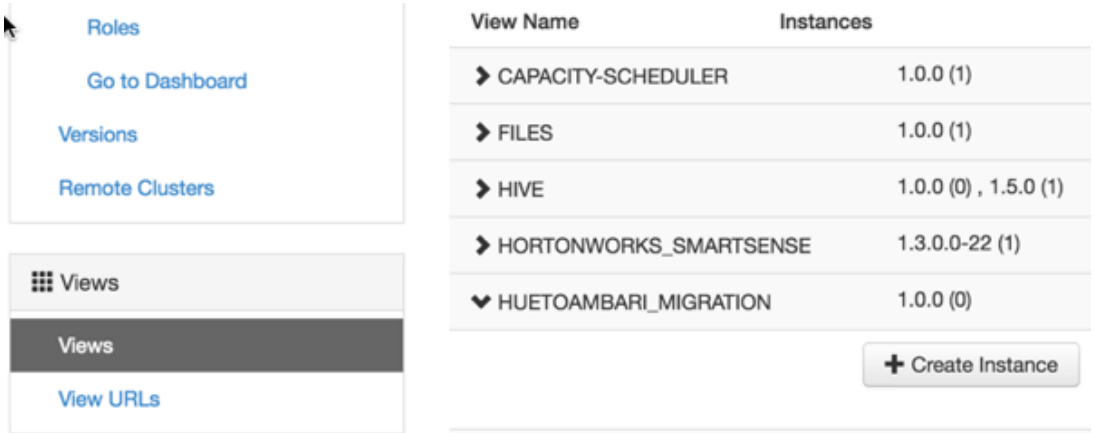
## 2.9.2. Creating a HueToAmbari View Instance

The Hue-to-Views migration tool instantiates an Ambari view named HUETOAMBARI_MIGRATION, but does not create a view instance automatically, when you install Ambari 2.5.0.

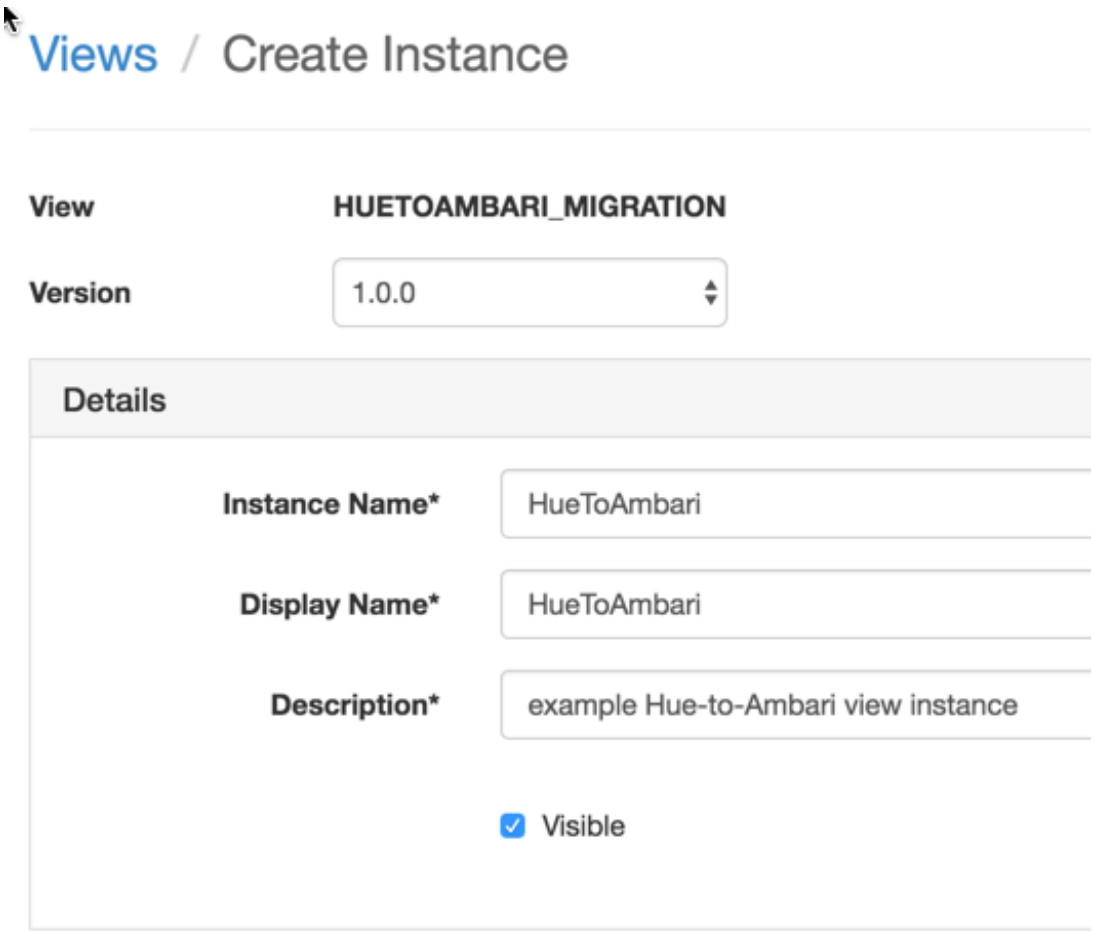
To create a HueToAmbari view instance that appears in the Ambari Views menu:

**Steps**

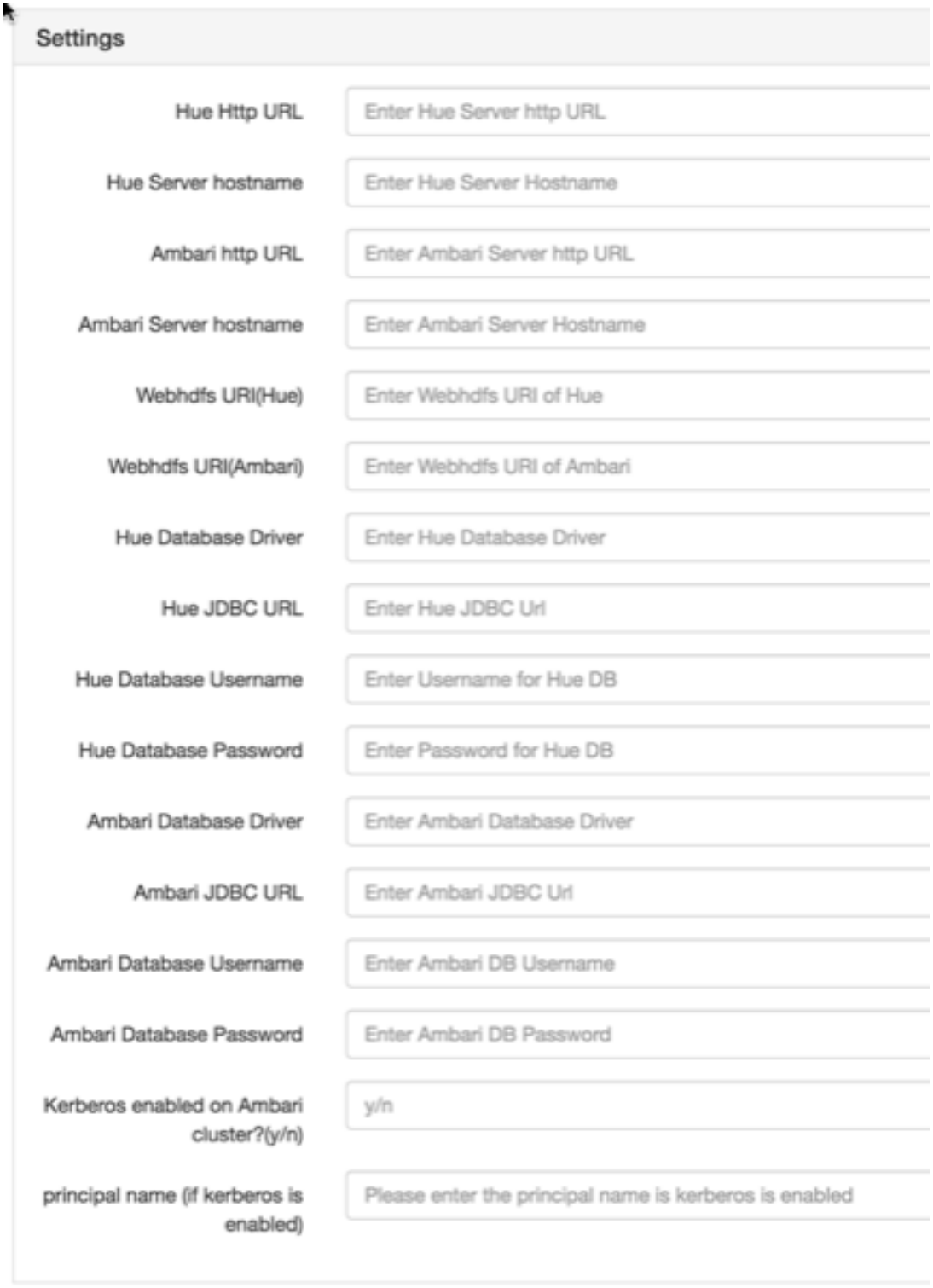
1. In **Ambari Web**, click **admin > Manage Ambari**.
2. On the **Ambari Admin** page, click **Views**.
3. In **View Name**, browse to **HUETOAMBARI_MIGRATION** then expand it.
4. Click **Create Instance**.




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.




The following table shows example values for all required Hue-to-Views migration settings, using a standard, Ambari-managed, 3-node cluster.

- 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 |
| 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 |

### More Information

https://docs.oracle.com/javase/7/docs/api/java/sql/DriverManager.html

7. Click **Save**.

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

**More Information**

Creating View Instances [10]

Migrate Hue Artifacts to an Ambari View [19]

## 2.9.3. Migrate Hue Artifacts to an Ambari View

An administrator-level user can use the HueToAmbari view instance to migrate existing Hue 2.6.1 artifacts into Ambari views.

**Prerequisites**

1. Review the requirements for Hue-to Views migration and complete all prerequisites.
2. Create a view instance, using the Hue-To-Views migration tool.

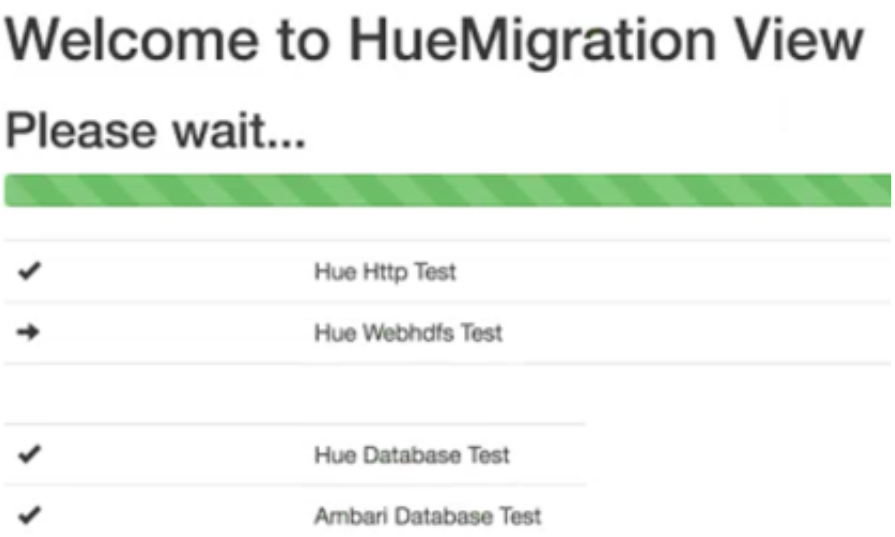
   In the following example, this instance is called HueMigration view.

**Steps**

1. On your standalone views server, using the **Ambari Admin** page, create an administrator-level user.

   Grant permission for the admin user to use the HueMigration view.
2. Log in to the HueMigration view as the administrator user.

   Logging in initializes the database for the new view.
3. Wait while the view performs database checks.




4. Transfer artifacts.

   To transfer artifacts:

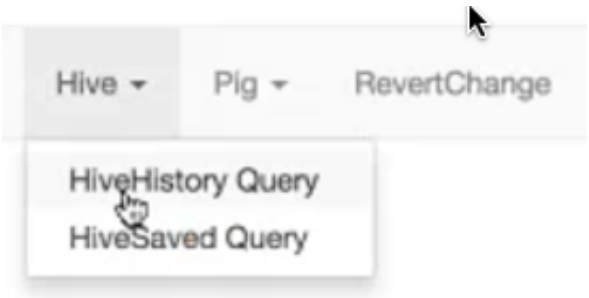
   **Steps**

   a. In Ambari Web, click **Views > HueMigration**.

      The HueMigration view shows Hive and Pig menus that list existing Hue artifacts.

      The RevertChange option allows you to clear any migration settings before you submit them.

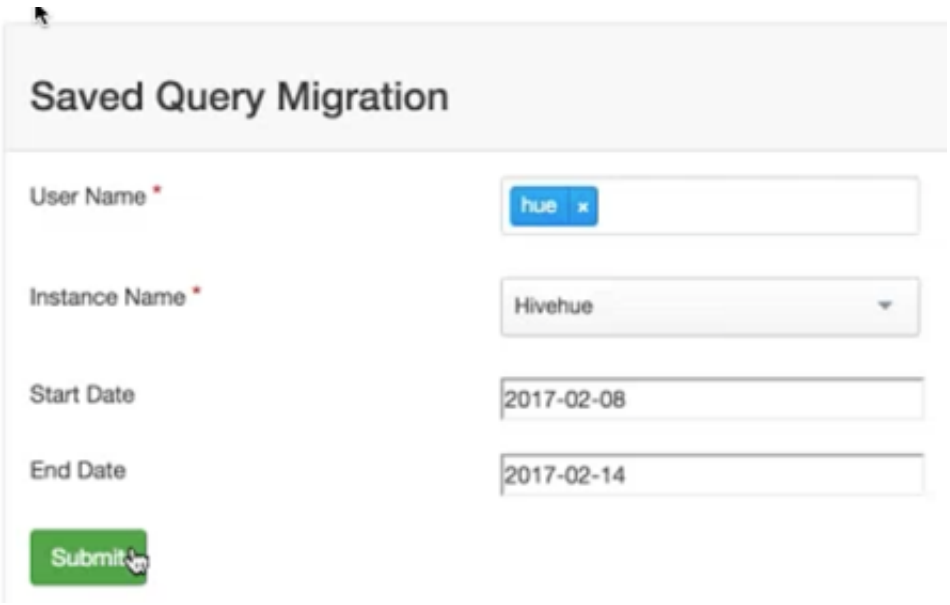
   b. From each artifact menu, click an artifact name.




      A migration settings page displays options for migrating the artifacts.

   c. In the migration settings page, specify:

      - a user name and instance name *to which* the artifacts will migrate
      - a date range (using Start and End dates) *from which* to migrate artifacts




d. Click **Submit**.

When the migration completes, a migration report displays.


e. Review the migration report to confirm artifacts transferred.

5. Verify that the artifacts for each user have transferred from Hue to each of the Hive and Pig views.

For each view and user.

**Steps**

a. Log in to your Views server, as the view user.

b. Open a view to which you migrated artifacts.

c. Verify that queries, scripts and UDFs migrated successfully.

**More Information**

Requirements for Hue-to-Views Migration [14]

Running Ambari Server Standalone [7]

Creating a HueToAmbari View Instance [15]

Setting View Permissions [12]

# 2.10. Configuring Specific Views

Ambari configures and deploys most views automatically, for each service added to a cluster.

Depending on your environment, each View (and associated service) may require some additional configuration or troubleshooting.

**More Information**

Configuring Capacity Scheduler View [22]

Configuring Files View [30]

Configuring Falcon View [34]

Configuring Hive View [40]

Configuring Pig View [50]

Configuring Slider View [58]

Configuring SmartSense View [59]

Configuring Storm View [61]

Configuring Tez View [63]

Configuring Workflow Manager View [68]

## 2.10.1. Configuring Capacity Scheduler View

**Prerequisites**

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

**More Information**

Creating a Capacity Scheduler View Instance [23]

Troubleshooting Capacity Scheduler View [30]

### 2.10.1.1. Creating a Capacity Scheduler View Instance

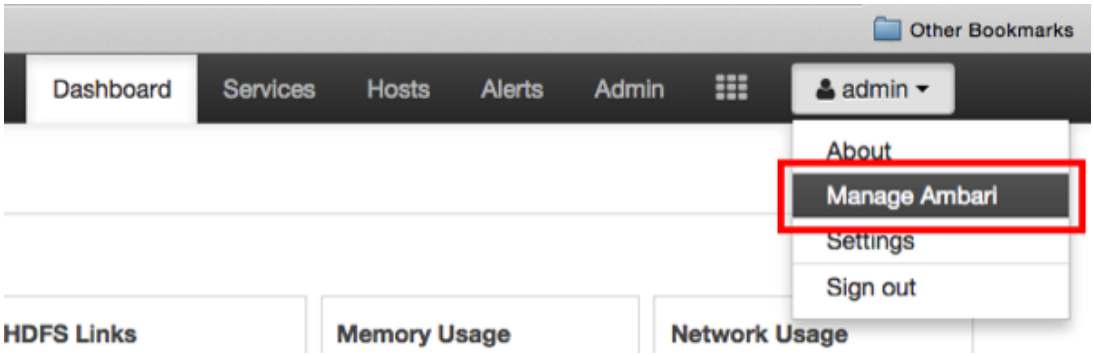
When you deploy a cluster using Ambari, a Capacity Scheduler View instance is automatically created. If you do not need to reconfigure the Ambari-created cluster, proceed to use the YARN Queue Manager View.

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.

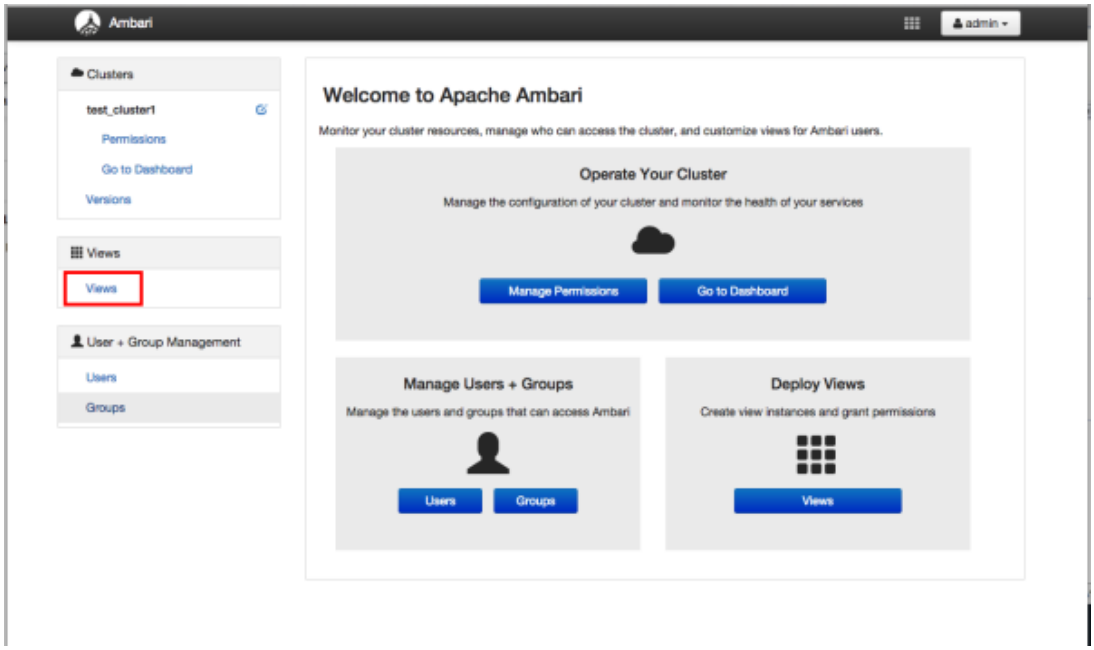
To set up a Capacity Scheduler / YARN Queue Manager view instance:

**Steps**

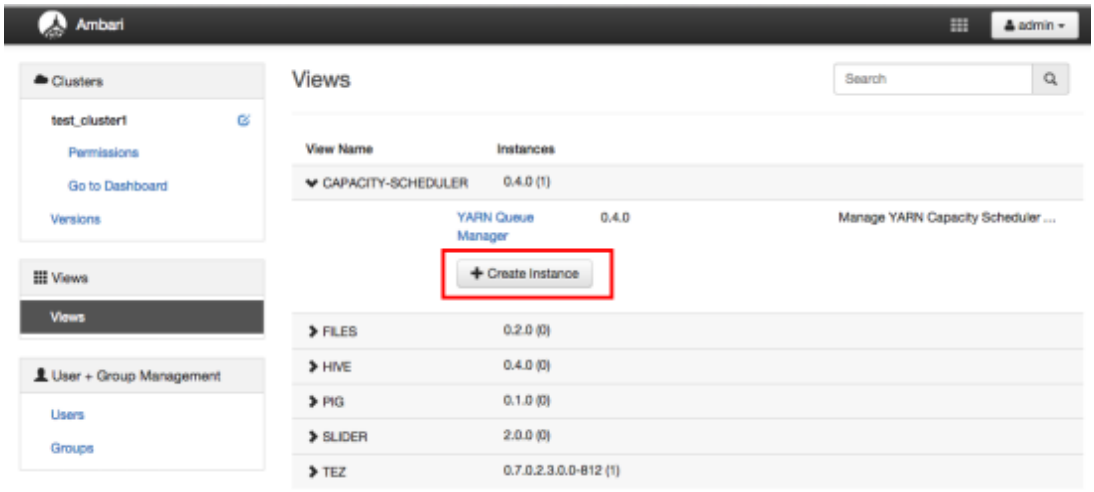
1. In the **Ambari Web** top menu, click **admin > Manage Ambari**.




2. On the **Ambari Admin** 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 an instance name, display name, and a description for the view.

**Note**

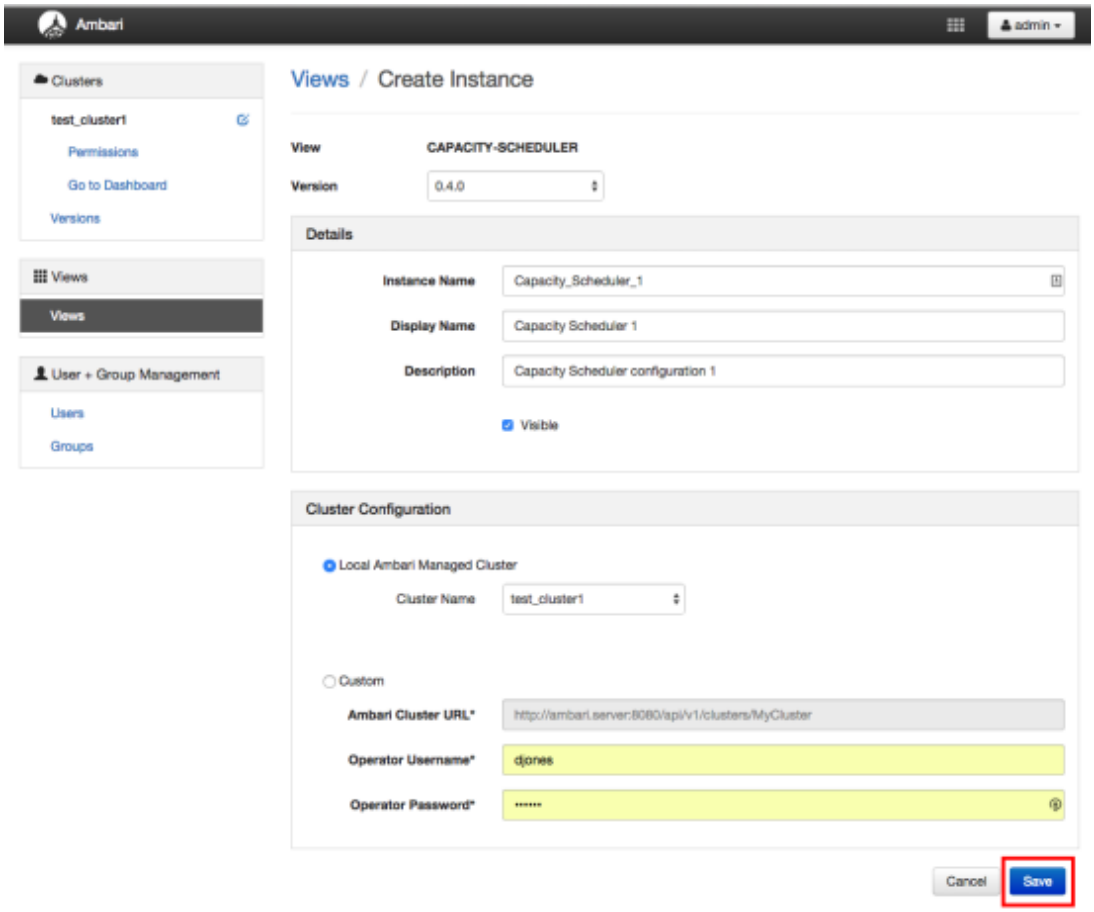
The instance name cannot contain spaces or special characters.

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

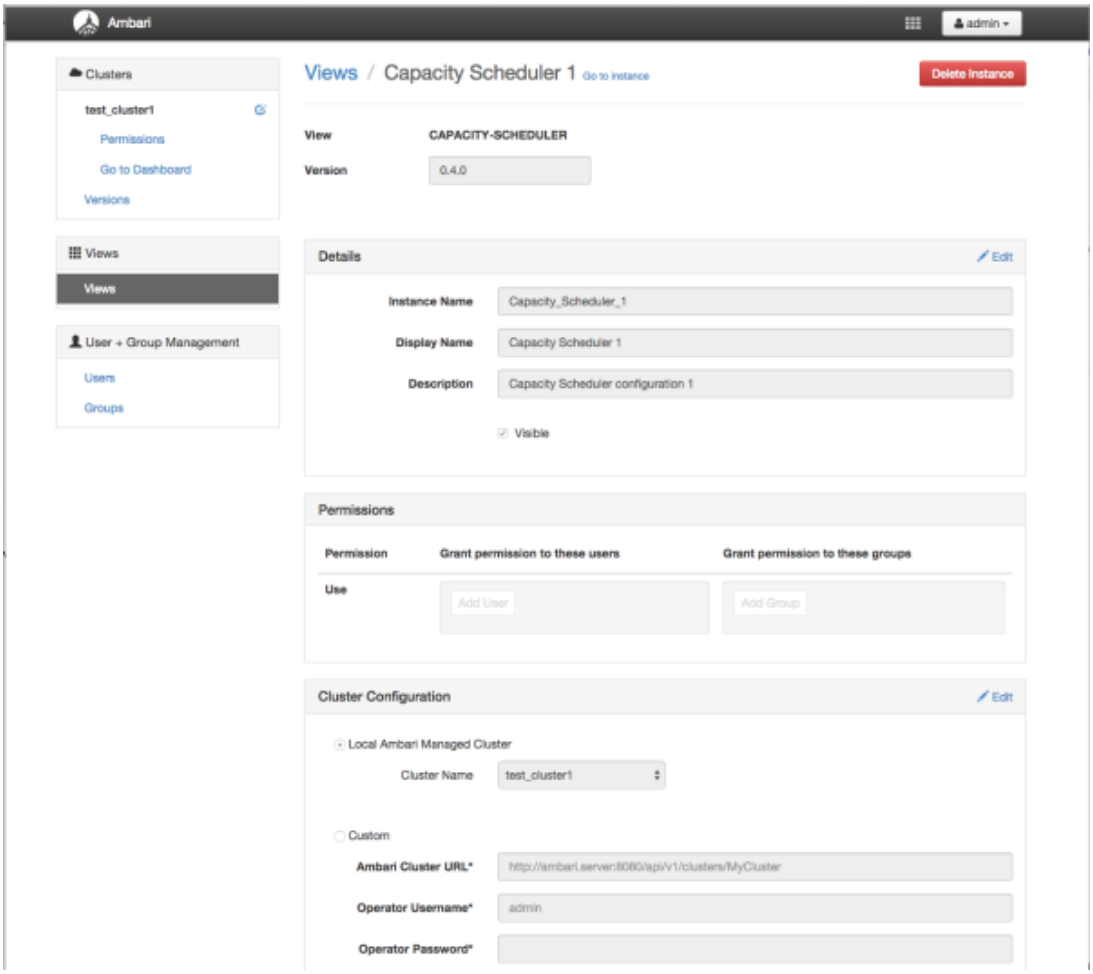
   - For HDP clusters that are local (managed by the local Ambari Server), click the **Local Ambari Managed Cluster** option, then click the local cluster name.

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

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




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




**More Information**

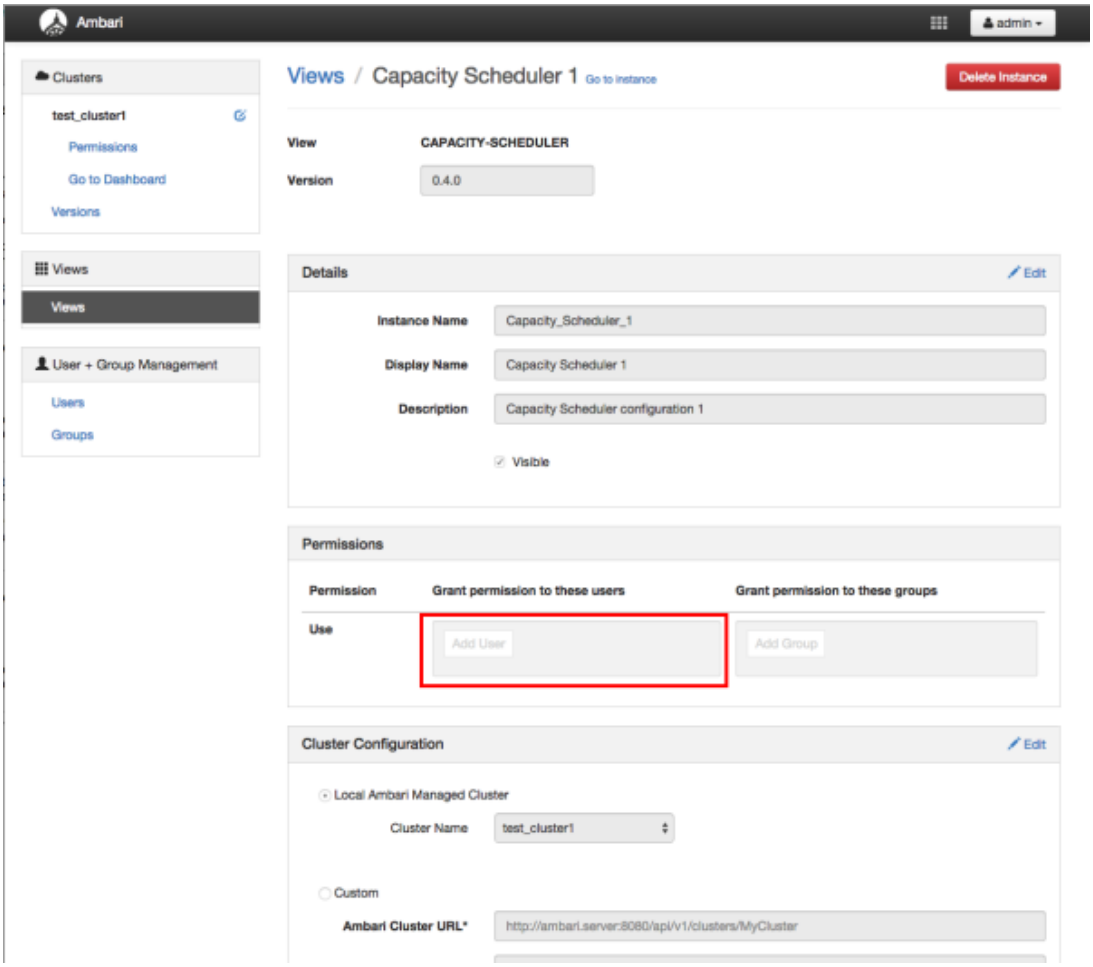
Using YARN Queue Manager View [75]

## 2.10.1.2. User Permissions for YARN Queue Manager View

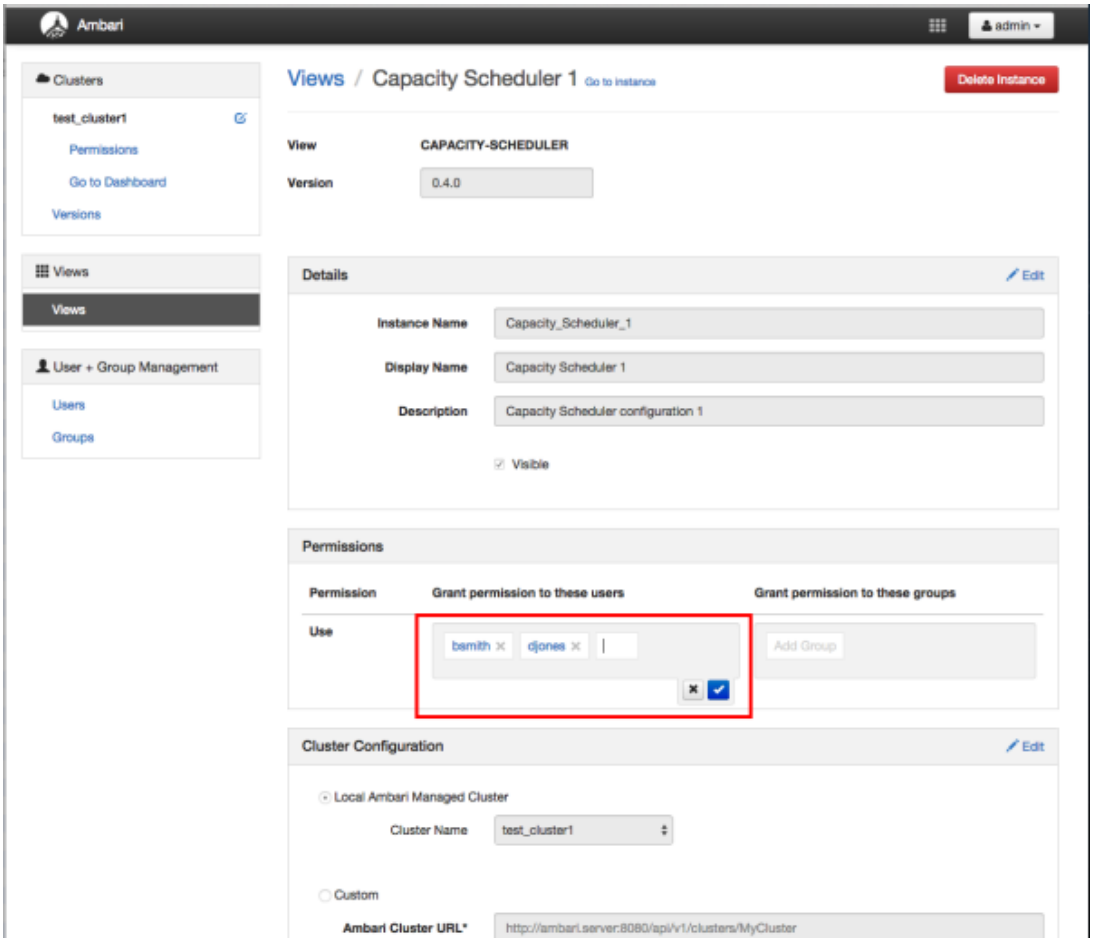
To add users and groups to a YARN Queue Manager view instance:

**Steps**

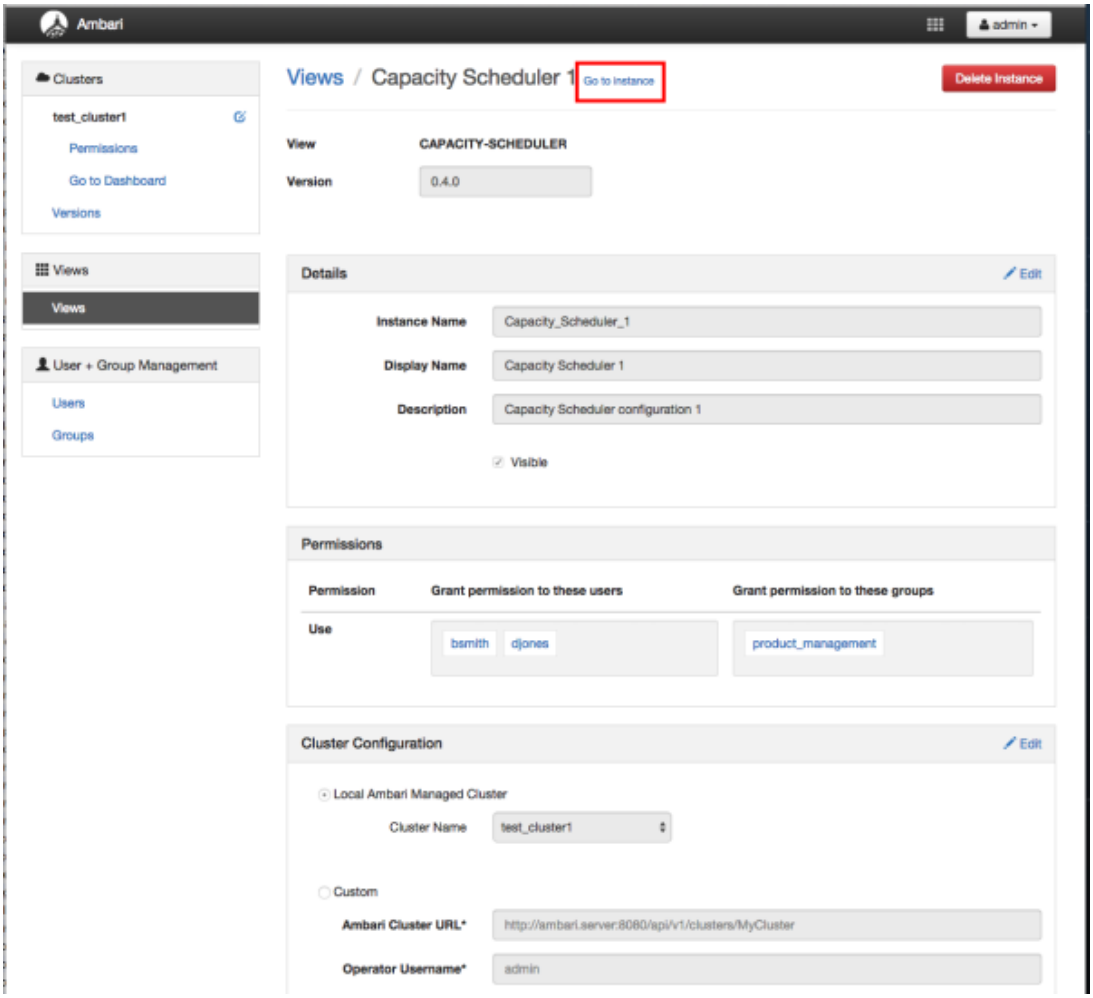
1. On the Capacity Scheduler view instance configuration page, click **Add User** in **Permissions**.



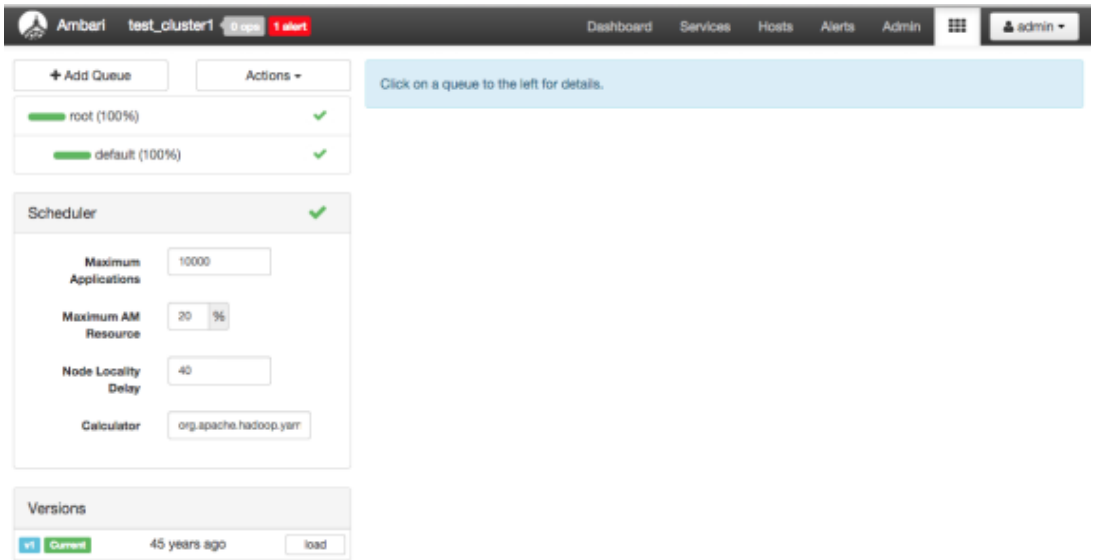

2. In **Use**, enter user names, then click the blue check mark to add the users. You can use the same method to add groups in **Add Group**.




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.



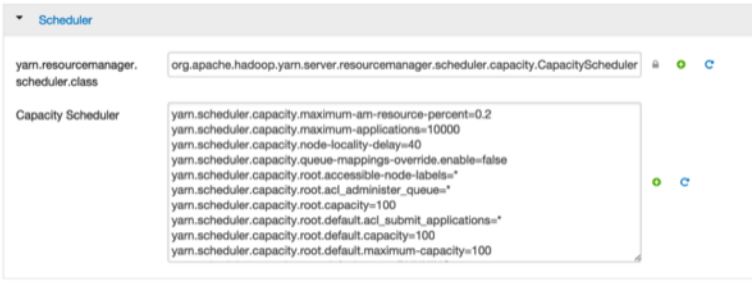

4. The Capacity Scheduler view instance page appears.



### 2.10.1.3. Troubleshooting Capacity Scheduler View

If you encounter an issue where the configurations cannot be applied from Capacity Scheduler 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, using **Ambari Web**, browse to **>Services > YARN**, then click the **Configs** tab. On the **Advanced** tab, expand **Scheduler**.




In **Scheduler** you can edit the underlying configurations for the YARN Queue Manager and fix any issues you may encounter.

## 2.10.2. Configuring Files View

This section describes how to configure a Files View instance and use the File browser UI to access HDFS.

- Configuring Your Cluster for Files View [30]
- Creating and Configuring a Files View Instance [31]
- Troubleshooting [34]

#### 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 and for more information.

**More Information**

Preparing Ambari Server for Views [5]

Running Ambari Server Standalone [7]

### 2.10.2.1. Configuring Your Cluster for Files View

For 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 Files View users.

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.

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:

**Steps**

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

| | |
|---|---|
| hadoop.proxyuser.root.groups="users"<br>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 the account name `ambariusr`, *you would use the following properties instead:* |
| hadoop.proxyuser.ambariusr.groups="users"<br>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"<br>hadoop.proxyuser.ambari-server.hosts=ambari-server.hostname | |

5. Save the configuration change and restart the required components as prompted by Ambari.

**More Information**

Configure Ambari Server for Kerberos

## 2.10.2.2. Creating and Configuring a Files View Instance

To create and configure a Files View instance:

**Steps**

1. Browse to the **Ambari Admin** page.

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

3. In **Details**, enter the following values for View instance properties:

| 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. Information that you provide in **Settings and Cluster Configuration** depends on your environment; specifically, whether:

   - your cluster is Kerberos-enabled or not
   - NameNode HA is enabled or not
   - your Files View instance is being configured in an operational or a standalone Ambari server

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

| Kerberos Enabled | NameNode HA Enabled | Operational Ambari Server | Standalone Ambari Server |
|---|---|---|---|
| No | No | Settings: defaults<br>Cluster Configuration: Local | Settings: defaults<br>Cluster Configuration: Custom |
| No | Yes | | |
| Yes | No | Settings : Kerberos<br>Cluster Configuration : Custom | |
| Yes | Yes | Settings: Kerberos<br>Cluster Configuration: Custom | |

The Local Ambari Managed Cluster Configuration option is enabled in the Ambari Admin page *only* if you are managing a cluster in an Operational Ambari Server.

**More Information**

Cluster Configuration: Local [33]

Cluster Configuration: Custom [33]

Kerberos Settings [32]

Running Ambari Server Standalone [7]

### 2.10.2.2.1. Kerberos Settings

**Prerequisites**

Before setting up Kerberos for Files View, you must first set up Kerberos for Ambari by configuring the Ambari Server daemon with a Kerberos principal and keytab.

**Steps**

After you have set up Kerberos for Ambari, in **Files View > Settings**, enter the following properties:

| 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<br><br>**This property is only needed if the view is Custom Configured or Ambari Server is Kerberized before 2.4.0. |

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**.

**More Information**

Configuring Views for Kerberos

#### 2.10.2.2.2. Cluster Configuration: Local

The **Local Ambari Managed Cluster Configuration** option is enabled on the **Ambari Admin** page 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 communicates with HDFS based on the fs.defaultFS property (for example: hdfs://namenode:8020). The View also determines whether NameNode HA is configured and adjusts accordingly.

#### 2.10.2.2.3. 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<br><br>For example:<br><br>nameservice |
| List of NameNodes | Comma-separated list of NameNodes for a given nameservice. | hdfs-site/dfs.ha.namenodes<br><br>For example:<br><br>namenode1,namenode2 |
| First NameNode RPC Address | RPC address for first name node. | hdfs-site/dfs.namenode.rpc-address.[nameservice].[namenode1] |

| Property | Description | Example Value |
|---|---|---|
| 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] |

#### 2.10.2.2.4. 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.<br><br>For example:<br><br>auth=KERBEROS;proxyuser=ambari-server |
| **500** User: ambari-server is not allowed to impersonate admin | HDFS has not been configured for Ambari as a proxy user. |
| **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.<br><br>For example:<br><br>webhdfs://namenode:50070 |

**More Information**

Kerberos Settings [32]

Configuring Your Cluster

Cluster Configuration: Custom [33]

## 2.10.3. Configuring Falcon View

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

This section describes:

- Configuring Your Cluster for Falcon View [34]
- Installing and Configuring Falcon View [36]

### 2.10.3.1. Configuring Your Cluster for Falcon View

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

#### 2.10.3.1.1. Set up 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**.

**Steps**

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.

**More Information**

Ambari Server for Kerberos

#### 2.10.3.1.2. Set up HDFS User Directory

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 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 Falcon View user:

**Steps**

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
```

### 2.10.3.2. Installing and Configuring Falcon View

You must manually copy the `.jar` file for Falcon View, then configure Ambari to access the view. You can install 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.

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

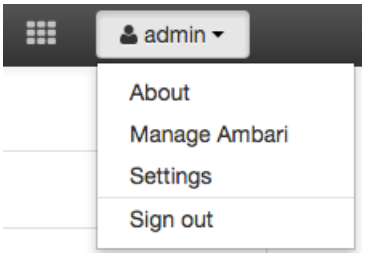
   - If the Falcon server is on a remote host, use the secure copy command for your operating system.

     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 Web, browse 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**.

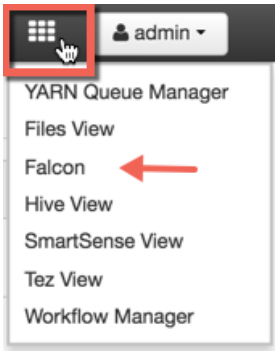
   The Permissions section displays at the bottom of the Views page.

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

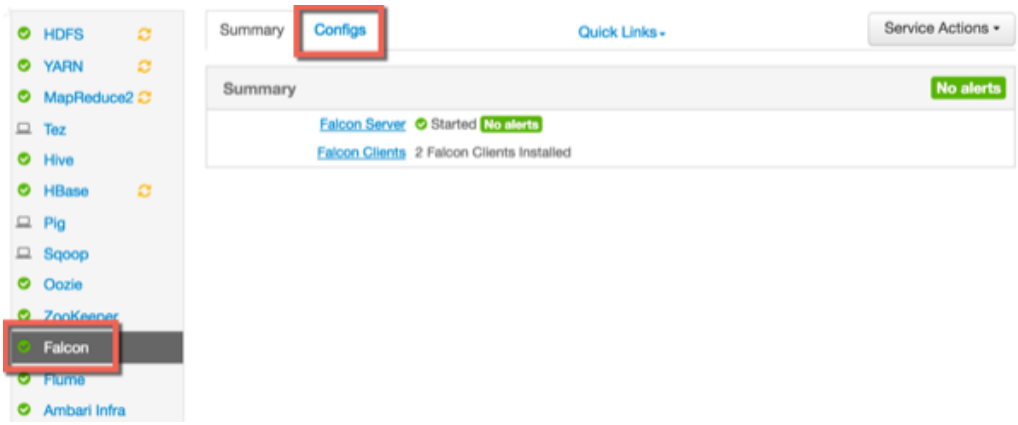
9. Hover over the **Views** icon to verify that your Falcon View is available in the menu.

   **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=*

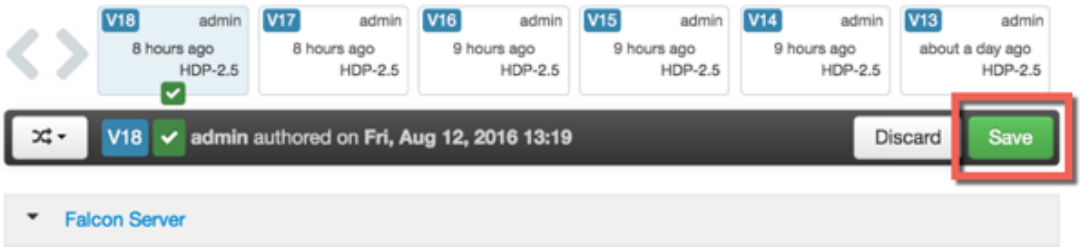
These are the groups that the users being impersonated must belong to.

### Example 2.1. 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.

**More Information**

Adding a Service to your Hadoop cluster

Installing Apache Falcon

Managing Users and Groups

## 2.10.4. Configuring Hive View

This section describes:

- Upgrading Your Hive View [40]
- Configuring Your Cluster for Hive View [41]
- Creating a Hive View Instance [43]
- Troubleshooting Hive View [49]

**Prerequisites**

- Prepare your Ambari Server for hosting views.

  It is strongly recommended that you:

  - increase the amount of memory available to your Ambari Server
  - run a standalone Ambari Server to host the views
- Install Tez View when you install Hive View. Tez View integrates with Hive View.

**Hive View Versions**

With the release of Apache Ambari 2.5.0, two Hive View versions install as part of your Hortonworks Data Platform distribution:

**Hive View 1.5**

**Hive View 2.0**

Both versions are JDBC-based. You can run both views simultaneously, use only one of the views, or upgrade your data from the older view to the newer view.

### Important

In the Ambari UI and in this documentation going forward, *Hive View* refers to version 1.5 of the view and *Hive View 2.0* is the term to differentiate the new version of the view from the previous version.

**More Information**

Preparing Ambari Server for Views [5]

Running Ambari Server Standalone [7]

Using Tez View [115]

### 2.10.4.1. Upgrading Your Hive View

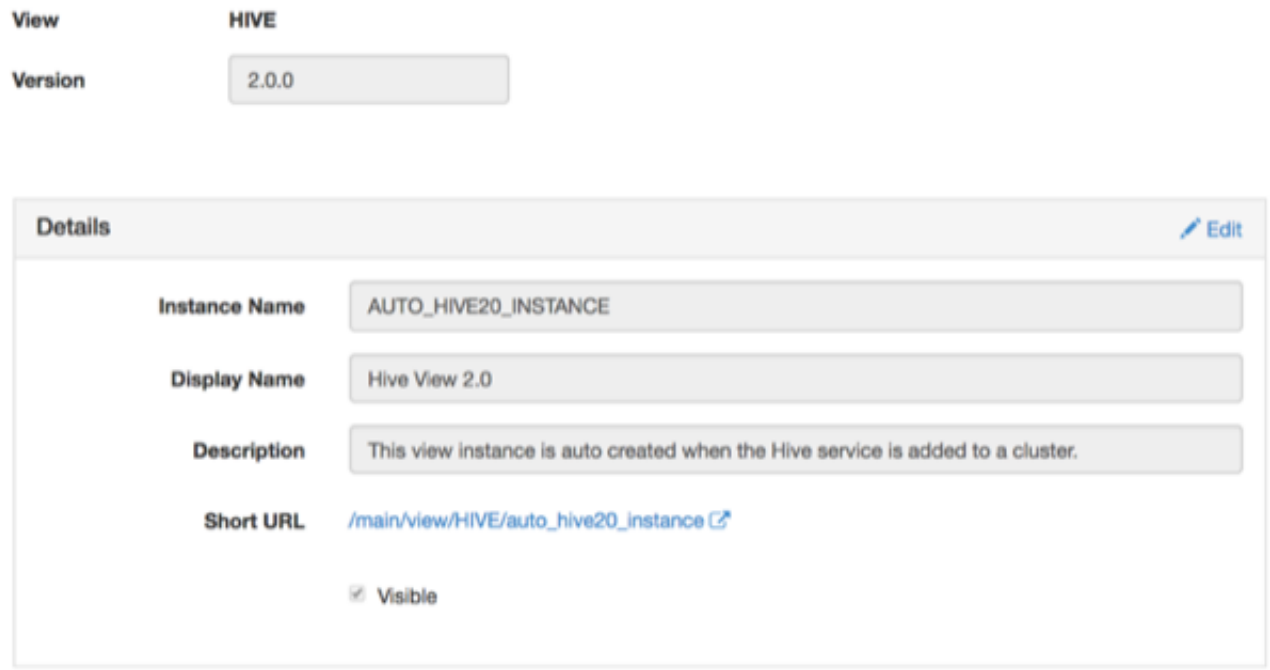
If you are upgrading from Apache Ambari 2.4.0 to Apache Ambari 2.5.0 and want to upgrade and migrate the data and queries from Hive View to Hive View 2.0, first create a new instance of the Hive View and then migrate your queries.

**Migrating your queries into the new view**

Create a new Hive View 2.0 instance and then migrate the saved queries from the Hive View 1.5 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:




## 2.10.4.2. Configuring Your Cluster for Hive View

For 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 Hive View users. This is critical since the Hive View will store metadata about their user Hive queries in HDFS. This also means users that access Hive View must have a user directory setup in HDFS.

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 HDFS proxy user and user directory settings are correct. If you are running views on a standalone server, you must set up proxy user settings manually.

### Important

*For clusters with wire encryption enabled*: You must configure a Hive session parameter to use Hive View:

1. Click the drop-down menu of your username profile (top right corner of window).

2. Click **Manage Ambari** > **Views** > **Hive** > **Hive View** or **Hive View 2.0**.
3. Click the **Edit** icon in the Settings part of the window.
4. In the the **Hive Session Parameters** field, enter **`sslTrustStore=/etc/security/serverKeys/hivetruststore.jks;trustStorePassword=`*`your_password`***, inserting a real password in place of *your_password*. If you do not specify a password, the default password that gets assigned is **`changeit`**.
5. Click **Save**.

**More Information**

Set up HDFS Proxy User [42]

Set up HDFS User Directory [43]

#### 2.10.4.2.1. Set up HDFS Proxy User

To set up an HDFS proxy user for the Ambari Server daemon account, you must 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:

**Steps:**

1. In Ambari Web, browse to **Services > HDFS > Configs**.
2. Under the **Advanced** tab, browse 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.

**More Information**

Ambari Server for Kerberos

#### 2.10.4.2.2. Set up 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, you must create the `/user/admin` folder in HDFS using these instructions before using Hive view.

To create user directories in HDFS, for each Hive View user:

**Steps**

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
```

### 2.10.4.3. Creating a Hive View Instance

**Steps**

1. Click **Manage Ambari** to open the Ambari Admin user interface.
2. Click **Views** > **Hive** > **Create Instance**.
3. On the Create Instance page, select the **Version**. If multiple Hive View JAR files are present, choose one.
4. Enter the following view instance details:

**Table 2.1. Hive View Instance Details**

| Property | Description | Example Value |
|---|---|---|
| Instance Name | This is the Hive view instance name. This value should be unique for all | AUTO_HIVE_INSTANCE |

| Property | Description | Example Value |
|---|---|---|
| | Hive view instances you create. This value cannot contain spaces and is required. | |
| 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. |
| Short URL | Alternative to full URL to quickly navigate to the Hive View. | 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.

6. Click **Save**.

**More Information**

Settings and Cluster Configuration [44]

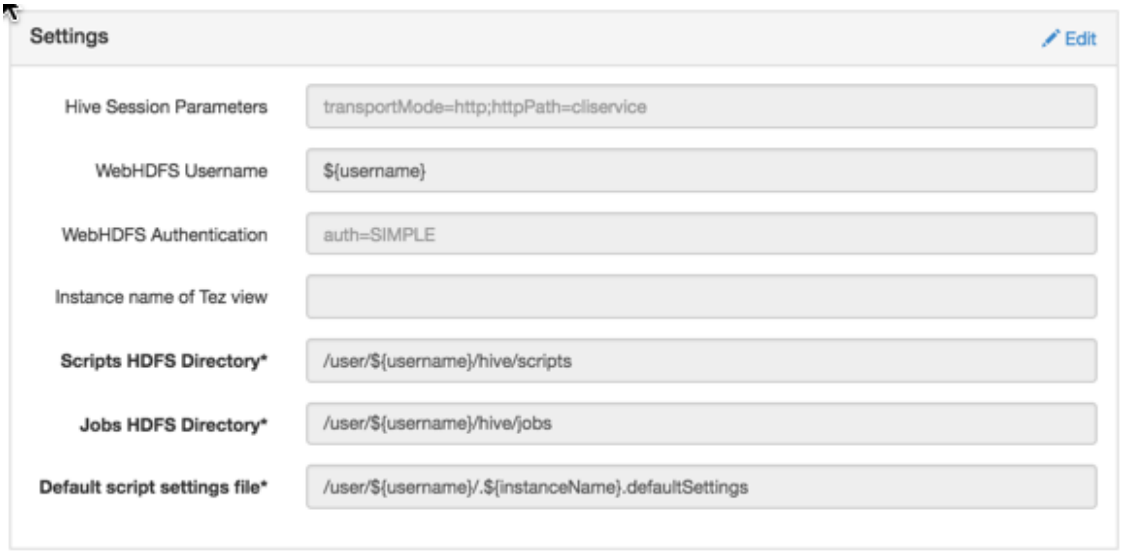
#### 2.10.4.3.1. Settings and Cluster Configuration

Ambari configures Hive View settings automatically when you choose to add the Hive service.

**Tip**

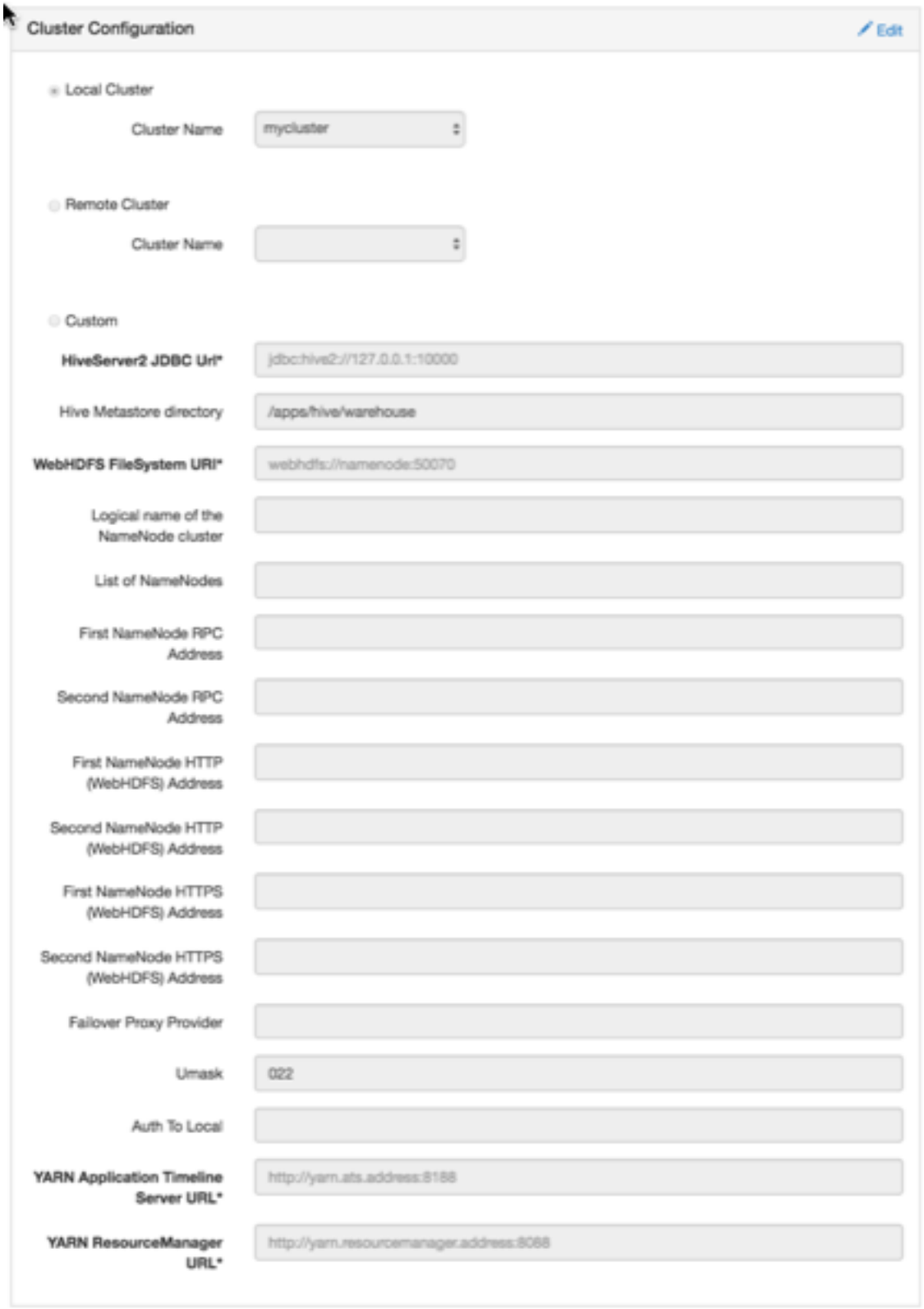
To use Hive View with a Hive LLAP (interactive query) environment, set the **Use Interactive Mode** property setting to `true`.

**Figure 2.2. Default Hive View Settings**



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

**Figure 2.3. Default Hive View Cluster Configuration**



If required for migrating view instances, find and modify the following cluster configuration settings, using Ambari Web.

| | |
|---|---|
| HiveServer2 JDBC URL | Click **Hive > Summary** to view the URL, displayed at the bottom of the Summary list.<br><br>For example:<br><br>jdbc:hive2://<br>c6403.ambari.apache.org:2181,c6401.ambari.apache.org:2181,c6402.am |
| 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<br><br>**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://<br>c6401.ambari.apache.org:50070 or webhdfs://<br>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**.<br><br>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**.<br><br>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**<br><br>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**<br><br>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**<br><br>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**<br><br>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**<br><br>The default value is 022. Do not change this value unless you are sure that you understand the effects of changing the value on your Hive View cluster. The umask property defines the file mode creation mask, which controls how file permissions are configured in new files. |
| 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 "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:

| | |
|---|---|
| First NameNode RPC Address, or Second NameNode RPC Address | Select the primary or secondary NameNode to view settings from that host in the cluster. 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 | 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 |

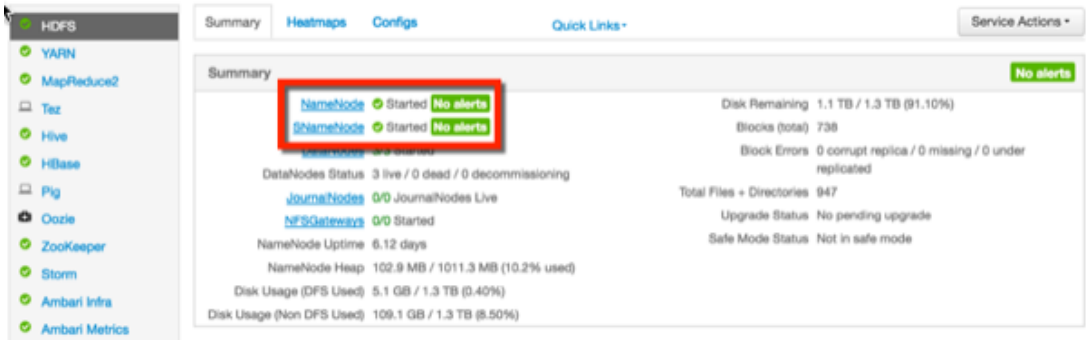
| | |
|---|---|
| NameNode HTTP (WebHDFS) Address | value you find in the **advanced hdfs-site** settings. For example, http://c6401.ambari.apache.org:50070 |

**To get First NameNode RPC Address values:**

**Steps**

1. In **Ambari Web**, browse to **HDFS > Summary** page. Click **NameNode** (primary) or **SNameNode** (secondary) to view the host page:

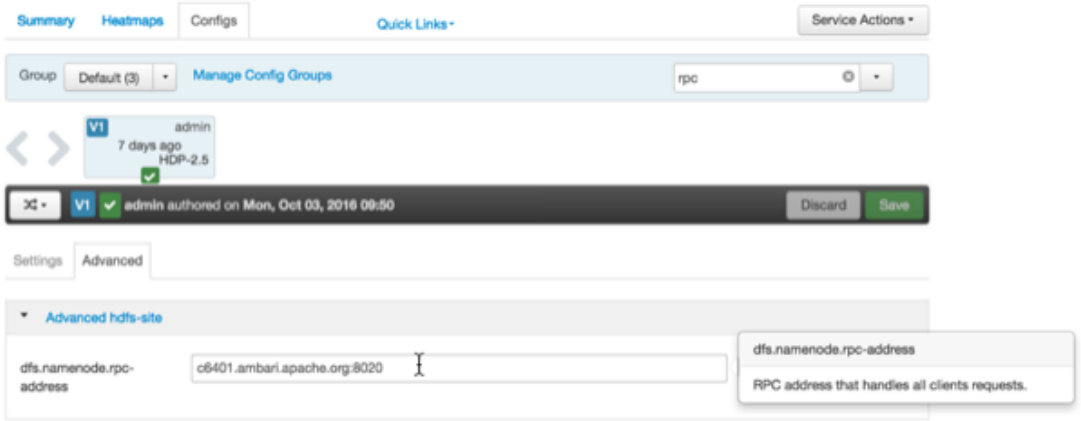
**Figure 2.4. HDFS Service Page in Ambari**




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 browse 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 2.5. Using the Filter to Search Advanced hdfs-site Settings**



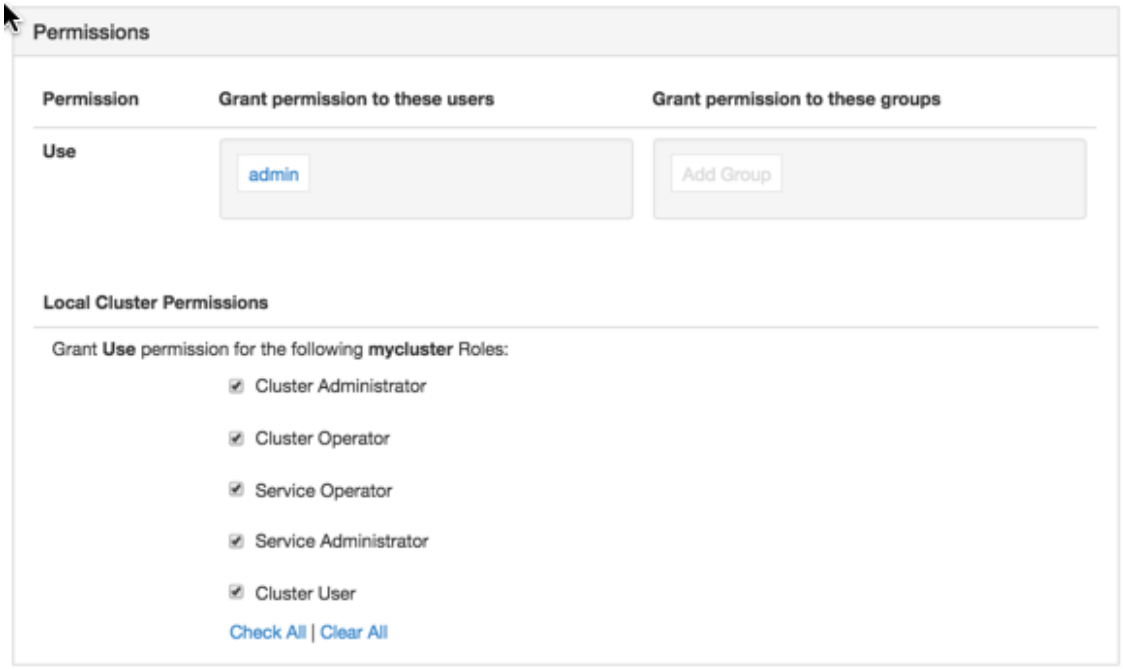

**More Information**

Kerberos Setup for Hive View [49]

### 2.10.4.3.2. 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 2.6. Granting User Permissions to Hive Views




#### 2.10.4.3.3. Kerberos Setup for Hive View

**Prerequisites**

Set up basic Kerberos for the Ambari server that manages Views.

**Steps**

After you have set up Kerberos on the Ambari Views server, you must manually set the following Hive View property:

**WebHDFS Authentication** `auth=KERBEROS;proxyuser=<ambari-principal-name>`

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

**More Information**

Set Up Kerberos for Ambari Server

## 2.10.4.4. Troubleshooting Hive View

### Table 2.2. 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. |
| 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. |

**More Information**

Set up HDFS Proxy User [42]

Set up HDFS User Directory [43]

## 2.10.5. Configuring Pig View

This section describes:

- Configuring Your Cluster for Pig View [50]
- Creating a Pig View Instance [53]

### 2.10.5.1. Configuring Your Cluster for Pig View

For 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 Pig View will store metadata about the user Pig scripts. This also means users that will access Pig View must have a user directory setup in HDFS. In addition, Pig View uses WebHCat to submit Pig scripts so the View needs a proxy user for WebHCat.

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 HDFS and WebHCat proxy user and the HDFS user directory settings are correct. If you are running views on a standalone server, you must setup proxy user settings manually.

**More Information**

Set up HDFS Proxy User [50]

Set up WebHCat Proxy User [51]

Set up HDFS User Directory [52]

#### 2.10.5.1.1. Set up 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:

**Steps**

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.

**More Information**

Ambari Server for Kerberos

#### 2.10.5.1.2. Set up 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 :

**Steps**

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:

**Steps**

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.

**More Information**

Ambari Server for Kerberos

### 2.10.5.1.3. Set up 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, for each PigView user :

**Steps**

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
```

## 2.10.5.2. Creating a Pig View Instance

To create a Pig View Instance:

**Steps**

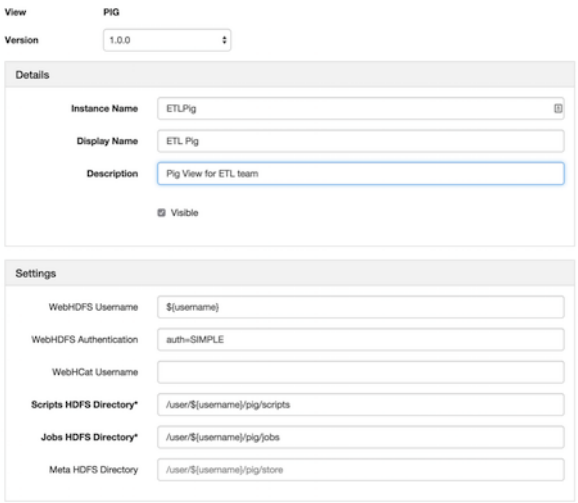
1. Browse to the Ambari Admin interface.

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.

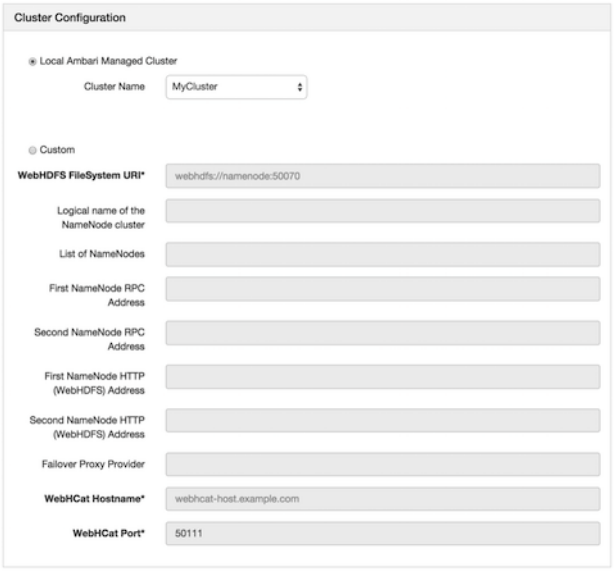
4. Enter the Details and Settings. The Instance Name appears in the URI, the Display Name appears in the Views drop-down list, and the Description helps multiple users identify the view:

**Figure 2.7. 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 2.8. 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.

## 2.10.5.3. 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.

| | |
|---|---|
| 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:

| | |
|---|---|
| First NameNode RPC Address or Second NameNode RPC Address | Select the primary or secondary NameNode to view settings from that host in the cluster. 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:**

**Steps**

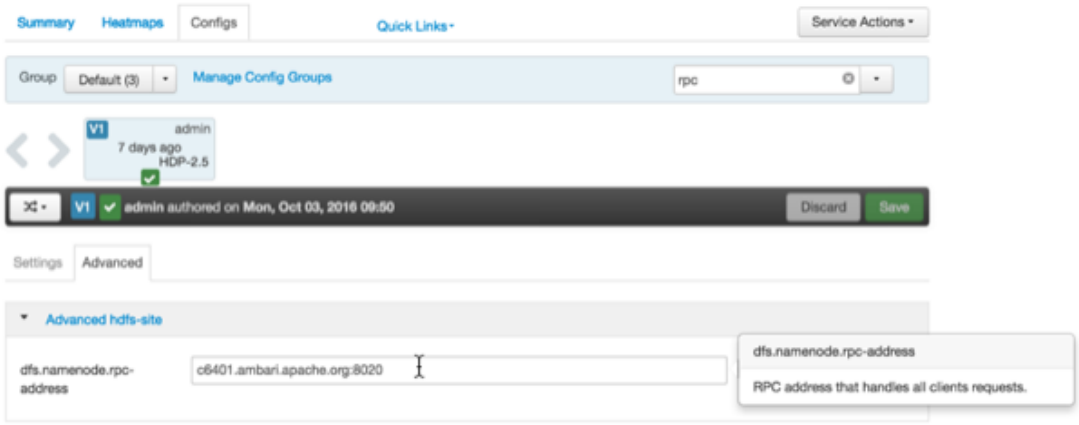
1. In Ambari Web, browse to the **HDFS Summary** page. Click **NameNode** (primary) or **SNameNode** (secondary) to view the host page:

**Figure 2.9. 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:

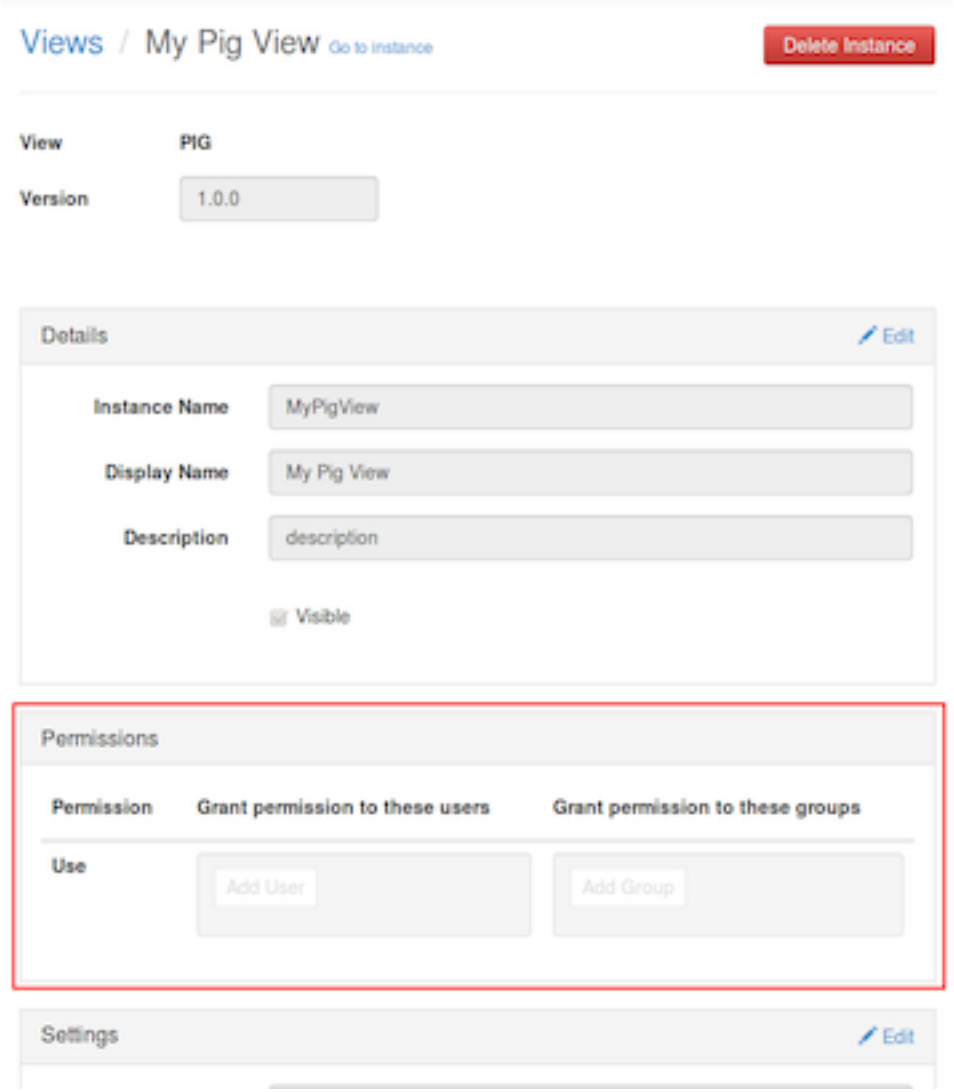
**Figure 2.10. Using the Filter to Search Advanced hdfs-site Settings**




## 2.10.5.4. User Permissions for Pig View

After saving the Pig View instance definition, grant view permissions for all Pig View users:

**Figure 2.11. Granting User Permissions to Pig View**




## 2.10.5.5. Kerberos Setup for Pig View

**Prerequisites**

Set up basic Kerberos for the Ambari views server.

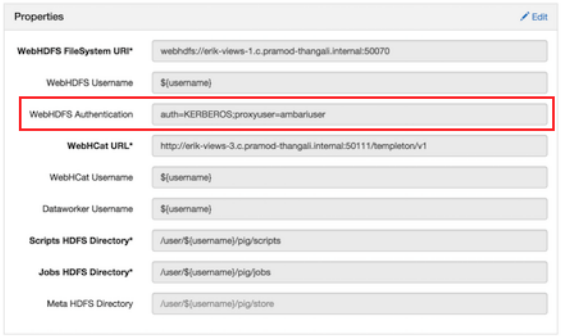
**Steps**

After you have set up Kerberos on the Ambari Views server, you must manually set the following Pig View property:

| | |
|---|---|
| **WebHDFS Authentication** | `auth=KERBEROS;proxyuser=<ambari-principal-name>`<br><br>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 2.12. Kerberos Settings for Pig View**




**More Information**

Set Up Kerberos for Ambari Server

## 2.10.6. Configuring Slider View

To configure Slider View:

**Steps**

1. From the Ambari Admin 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.COM |
| 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.

# 2.10.7. Configuring SmartSense View

This chapter describes:

- Configuring Your Cluster for SmartSense View [59]
- Creating a SmartSense View Instance [59]
- Using SmartSense View [108]

## 2.10.7.1. Configuring Your Cluster for SmartSense View

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.

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 a SmartSense instance manually.

Before accessing 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.

**More Information**

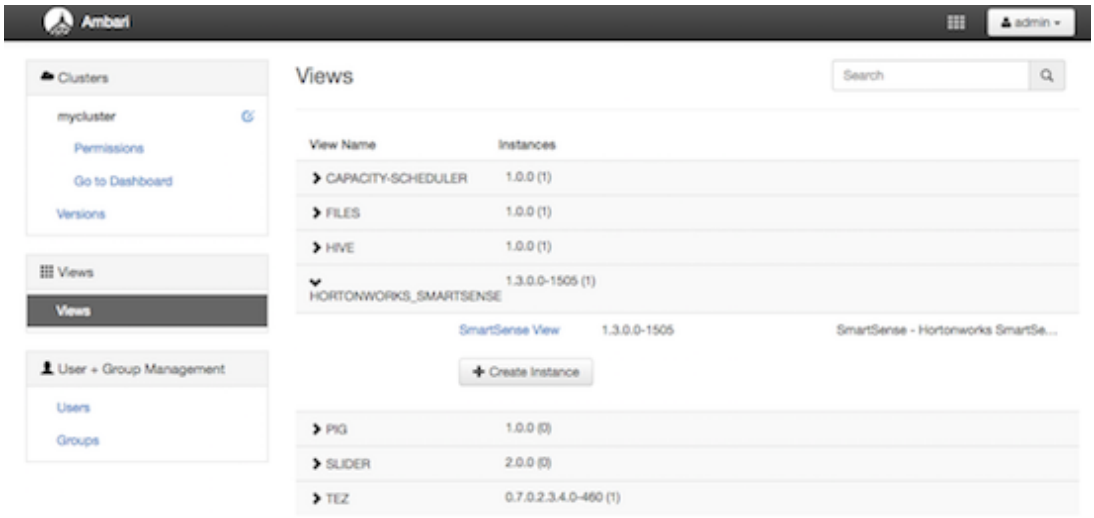
Creating a SmartSense View Instance [59]

## 2.10.7.2. Creating a SmartSense View Instance

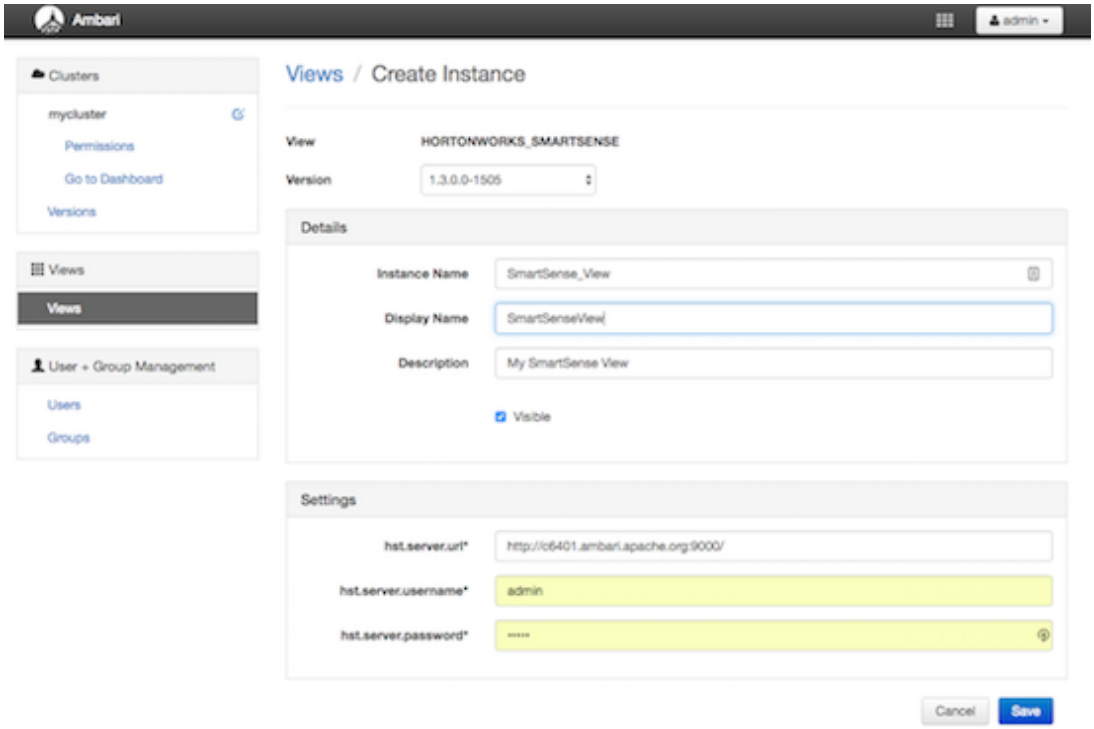
To create a SmartSense view instance manually:

**Steps**

1. Browse to the Ambari Admin 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:

| | |
|---|---|
| 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. |
| 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:

| | |
|---|---|
| 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**.

# 2.10.8. Configuring Storm View

## Important

This view has been marked deprecated.

**Prerequisites**

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

Prepare your Ambari Server for hosting views.

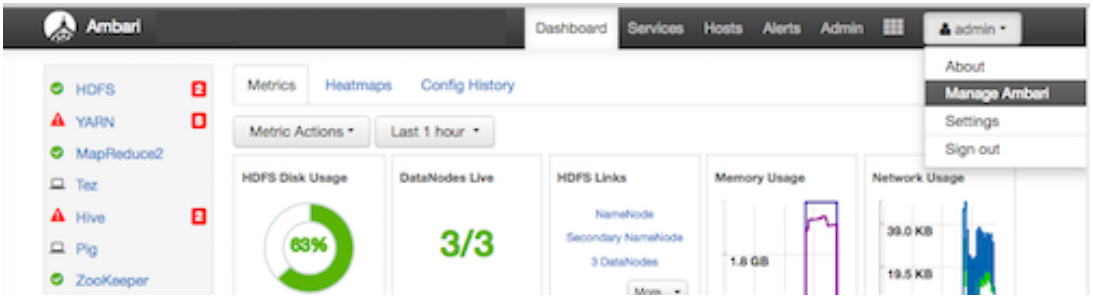
It is strongly recommended that you:

- increase the amount of memory available to your Ambari Server
- run a standalone Ambari Server to host the views

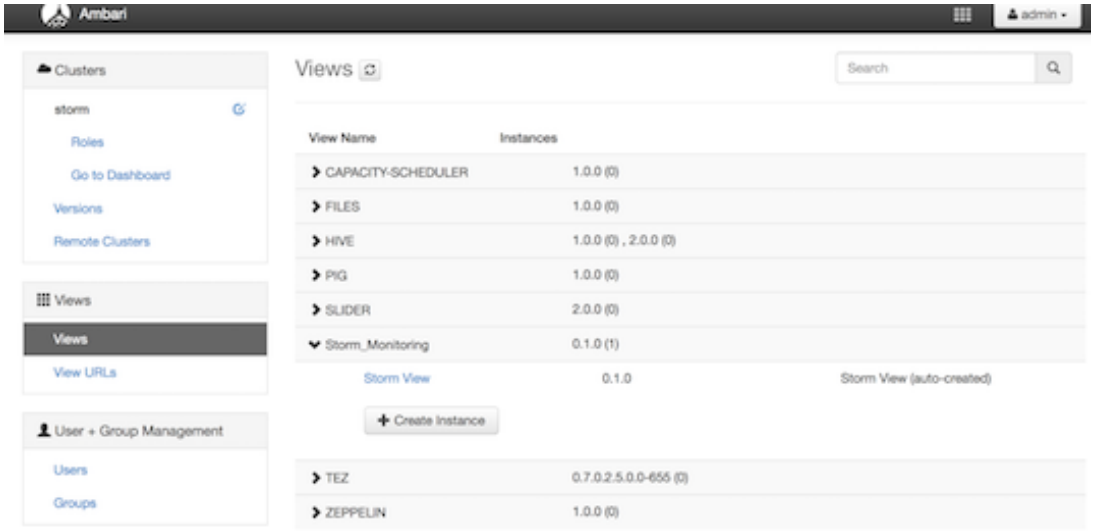
To create a Storm View instance:

**Steps**

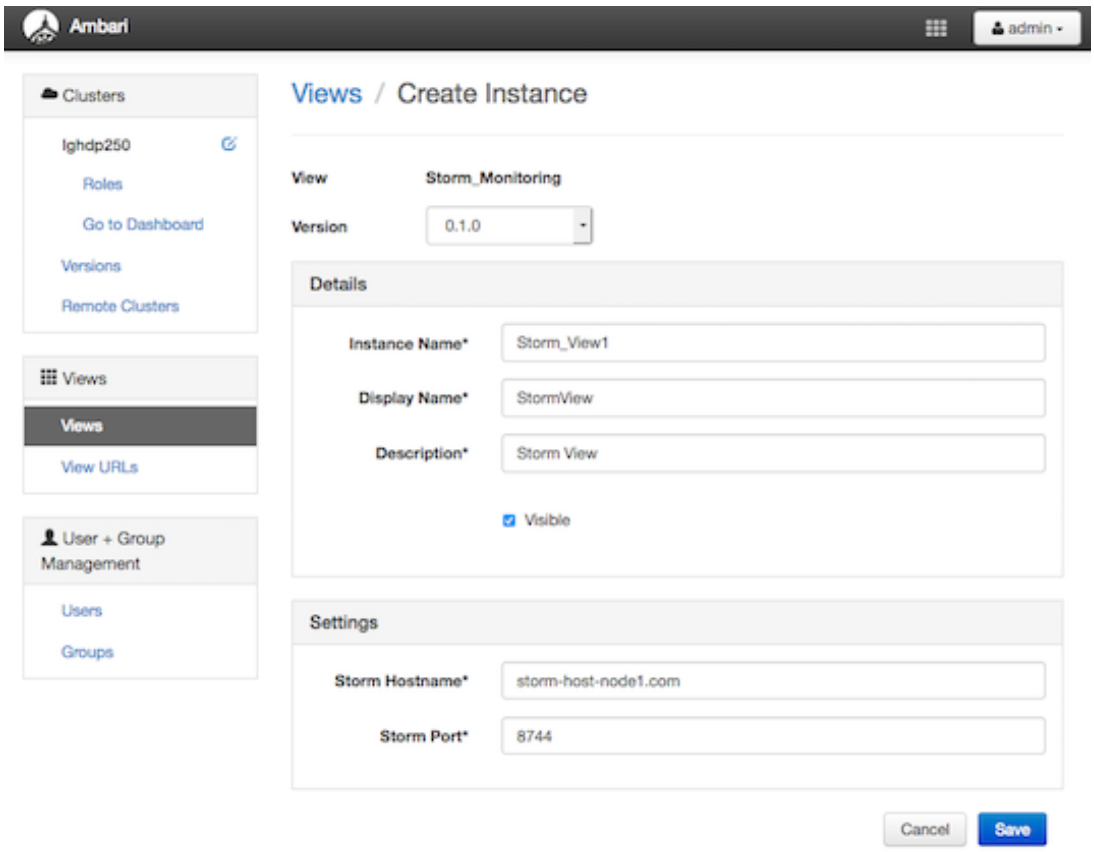
1. Browse to the Ambari Admin interface: from the dashboard, open the administrator account menu and click **Manage Ambari**:




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 2.3. 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 2.4. 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 Storm View with a Kerberos-enabled cluster.

6. Click **Save**.

**More Information**

Preparing Ambari Server for Views [5]

Running Ambari Server Standalone [7]

Set Up Kerberos for Ambari Server

Configuring Views for Kerberos

## 2.10.9. Configuring Tez View

This sections describes:

- Configuring Your Cluster for Tez View [63]
- Creating or Editing a Tez View Instance [64]

### 2.10.9.1. 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 Tez View on your standalone Ambari server.

To configure your cluster for Tez View:

**Steps**

1. Confirm the following configurations are set:

**Table 2.5. 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.

## 2.10.9.2. Creating or Editing a Tez View Instance

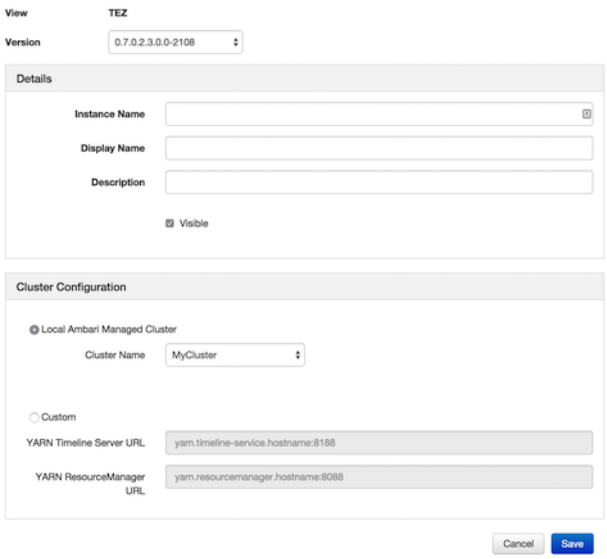
Depending on whether you must create a new Tez View instance for a manually deployed cluster or modify an Ambari-created Tez View, using one of the following procedures:

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

**Steps**

1. Browse to the Ambari Admin 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 2.13. 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 manually.

5. Click **Save**, grant Permissions on the view , then click **Go to instance** to use the view.

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

**Steps**

1. Browse to the Ambari Admin 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:

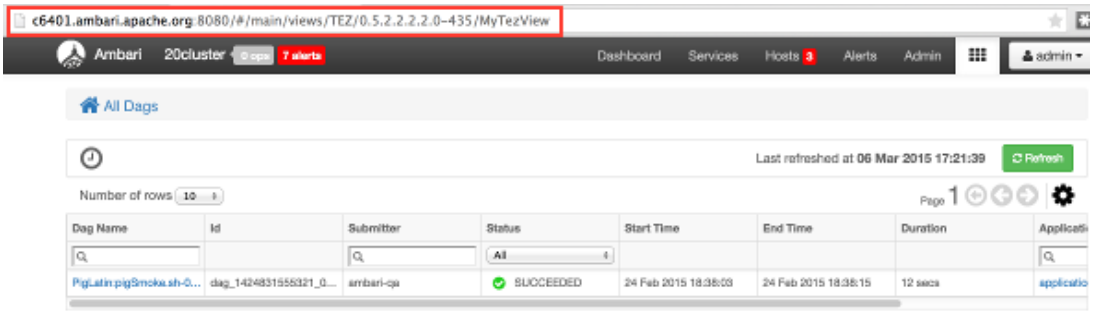
**Table 2.6. Cluster Configuration Values for Tez View in Ambari**

| 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`. |

| Property | Value |
|---|---|
| | When you enter the value in the view definition, pre-pend "http://" to the value you find in the yarn-site.xml file. For example, `http://<timeline server host>:8188`<br><br>**For wire encryption-enabled clusters:**<br><br>Set this based on the value of `yarn.timeline-service.webapp.https.address` in `yarn-site.xml`<br><br>When you enter the value in the view definition, pre-pend "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`.<br><br>When you enter the value in the view definition, pre-pend "http://" to the value you find in the yarn-site.xml file. For example, `http://<resourcemanager host>:8088`<br><br>**Important:** If YARN ResourceManager HA is enabled, provide a comma-separated list of URLs for all the Resource Managers.<br><br>**For wire encryption-enabled clusters:**<br><br>Set this based on the value of `yarn.resourcemanager.webapp.https.address` in `yarn-site.xml`<br><br>When you enter the value in the view definition, pre-pend "https://" to the value. For example, `https://<resourcemanager host>:8090` |

6. Click **Save** and grant Permissions on the view.
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 2.14. 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.

### 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.

**More Information**

User Permissions for Tez Views [67]

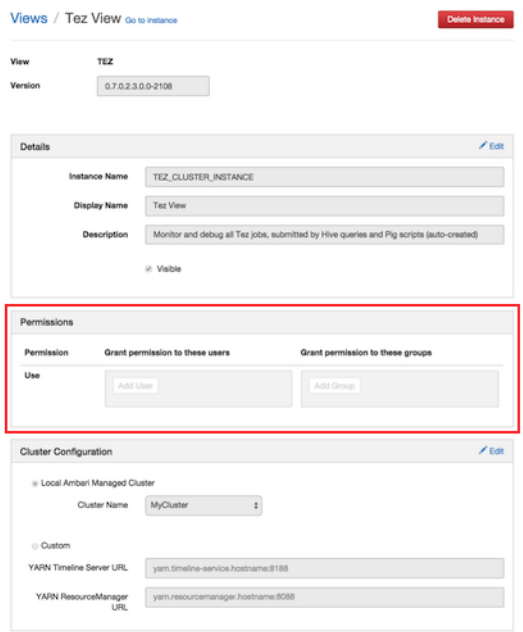
Using Tez View [115]

Kerberos Setup for Tez View [68]

#### 2.10.9.2.1. User Permissions for Tez Views

After saving the Tez View instance definition, grant permission on the view for all Tez View users:

**Figure 2.15. Granting User Permissions to Tez View**




### 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.

**More Information**

Managing Users and Groups

#### 2.10.9.2.2. Kerberos Setup for Tez View

**Prerequisites**

Set up basic Kerberos for views, on the Ambari Views server.

After you have set up basic Kerberos, Tez View requires that you set the following configuration properties:

**Steps**

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 2.7. Kerberos Settings for Tez View**

| 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` | * |
| `Timeline HTTP Auth` | `kerberos` |
| `RM HTTP Auth` | `kerberos` |

   **Note**

   Tez View will not work in a kerberized cluster, if Timeline HTTP Auth and RM HTTP Auth properties are not set to kerberos.

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

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

**More Information**

Set Up Kerberos for Ambari Server

## 2.10.10. Configuring Workflow Manager View

Before you can access Workflow Manager, you must complete several configuration tasks.

See the following content:

Section 2.10.10.1, “Configuring Your Cluster for Workflow Manager View” [69]

Section 2.10.10.2, “Kerberos Setup for Workflow Manager” [71]

Section 2.10.10.3, “Creating and Configuring a Workflow Manager View Instance” [72]

## 2.10.10.1. Configuring Your Cluster for Workflow Manager View

For Workflow Manager View to access HDFS, the Ambari Server daemon hosting the view must act as the proxy user for HDFS. This allows Ambari to submit requests to HDFS on behalf of the Worklow Manager View users.

Each Worklow Manager View user must have a user directory set up in HDFS.

If the cluster is configured for Kerberos, ensure that the Section 2.8, "Configuring Views for Kerberos" [13] has been completed.

### 2.10.10.1.1. Set up HDFS Proxy User

**Note**

If you previously set up the proxy user for another View, you can skip this task.

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**.

**Steps**

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.

**More Information**

Ambari Server for Kerberos

#### 2.10.10.1.2. Set up HDFS User Directory

You must set up a directory for each user that accesses the Workflow Manager View. Workflow Manager View stores user metadata in the user directory in HDFS.

**About This Task**

By default, the location in HDFS for the user directory is `/user/${username}`, where `${username}` is the username of the currently logged in user that is accessing Workflow Manager View.

**Important**

Since many users leverage the default Ambari *admin* user for getting started with Ambari, you should create a `/user/admin` directory if one does not exist, in addition to directories for other Workflow Manager View users.

**Steps**

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

   ```
   su - hdfs
   ```

3. If working on a secure Kerberos cluster:
   a. Destroy any existing Kerberos tickets:

      ```
      kdestroy
      ```

      If no ticket is found, you get an error message that you can ignore: No credentials cache file found while destroying cache

   b. Obtain a Kerberos ticket-granting ticket:

      ```
      kinit -kt /etc/security/keytabs/hdfs.headless.keytab hdfs
      ```

4. Using the HDFS client, make an HDFS directory for the user.

   For example, if your username is *wfmadmin*, you would create the following directory.

   ```
   hadoop fs -mkdir /user/wfmadmin
   ```

5. Set the ownership on the newly created directory.

   For example, if your username is *wfmadmin*, the directory owner should be *wfmadmin:hadoop*.

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

6. Log in as root user.

   ```
   su -
   ```

7. Access the Kerberos administration system.

```
kadmin.local
```

8. Create a new principal and password for the user.

   You can use the same user name and password that you used for HDFS directory.

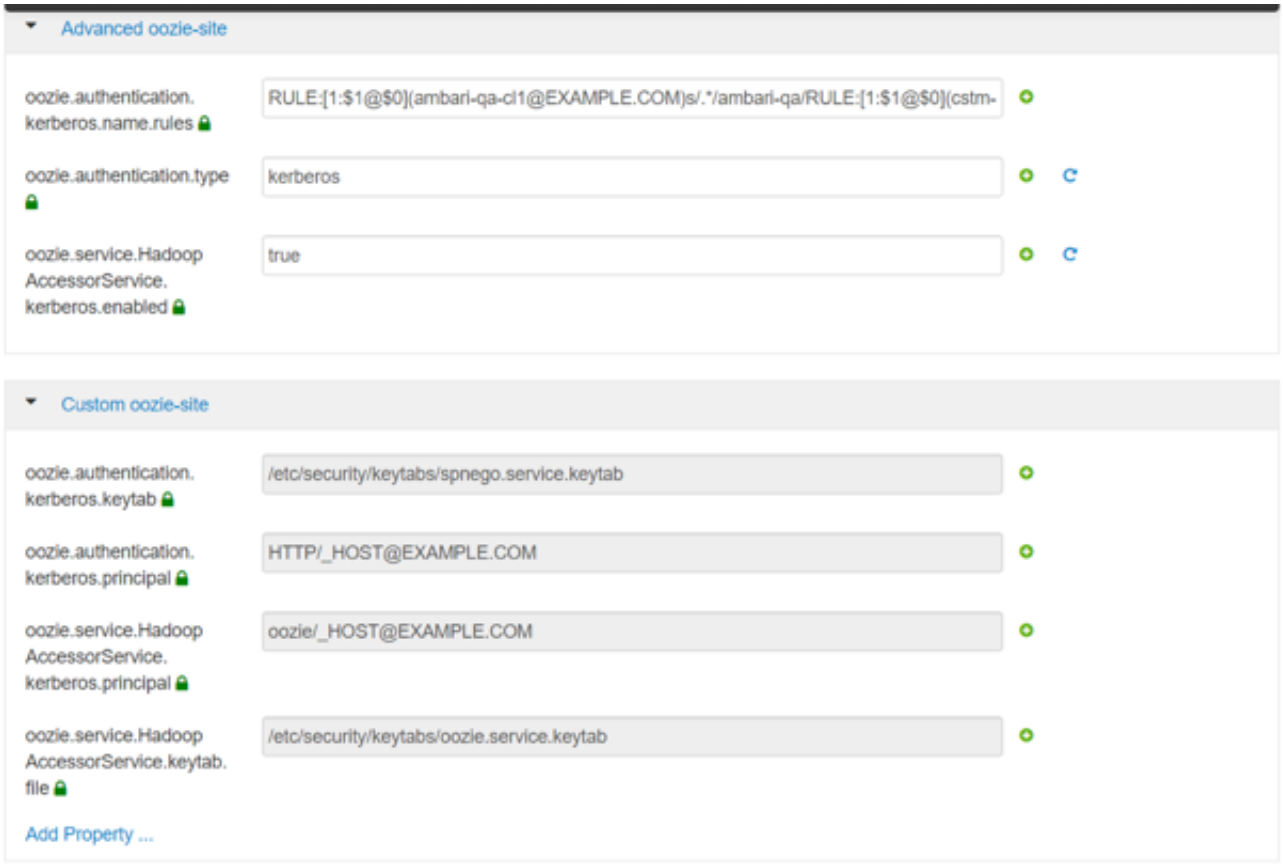
   ```
   addprinc -pw [wfmadmin-password] wfmadmin@EXAMPLE.COM
   ```

9. Repeat steps 2 through 8 for any additional Workflow Manager View users.

## 2.10.10.2. Kerberos Setup for Workflow Manager

If you install Ambari using Kerberos, the Kerberos settings for Oozie that are required for Workflow Manager are configured automatically. The image below shows what the settings should look like.

### Figure 2.16. Workflow Manager Kerberos Configuration for Oozie




### 2.10.10.2.1. Set up Proxy User for Oozie

If you are using Kerberos, you must configure the proxy user for Oozie. Workflow Manager uses Oozie as its scheduling engine.

**Steps**

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

2. Expand the **Custom oozie-site** section.

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

   oozie.service.ProxyUserService.proxyuser.[*ambari-server-cl1*].groups=*

   oozie.service.ProxyUserService.proxyuser.[*ambari-server-cl1*].hosts=*

   Replace *ambari-server-cl1* with the server principal name used when configuring Kerberos.

   **Figure 2.17. wfm-oozie-proxy-user.png**

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

**More Information**

Set Up Kerberos for Ambari Server

## 2.10.10.3. Creating and Configuring a Workflow Manager View Instance

You can configure multiple WFM View instances. You might want to have multiples instances if you want to assign different users and permissions for each instance. You can also have instances that run locally and others that run remotely.

**Steps**

1. Click **Manage Ambari** to open the Ambari Admin user interface.

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

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

   If multiple View versions are present, choose one.

4. Enter the following view instance Details:

### Table 2.8. Workflow Manager View Instance Details

| Property | Description | Example Value |
|---|---|---|
| Instance Name | This is the Workflow Manager View instance name and must be unique for all instances you create. The instance name cannot be modified after the instance is created. | wfm_local_instance<br><br>This value cannot contain spaces or special characters other than an underscore. |
| Display Name | This is the name of the view link displayed to the user in Ambari Web. The display name can be modified after the instance is created. | WFM View<br><br>This value can contain spaces and underscores, but no other special characters. |
| Description | This is the description of the view displayed to the user in Ambari Web. | Local instance of WFM |
| Visible | This checkbox determines whether the view is displayed to users in Ambari Web. | Visible or Not Visible |

5. Under Cluster Configuration, select the **Local Cluster** or **Remote Cluster**instance type and select the cluster name.

6. Review the remaining settings and make changes as desired.

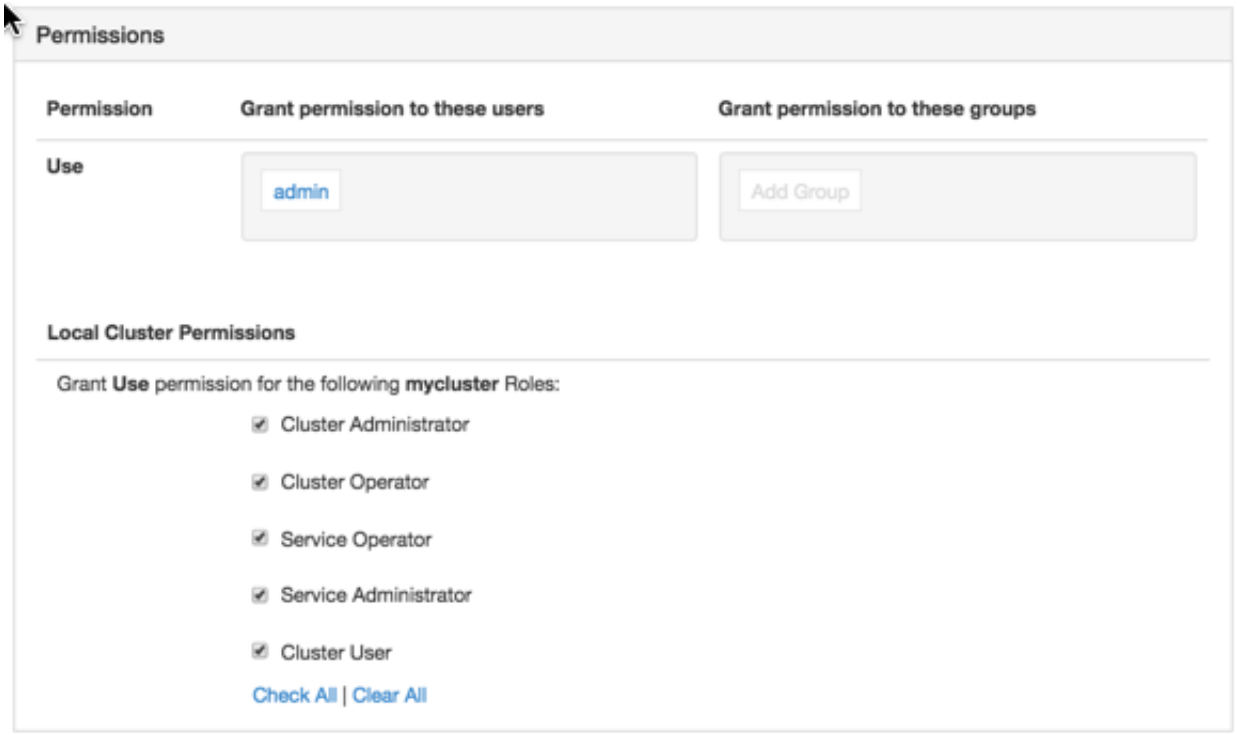
   The Settings and Cluster Configuration options depend on a few cluster and deployment factors in your environment. Typically, you can accept the default Settings unless you are using the Workflow Manager View with a Kerberos-enabled cluster.

7. Click **Save**.

   The instance is created and a success message displays.

8. Scroll to the **Permissions** section at the bottom of the Views configuration form.

9. Grant permission on the Workflow Manager View for the set of users and groups who can access the view instance.




**More Information**

For descriptions of the View instance types, see Section 2.3, "Configuring View Instances" [9].

You can access the Workflow Manager documentation in the Workflow Management guide.

# 3. Using 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.

The YARN Queue Manager View is designed to help hadoop operators configure workload management policies for YARN. In 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.

**Next Steps**

- Setting up Queues [75]
- Configuring Queues [80]
- Enabling Preemption [81]
- Setting YARN Queue Priorities [85]
- Configuring Cluster Scheduler Settings [87]
- Applying the Configuration Changes [88]

## 3.1. Setting up Queues

To set up Capacity Scheduler queues on a view instance.

**Steps**

1. On the YARN Queue Manager view instance configuration page, click **Add Queue**.

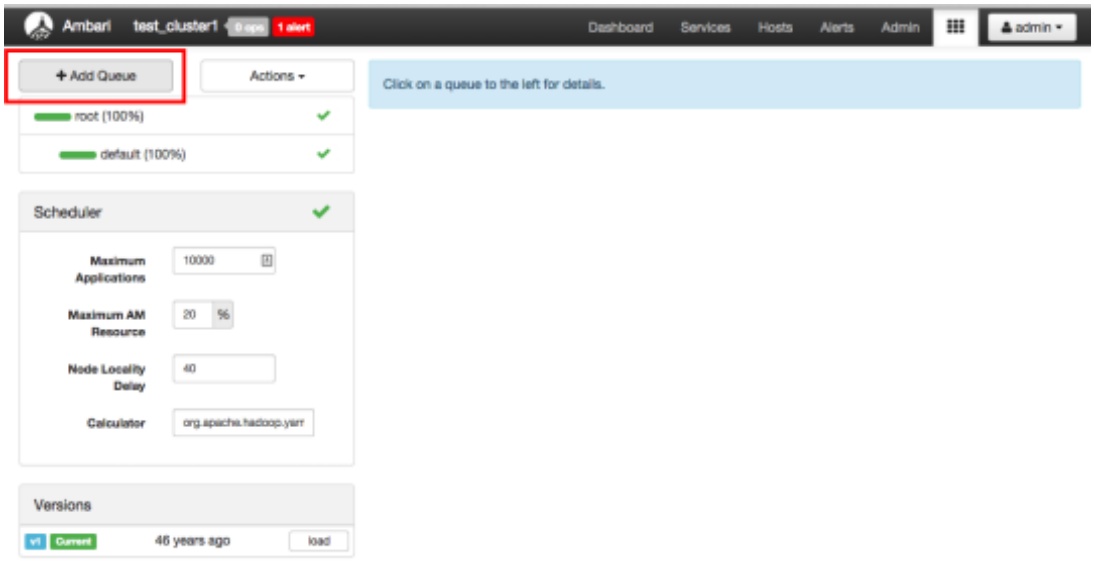
   The queue will be added under the top level, or "root" queue. A "default" queue already exists under the root queue.

   To return to a *previously created* YARN Queue Manager view instance:

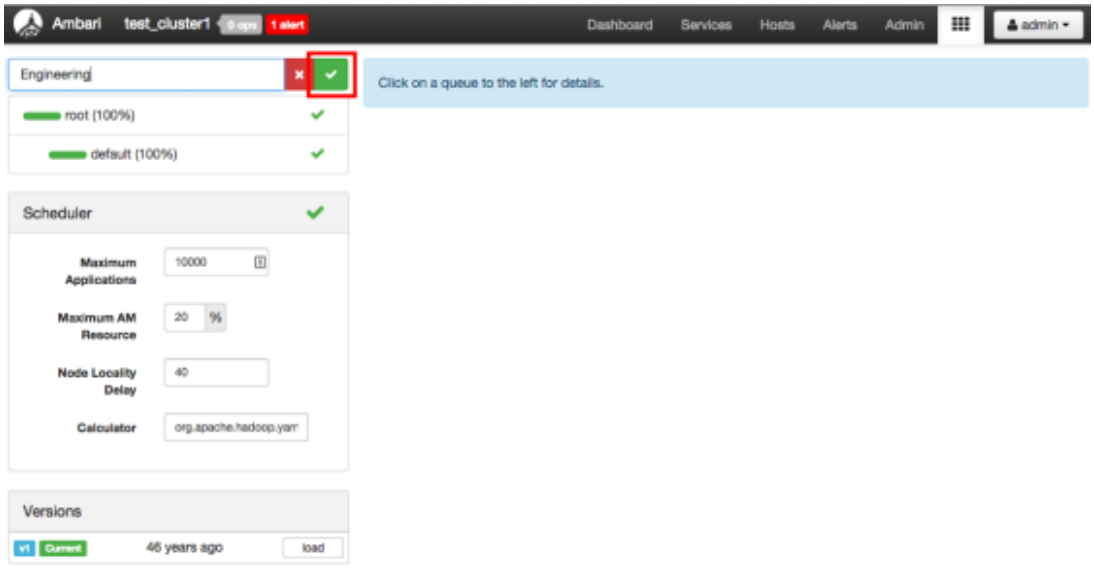
   a. Click **Views** on the Manage Ambari page.

   b. Click **CAPACITY-SCHEDULER**.

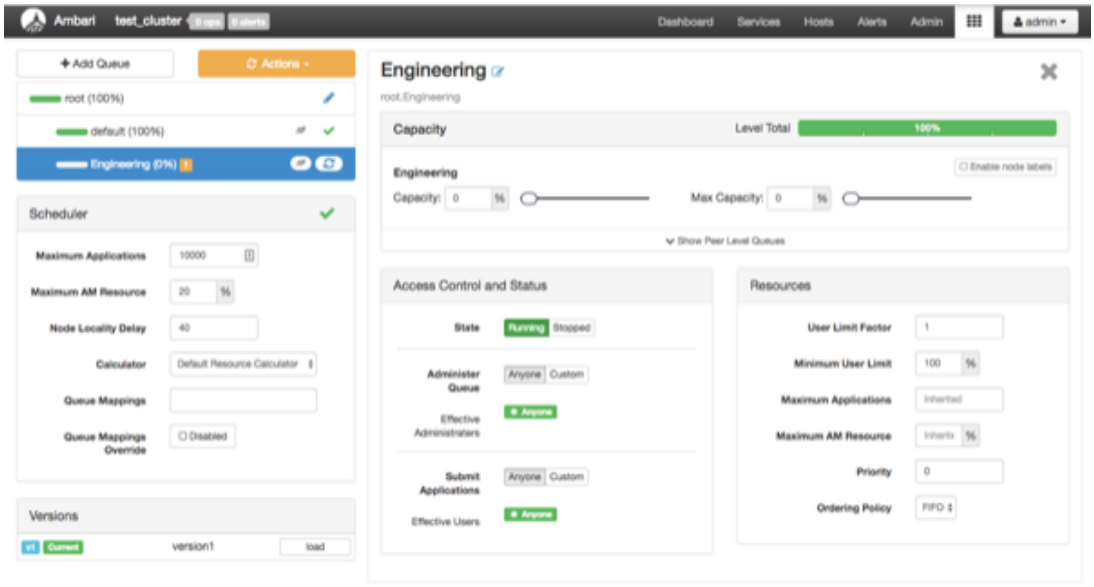
   c. Click the applicable YARN Queue Manager view instance, then click **Go to instance** at the top of the page.



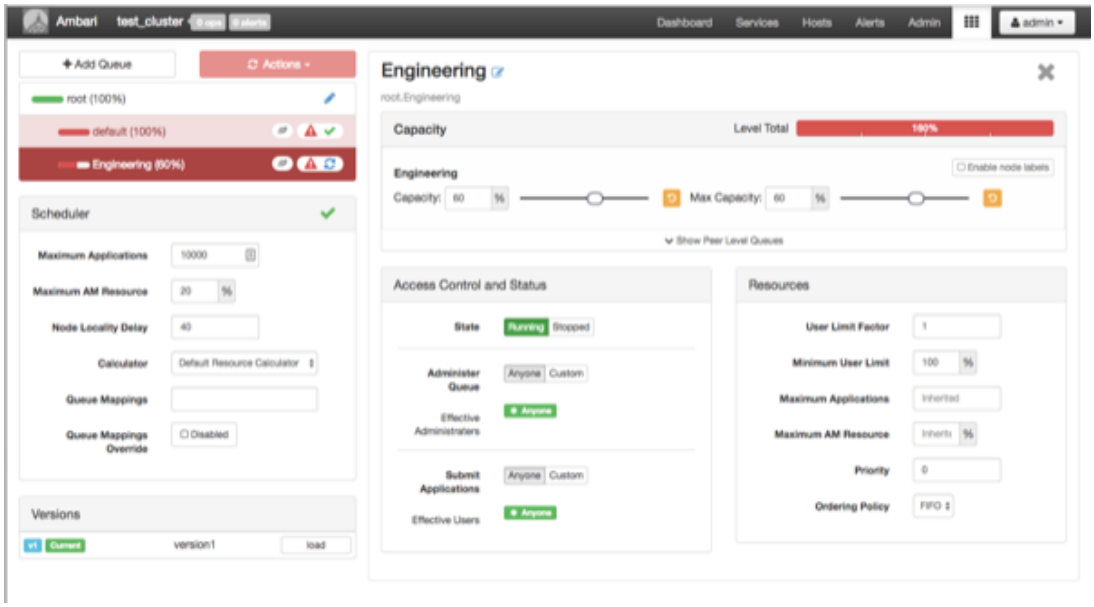

2. Type in a name for the new queue, then click the green check mark to create the queue. In the following example, we're creating the "Engineering" queue.



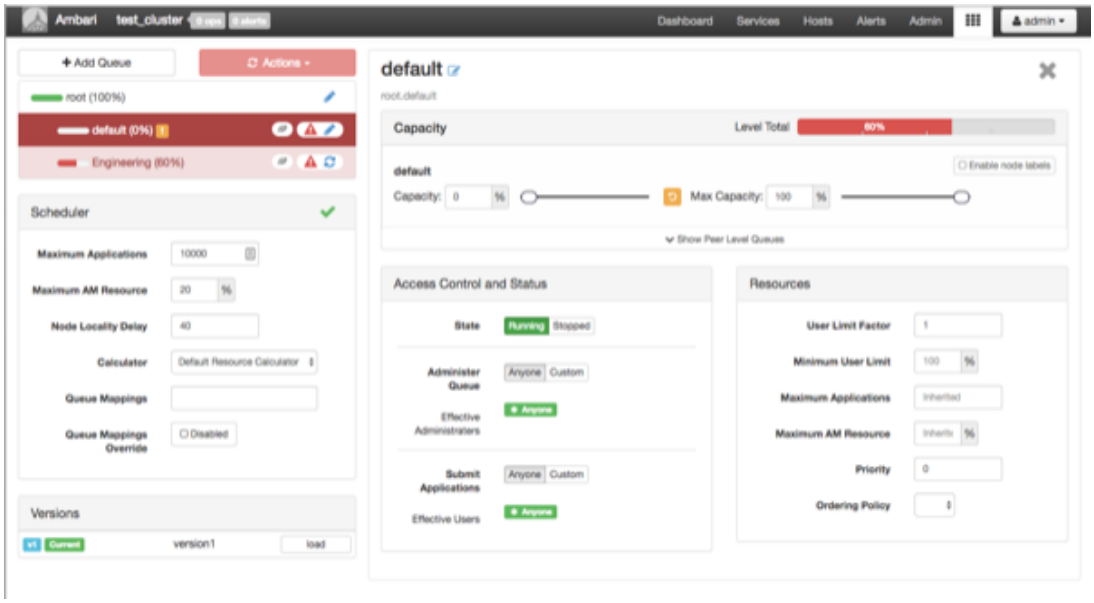

3. The "Engineering" queue is added, and its configuration page appears.



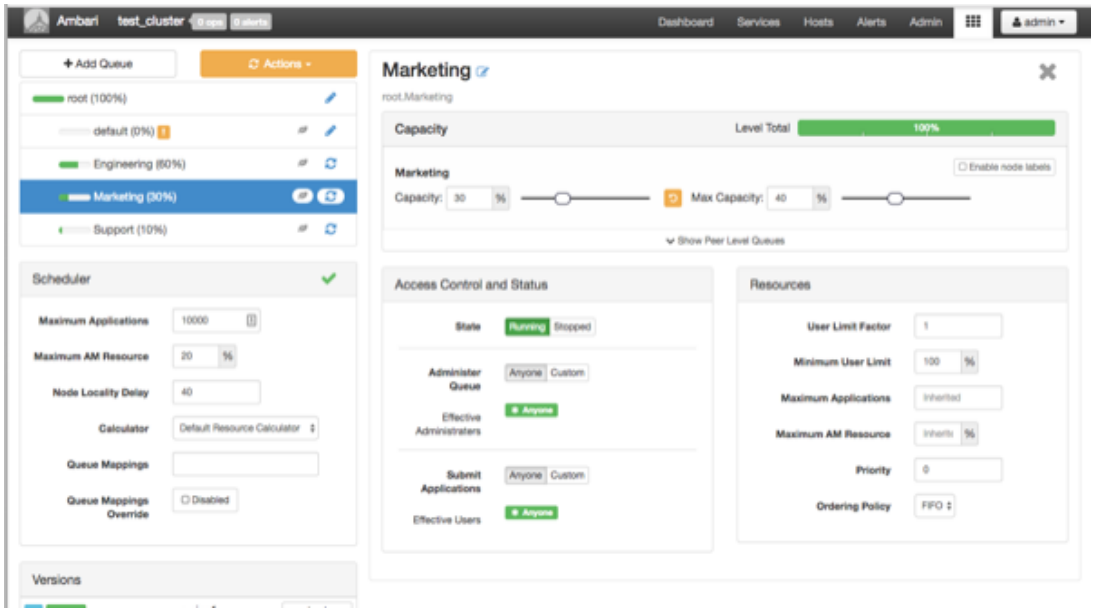

4. The sum of queue capacities at any level in the YARN Queue Manager configuration must total 100%. Here the default queue is already set to 100%. Therefore, if we try to set the "Engineering" queue capacity to 60%, error messages appear warning that the total at this level is 160%.




5. If we click the "default" queue and set its capacity to 0%, the Level Total bar at the top of the page lists the total queue capacity at this level as 60%.



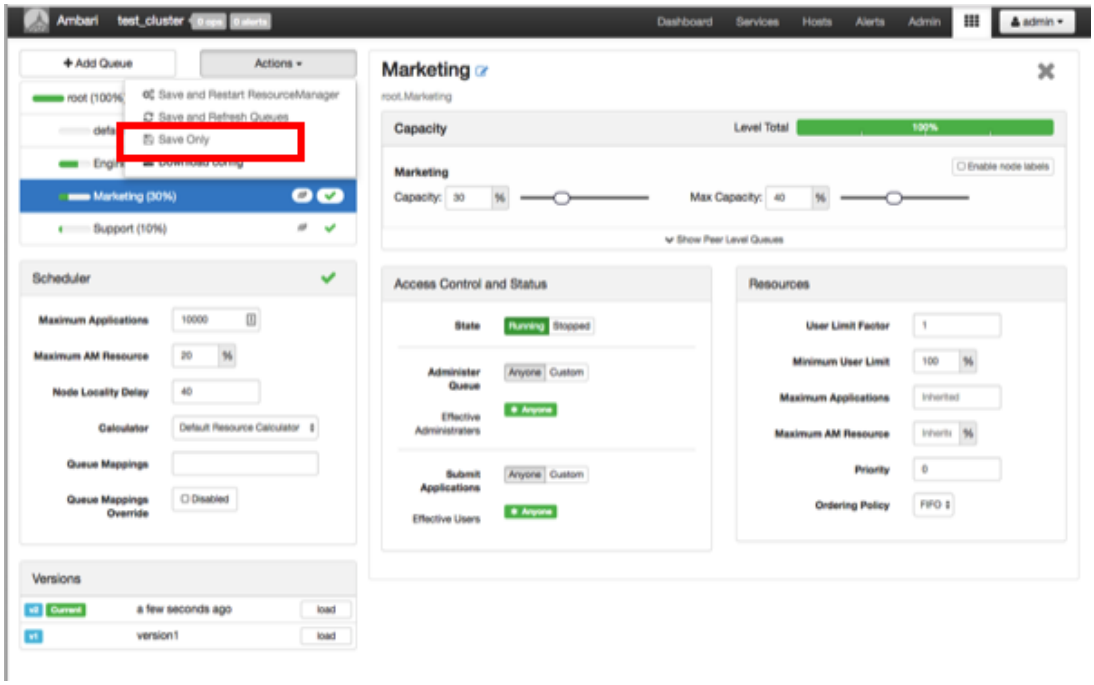

6. To add more queues at the root level, click the **root** queue, then click **Add Queue**. In the following example, we have added a "Support" queue set to 10% of the level capacity, and a "Marketing" queue set to 30%. The root-level queue capacities now total 100%.




7. To save your configuration, click **Actions > Save Only**.

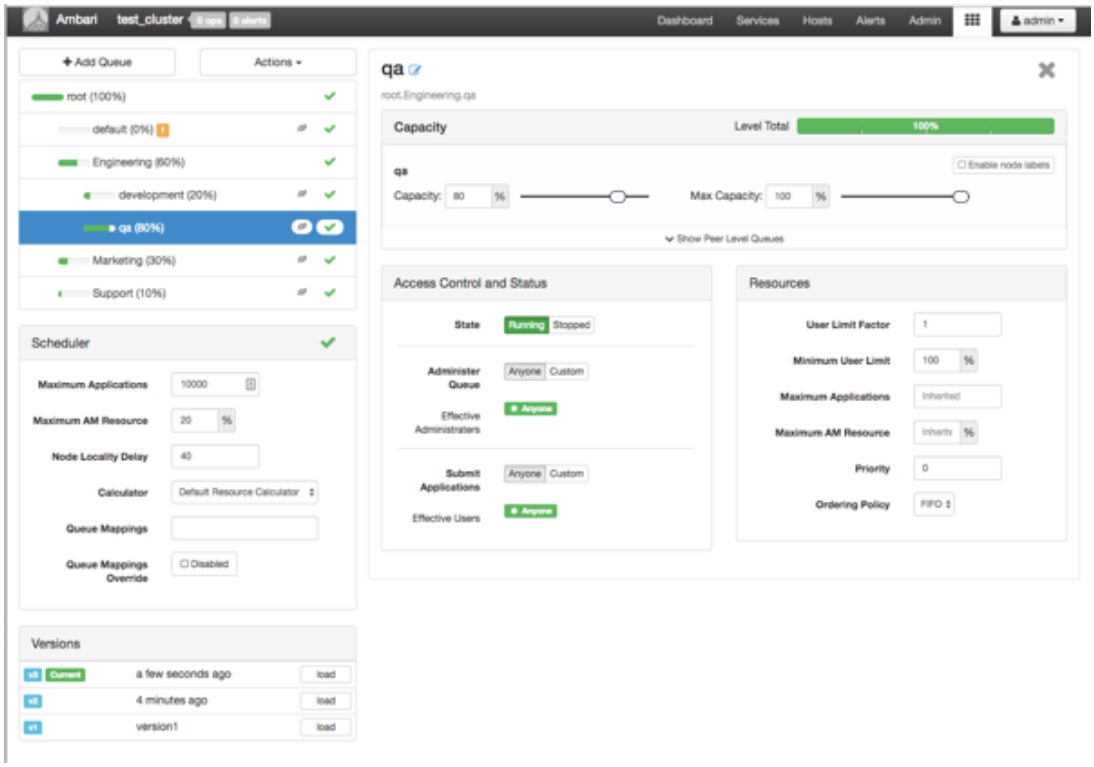
8. On the **Notes** pop-up, enter an optional description of your changes, then click **Save**.

   Each version is retained and listed in the Versions box.




9. 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.



# 3.2. Configuring Queues

To configure a queue:

**Steps**

1. Click the queue name.
2. 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**

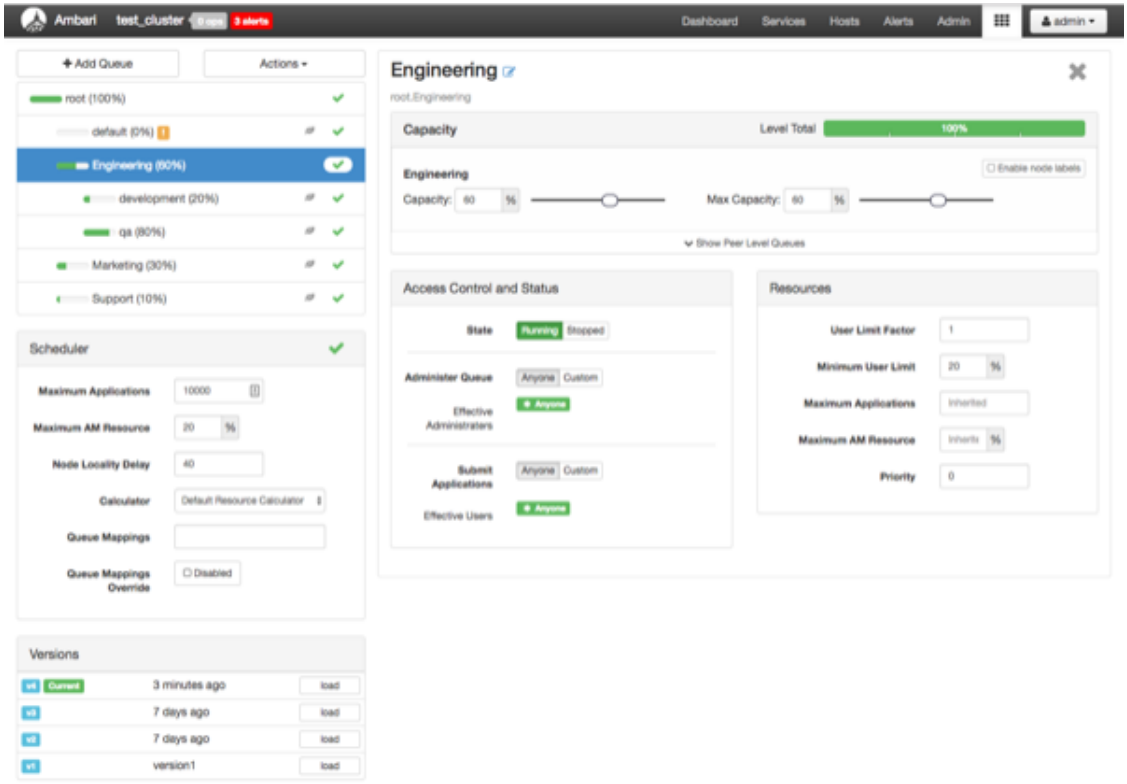
- State – Running is the default state. Setting this to Stopped lets you gracefully drain the queue of jobs (for example, before deleting a queue).
- Administer Queue – Click **Custom** to restrict administration of the queue to specific users and groups.
- Submit Applications – Click **Custom** to restrict 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. The default setting is Inherited (no override).

- Maximum AM Resource – This setting enables you to override the Scheduler Maximum AM Resource setting. The default setting is Inherited (no override).
- Priority – An integer value that sets a relative priority for a queue. The default value is 0 for all queues. Setting this to a higher value gives a queue access to cluster resources ahead of queues with lower priorities. In order for YARN Queue Priorities to be applied, you must enable preemption. For more information see Setting YARN Queue Priorities.

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




**More Information**

Enabling Preemption [81]

Setting YARN Queue Priorities [85]

Configuring Cluster Scheduler Settings [87]

# 3.3. Enabling Preemption

**About This Task**

When using YARN queues, a scenario can occur in which a queue has a guaranteed level of cluster resources, but must wait to run applications because other queues are utilizing all of the available resources. If Preemption is enabled, higher priority applications do not have to wait because lower priority applications have taken up the available capacity. With Preemption enabled, under-served queues can begin to claim their allocated cluster

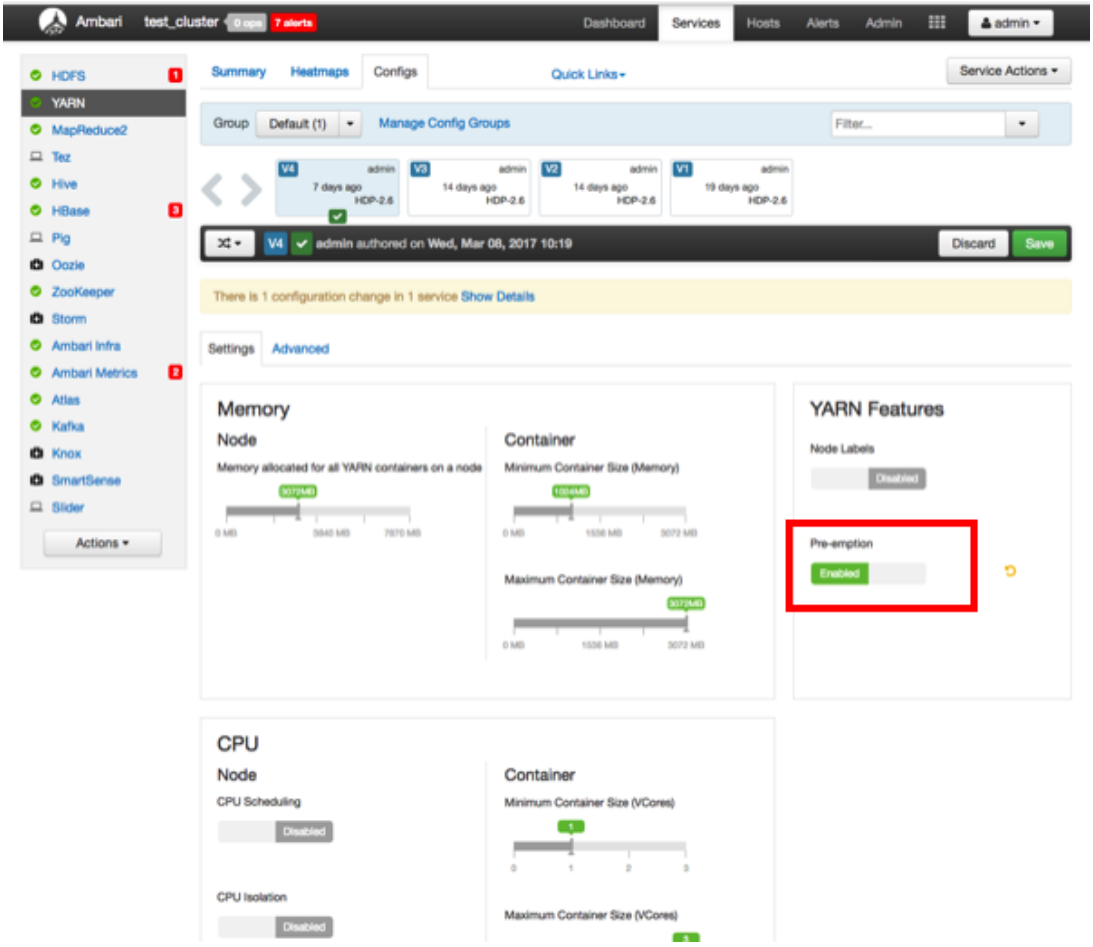
resources almost immediately, without having to wait for other queues' applications to finish running.

## Note

For more information about Preemption, see Better SLAS Via Resource Preemption in the YARN Capacity Scheduler.

**Steps**

1. On the Ambari dashboard, select **YARN > Configs**. Under YARN Features, click **Pre-emption**. The button label changes to indicate that Preemption is enabled.




2. Select the **Advanced** tab, then scroll down and select **Custom yarn-site**. Click **Add Property** and use the Add Property pop-up to add a new property in the following format:

```
yarn.resourcemanager.monitor.capacity.preemption.total_preemption_per_round=
<(memory-of-one-NodeManager)/(total-cluster-memory)>
```

This is the maximum percentage of resources preempted in a single round. You can use this value to restrict the pace at which Containers are reclaimed from the cluster. After computing the total desired preemption, the policy scales it back to this limit. This should be set to (memory-of-one-NodeManager)/(total-cluster-memory). For

example, if one NodeManager has 32 GB, and the total cluster resource is 100 GB, the total_preemption_per_round should set to 32/100 = 0.32. The default value is 0.1 (10%):

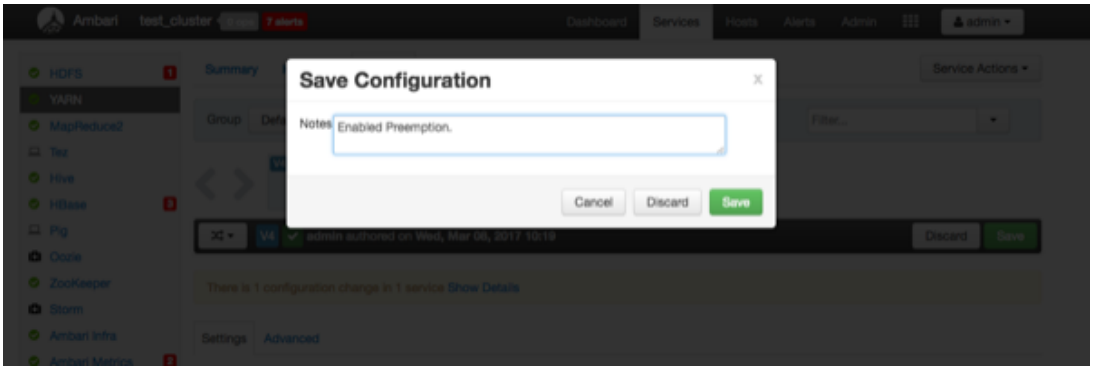
```
yarn.resourcemanager.monitor.capacity.preemption.total_preemption_per_round=
0.1
```

3. Click **Add Property** again and use the Add Property pop-up to add the following **Custom yarn-site** property:

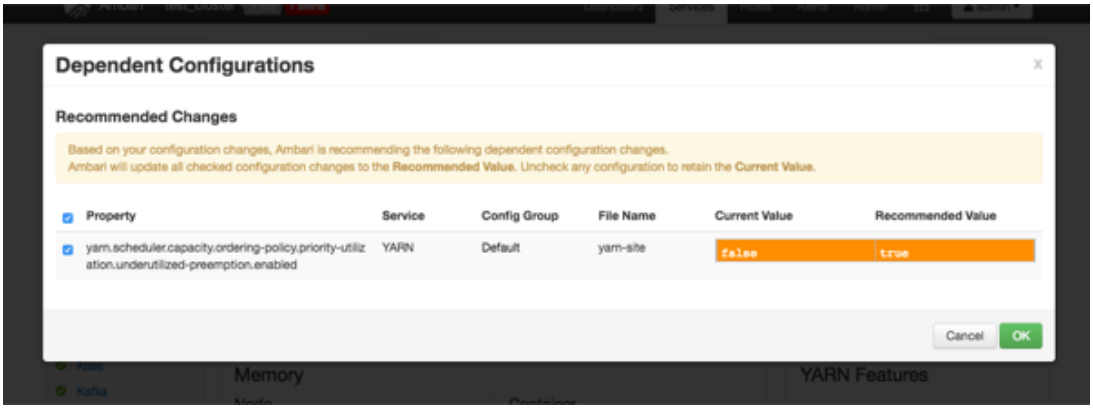
```
yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor=
1.0
```

Similar to total_preemption_per_round, you can apply this factor to slow down resource preemption after the preemption target is computed for each queue (for example, "give me 5 GB back from queue-A"). For example, if 5 GB is needed back, in the first cycle preemption takes back 1 GB (20% of 5GB), 0.8 GB (20% of the remaining 4 GB) in the next, 0.64 GB (20% of the remaining 3.2 GB) next, and so on. You can increase this value to speed up resource reclamation. The recommended value for this parameter is 1.0, meaning that 100% of the target capacity is preempted in a cycle.

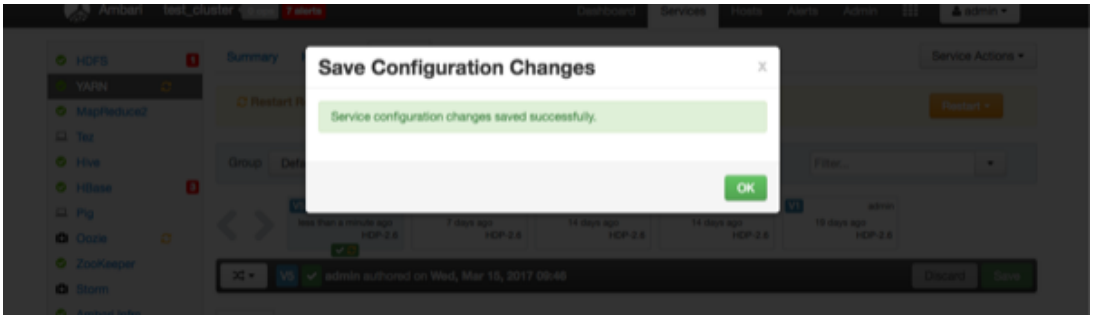
4. Click **Save** to save the new configuration settings. On the Save Configuration pop-up, type a description of the changes in the Notes box, then click **Save**.



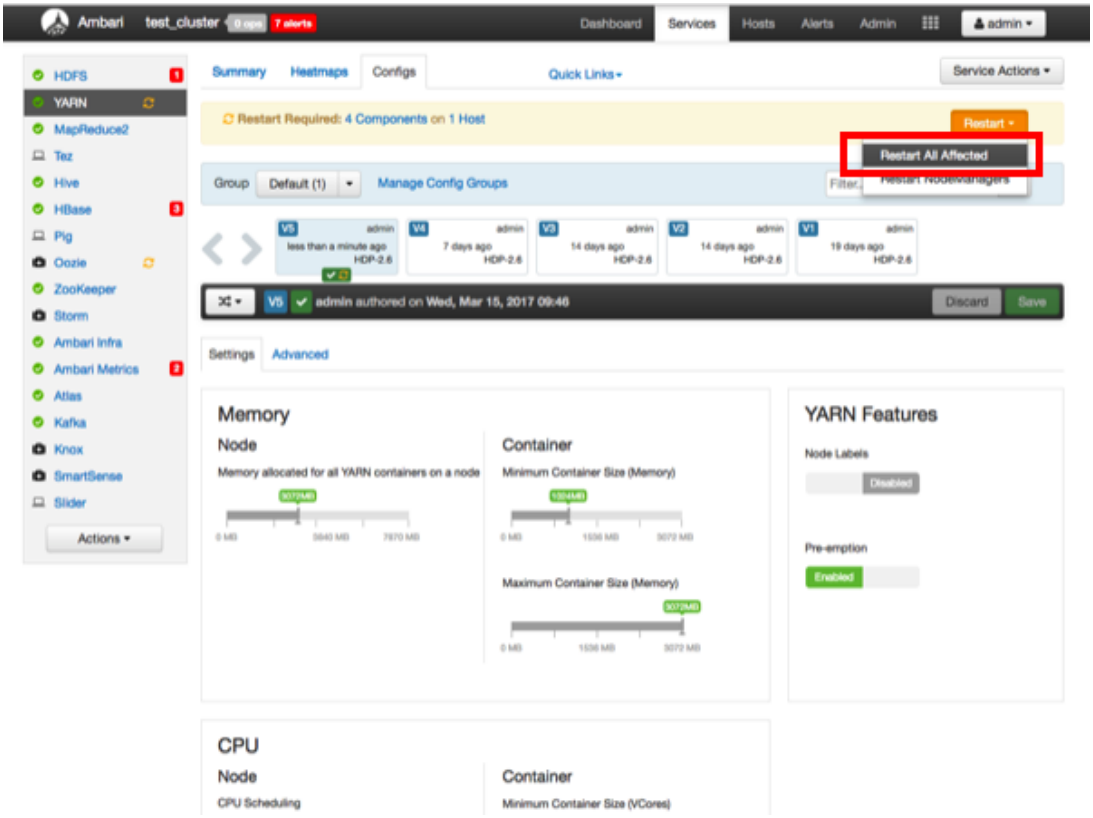

5. On the Dependent Configurations pop-up, click **OK** to accept the recommended value of `true` for the `yarn.scheduler.capacity.ordering-policy.priority-utilization.underutilized-preemption.enabled` property.



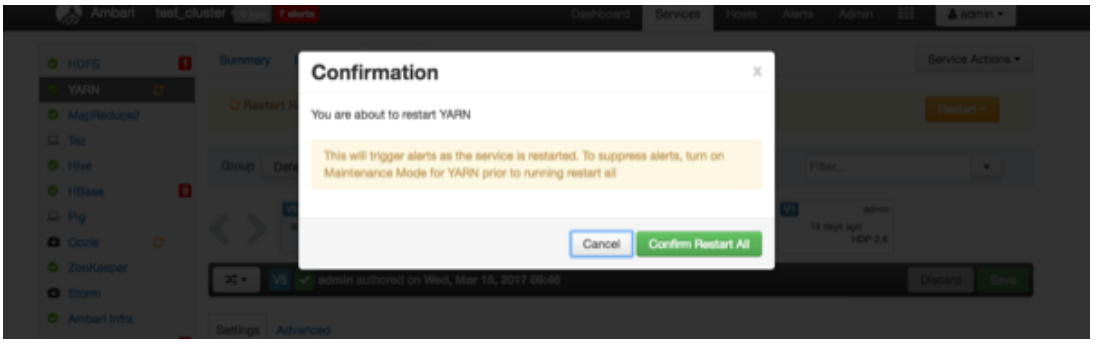

6. Click **OK** on the Save Configuration Changes pop-up.




7. Select **Restart > Restart All Affected** to restart the YARN service and load the new configuration.




8. Click **Confirm Restart All** on the confirmation pop-up to confirm the YARN restart.



9. After YARN restarts, Preemption will be enabled. Other components may also require a restart.

# 3.4. Setting YARN Queue Priorities

**About This Task**

Even with preemption enabled, there are some use cases where applications might not have access to cluster resources without setting priorities:

- Long-running applications – Without setting priorities, long-running applications in queues that are under capacity and with lower relative resource usage may not release cluster resources until they finish running.

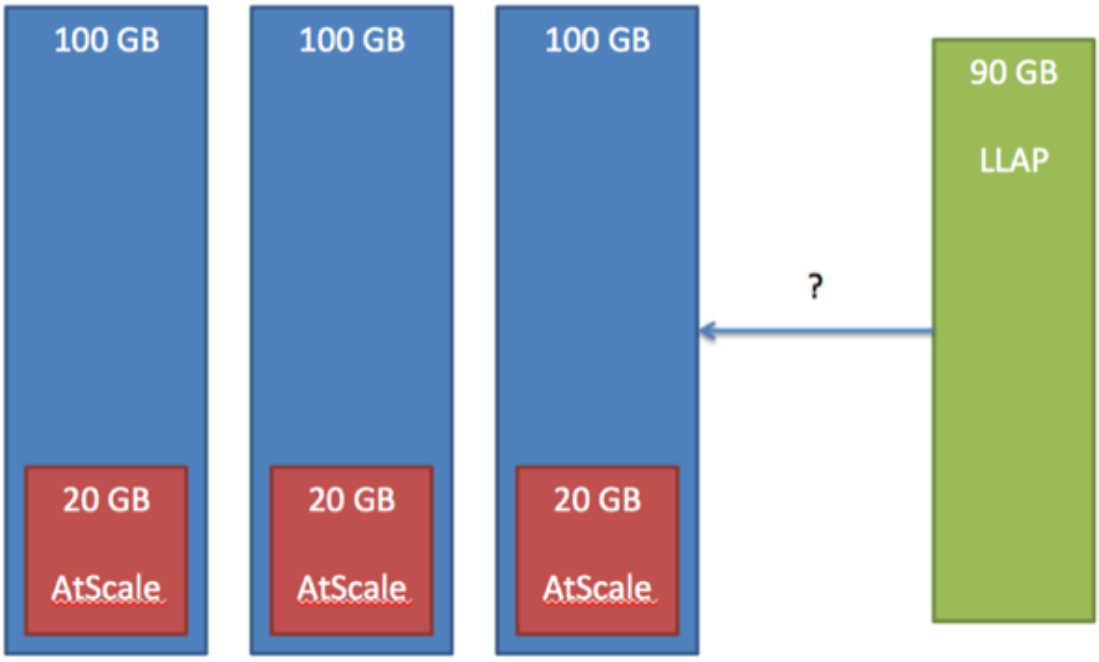
- Applications that require large containers – The issue with long-running applications is exacerbated for applications that require large containers. With short-running applications, previous containers may eventually finish running and free cluster resources for applications with large containers. But with long-running services in the cluster, the large containers may never get sufficiently large resources on any nodes.

- Hive LLAP – Hive LLAP (Low-Latency Analytical Processing) enables you to run Hive queries with low-latency in near real-time. To ensure low-latency, you should set the priority of the queue used for LLAP to a higher priority, especially if your cluster includes long-running applications.

## Note

To set the queue used for Hive LLAP, select **Hive > Config > Settings** on the Ambari dashboard, then select a queue using the **Interactive Query Queue** drop-down. For more information, see the Hive Performance Tuning guide.

For example, the following figure shows a 3-node cluster with long-running 20 GB containers. The LLAP daemons require 90 GB of cluster resources, but preemption does not occur because the available queues are under capacity with lower relative resource usage. With only 80 GB available on any of the nodes, LLAP must wait for the long-running applications to finish before it can access cluster resources.



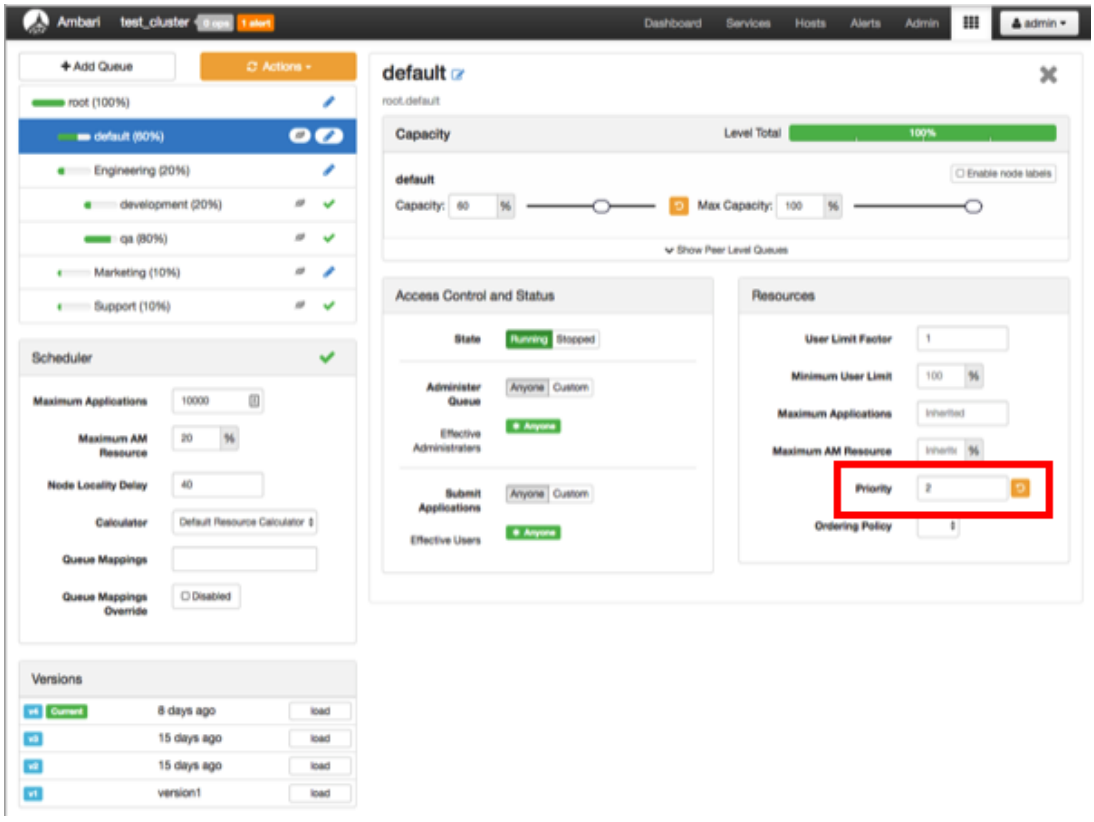

**Prerequisites**

**Important**

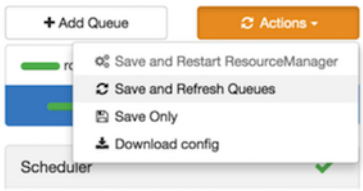
In order for YARN Queue Priorities to be applied, you must enable preemption.

**Steps**

1. On the YARN Queue Manager view, select a queue, then enter a priority in the **Priority** box under Resources. All queues are set to a priority of 0 by default. Higher numbers indicate higher priority.

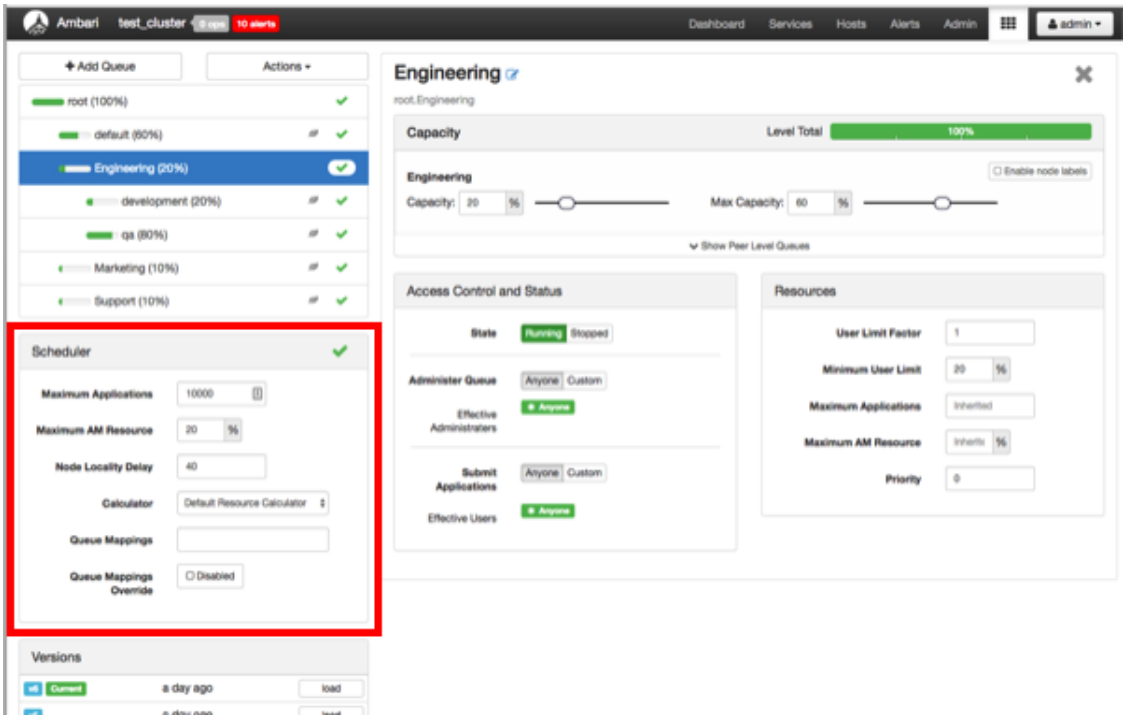



2. Select **Actions > Save and Refresh Queues** to save the priority setting.




# 3.5. 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:

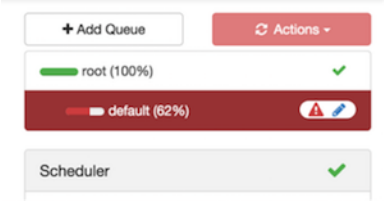
- Maximum Applications – To avoid system-thrash due to an unmanageable load – caused either by malicious users, or accidentally – the Capacity Scheduler enables you to place a static, configurable limit on the total number of concurrently active (both running and pending) applications at any one time. 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.
- Queue Mappings – You can use this box to specify default queue mapping based on user or group.
- Queue Mappings Override – Select this box to enable override of default queue mappings.

# 3.6. Applying the Configuration Changes

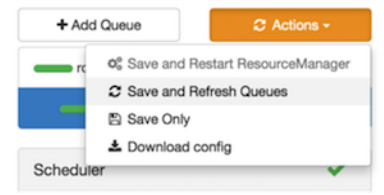
You can use the Actions menu to apply configuration changes made to the queue hierarchy.

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

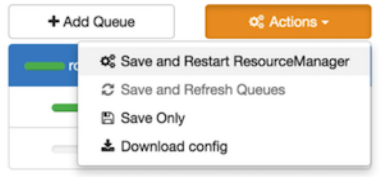
For changes that are not valid and cannot be applied, the **Actions** button will turn red, and the menu will not appear.




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**.



# 4. Using Files View

**Files View** provides a convenient way to access files and folders in your cluster's file system, using a browser-style user interface. Ambari creates a Files View instance and one default Files View automatically during cluster deployment. The default Files View points to the HDFS file system.

You can also create a custom view from the Files View instance that points to an Amazon Web Services (AWS) S3 file system bucket.

To create a custom Files View to work with files in S3, provide the bucket URL, and valid, AWS credentials in the **Settings > View Configs** field when you create the view instance that accesses an S3 bucket. You must enter the AWS credentials as a single string containing no space characters using the following syntax:

**fs.defaultFS=s3a://**<bucketURL>**/;fs.s3a.access.key=**<validaccesskeystring>**;fs.s3a.secret.key**<validsecretkey

–

Files View supports the following tasks:

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

**More Information**

Creating View Instances [10]

About Access Keys

# 5. Using 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.

**More Information**

Data Movement and Integration

# 6. Using Hive View 2.0

## Important

This chapter focuses on Hive View 2.0. For information about how to use Hive View 1.5, see Using the Hive View in version 2.4.2 of the *HDP Apache Ambari Views Guide*.

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. **Hive View** is designed to help you author, optimize, and execute Hive queries.

Use Hive View to:

- Browse databases
- Write queries or browse query results in full-screen mode, which can be particularly helpful with complex queries or large query results
- Manage query execution jobs and history
- View existing databases, tables, and their statistics
- Create tables and export table DDL to source control
- View visual explain plans

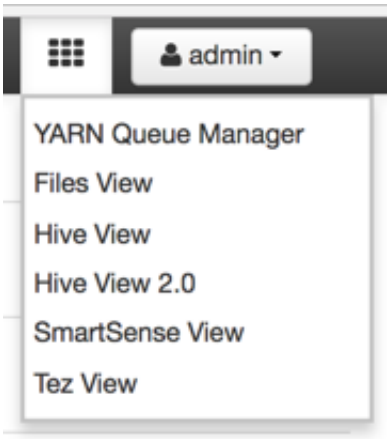
## Tip

**Consider using the Tez View of Ambari as a complement to Hive View if you are using the Tez execution engine. The Tez View helps you debug Hive queries by displaying performance metrics and envisioning the Directed Acyclic Graph (DAG) process at runtime.**

**There are two simple ways to access either a Hive View or the Tez View in Ambari:**

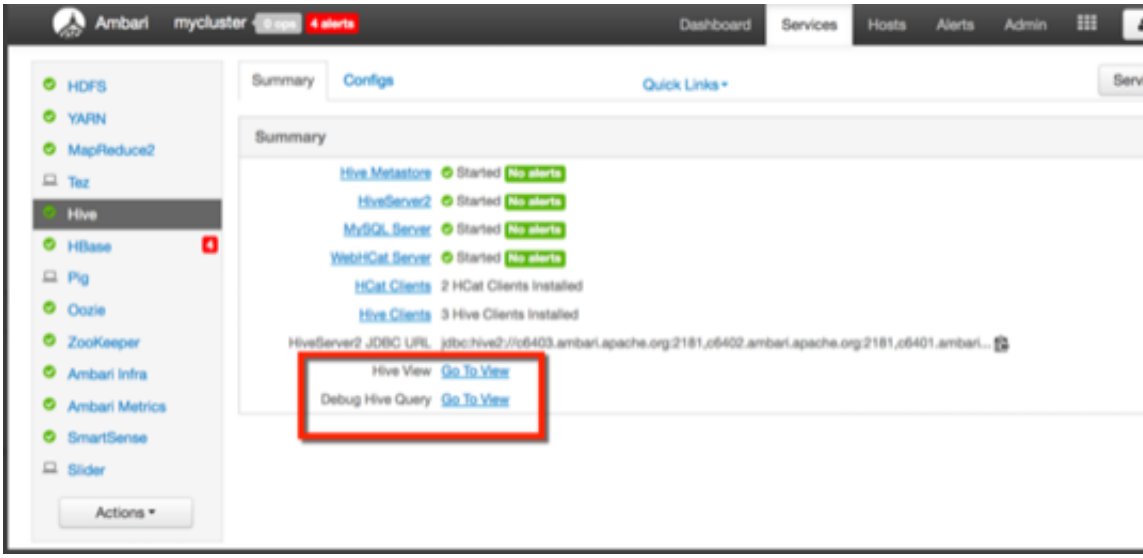
- Click the views tile in the upper right corner, and select the Hive View or Tez View that you want to use:

**Figure 6.1. Views Menu of Ambari**




- Alternatively, you can click a **Hive View** hyperlink or the **Debug Hive Query** hyperlink (which opens the Tez View) on the **Summary** tab when the Hive Service is selected in Ambari:

**Figure 6.2. Links to Hive-Related Views in Ambari**




**More Information**

Query Tab [94]

Saved Queries Tab [103]

UDFs Tab [103]

# 6.1. Query Tab

The **Query** tab helps you write, plan, execute, and save queries with embedded tools for locating Hive databases and tables of your system in the same window.

**Important Features of the Query Tab**

- Use the **Browse** button to find databases. Click on a database to enable it as a selectabe database in Hive View. The **Browse** button acts as a filter in case you are working with more databases than you want to work with in Hive View.
- Click one or more databases in the database and tables pane (*not* in the **DATABASE** field) to designate active databases. All queries in the current tab are then run against the active database or databases.
- The database and tables pane displays all the tables of the active databases in a single view.
- The Query Editor field and an active **RESULTS** tab can be toggled between compact and full-screen mode. The screenshot below shows the compact mode. Full-screen mode enlarges the query editor field, which can help you author large, complex SQL queries or browse large query results.
- Use multiple **Worksheet** tabs to work on and analyze queries in parallel.

**Query Editor and Database and Table Pane**

Use the **Browse** button in the Query Editor to find and select databases. After you select databases and make them active in the view, they appear in the database and table pane.

**Figure 6.3. Query Editor**

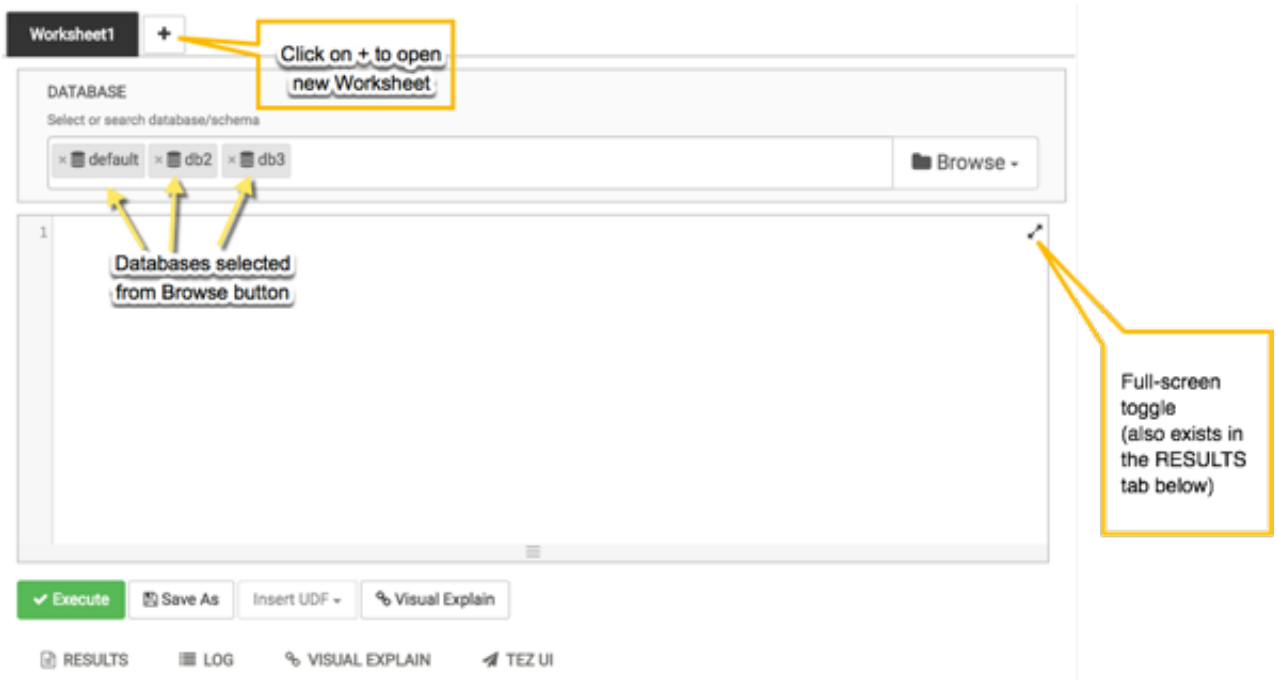
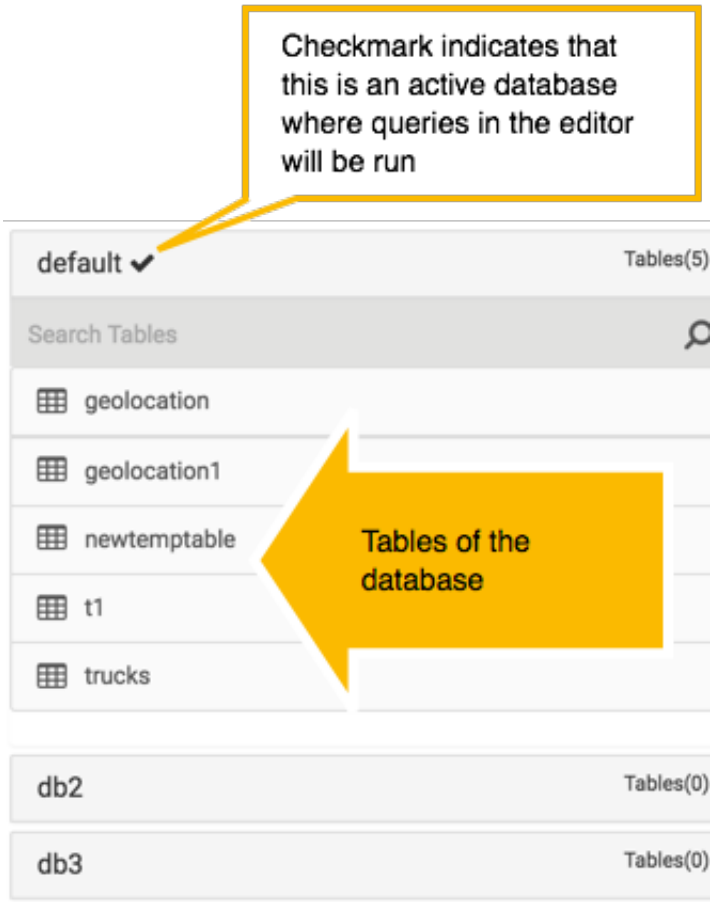


**Figure 6.4. Database and Table Pane**




*How to Use the Query Editor*

- Enter queries in the blank field. To run a specific query, highlight it, and click **Execute**. All queries contained in a Worksheet tab execute sequentially, and they run in the same session.
- Click the **Results** tab to view the results after a query is exeucuted.
- Similarly, click the **Logs** tab to view the log from an executed query.
- When the first query is executed in a Worksheet, a Tez session is started. Click the **Tez UI** tab to see details about the DAG execution.
- 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 on the + symbol 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.

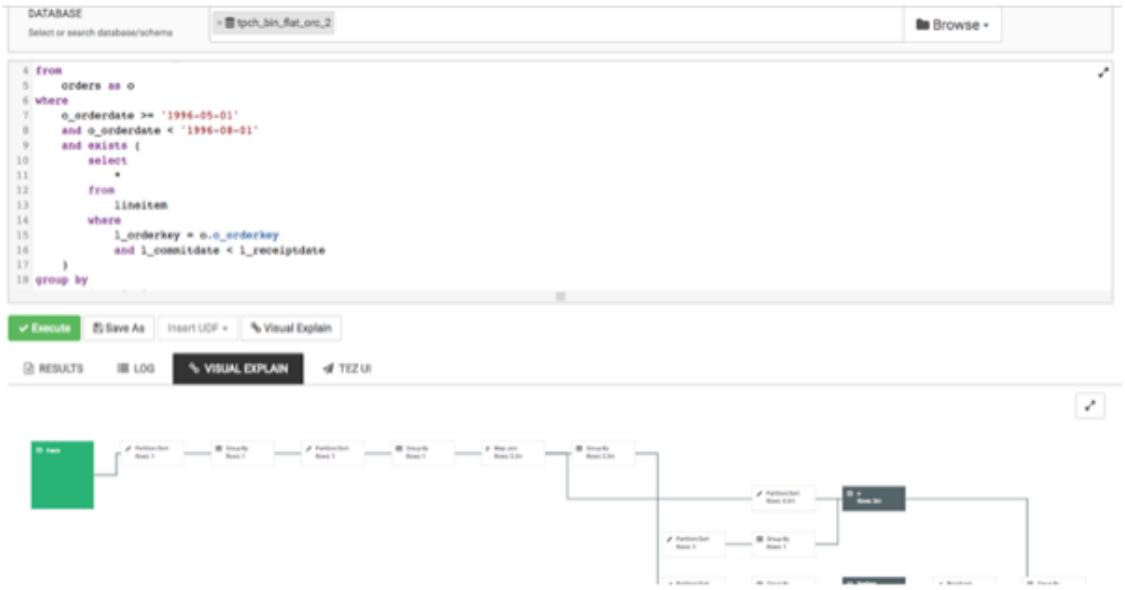
- Click the double arrow icon in the upper right corner of the Query Editor to expand the Worksheet area and cover Database Explorer. Click the icon again to collapse the Worksheet and make Database Explorer available again.
- Click the icon at the bottom of the Worksheet window and drag it down to expand the authoring space.

## 6.1.1. Visual Explain Plan

Hive View 2.0 includes a simple, graphical visualization of the Hive execution plan that focuses on key information that can help you spot expensive operations in your query.

Clicking the **Visual Explain** tab launches a visual explain plan.

**Figure 6.5. SQL in Query Editor with Resulting Visual Explan Plan**




The visual explain plan illustrates the execution of the query, regardless of whether you run the query or not. So you can use the Visual Explain feature truly as a planning tool.

### Important

The workflow sequence of the visual explain plan is plotted beginning on the right side and proceeding to the left.

The nodes of the explain plan contain kernels of information that help you forecast or troubleshoot execution problems. Key information of the plan includes:

- *Execution "cost"*: The number of processed rows for each node of the visual explain plan is displayed, which gauges the resource demands of the various runtime processes of an executing query.
- *Labeling of runtime processes for easy comparison*: Each node is identified by type of operation or process. For example, you can see which nodes represent "cheap" Map Join activity and compare them to "expensive" Partition or Sort operations.

### Tip

If your query performs Partition or Sort operations on a large number of records, the query runtime is relatively long. In this case, consider tuning or rewriting the query.

Click on a node to view detailed information about SQL operators:

**Figure 6.6. Details of a Map Join Node**


**Debugging Hive Query Execution Using the Tez UI**

Query execution can be debugged using the embedded Tez UI. To access the Tez UI, click the **TEZ UI** tab at the bottom of the Query Editor window.

### Important

Clicking the Tez UI in Hive View instantiates an Ambari Tez View for the current query. Therefore, for more information about how to use the Tez UI, see Using Tez View.

# 6.2. Jobs Tab

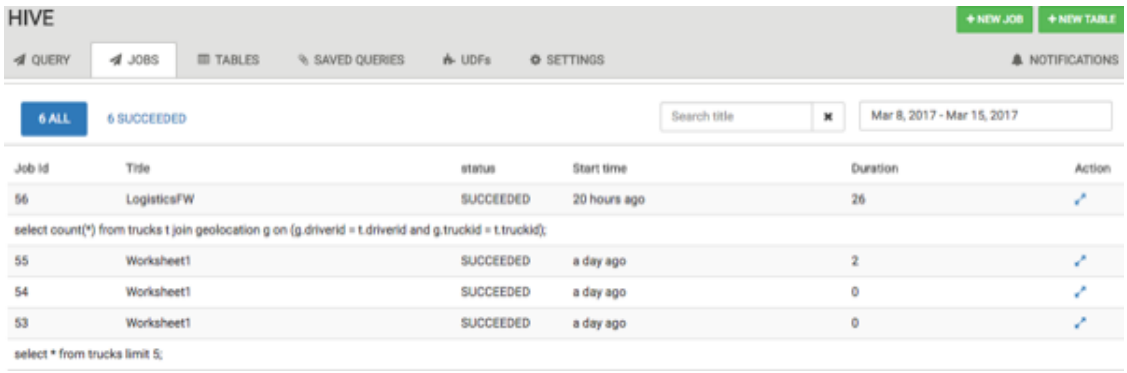
You can view the history of all queries that the current user executed in the Hive View instance on the **Jobs** tab. In addition, activity in the Hive View that does not execute queries is logged, such as generating a visual explain plan for a query that is not run.

The **History** tab of Hive View in Ambari 2.4.2 and earlier versions was a predecessor to the **Jobs** tab. The functionality of the **Jobs** tab is similar to the **History** tab. The main differences are:

- The **Stop execution** button that was on the **History** tab is not on the **Jobs** tab. Instead, you can stop an executing query by clicking the **Stop** button on the **Query** tab.

- The **History** tab displayed *all* queries that were run by the current user on the databases selected in Hive View, even if the user executed the queries outside Hive View (such as in a JDBC or ODBC client). The **Jobs** tab only shows activity of the current user in Hive View.

### Figure 6.7. Jobs Tab of Hive View 2.0



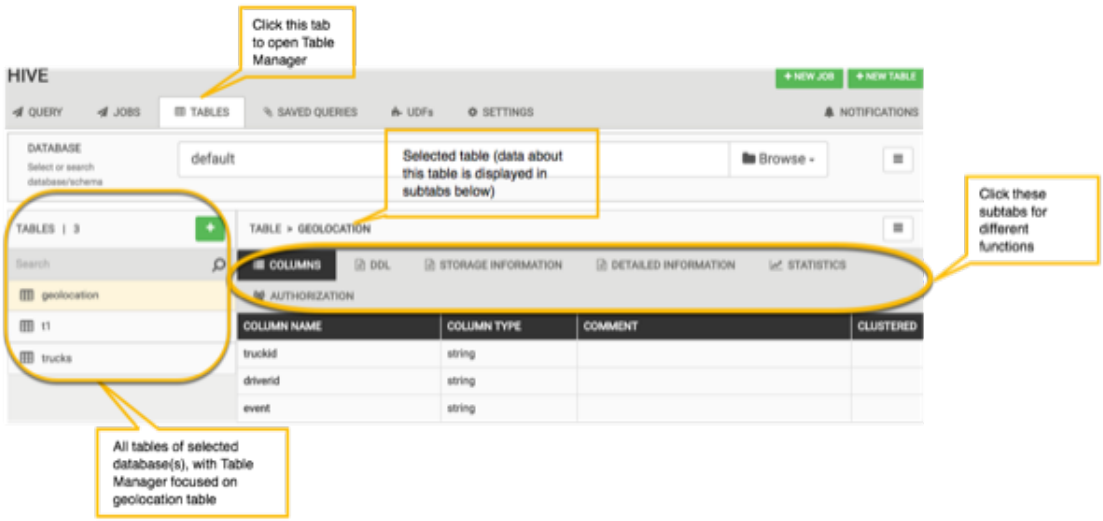

Note the following details about the way the UI operates:

- The **Title** column displays query names. If no name was assigned to the query, the worksheet number appears.
- Click the icon in the **Action** column to toggle between compact and expanded view of each job. Expanding a job on this tab shows the SQL statement for the action. For example, in the Jobs Tab screenshot above, Job 56 and Job 53 are expanded.

## 6.3. Tables Tab

Click the **Tables** tab to access the Table Manager. The Table Manager is one central place to view, create, delete, and manage tables of whichever databases that you select after clicking the **Browse** button. By default, the Table Manager opens with the **Columns** subtab enabled.

### Figure 6.8. Table Manager



### Gathering Information about a Table

In the **DDL**, **Storage Information**, and **Detailed Information** subtabs, you can view properties, storage, and other information of the particular table selected in the left-side pane.

**Figure 6.9. Example of Information in the DDL Subtab**

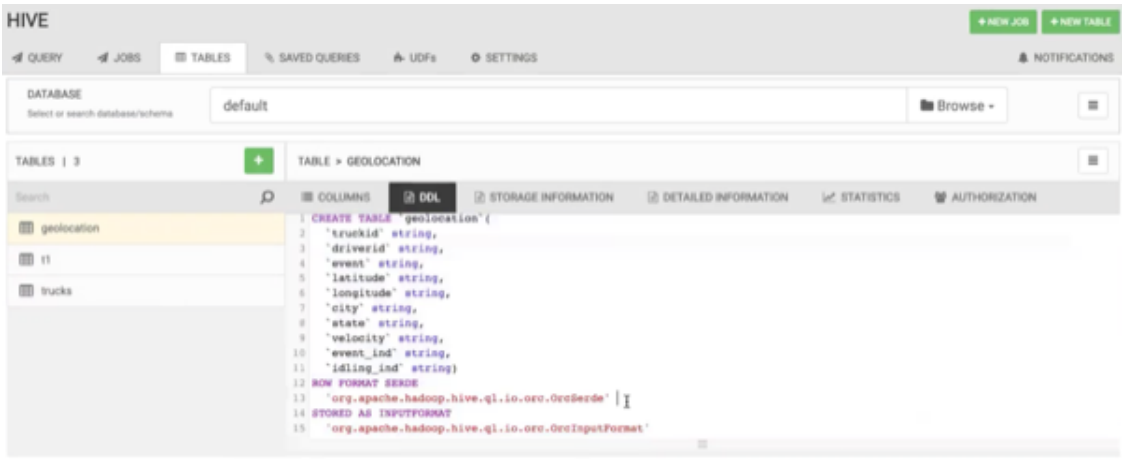



**Figure 6.10. Example of Storage Information Subtab**

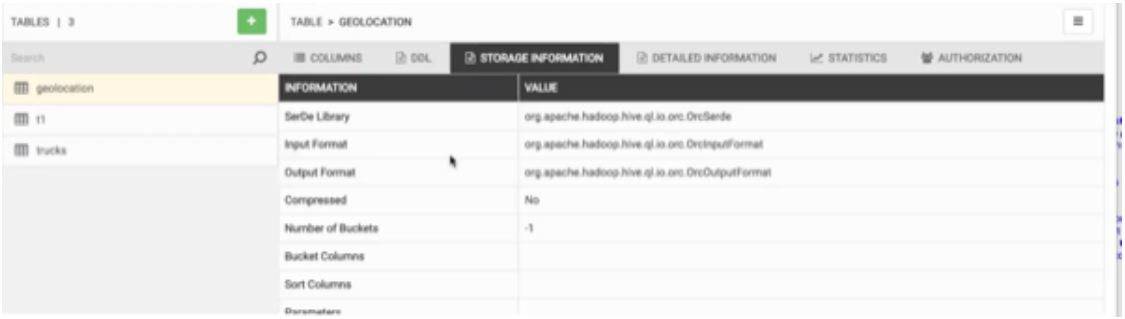
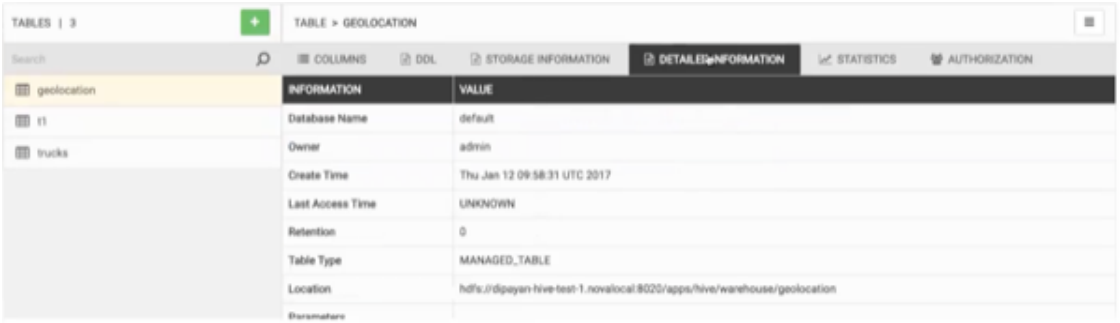



**Figure 6.11. Example of Detailed Information Subtab**



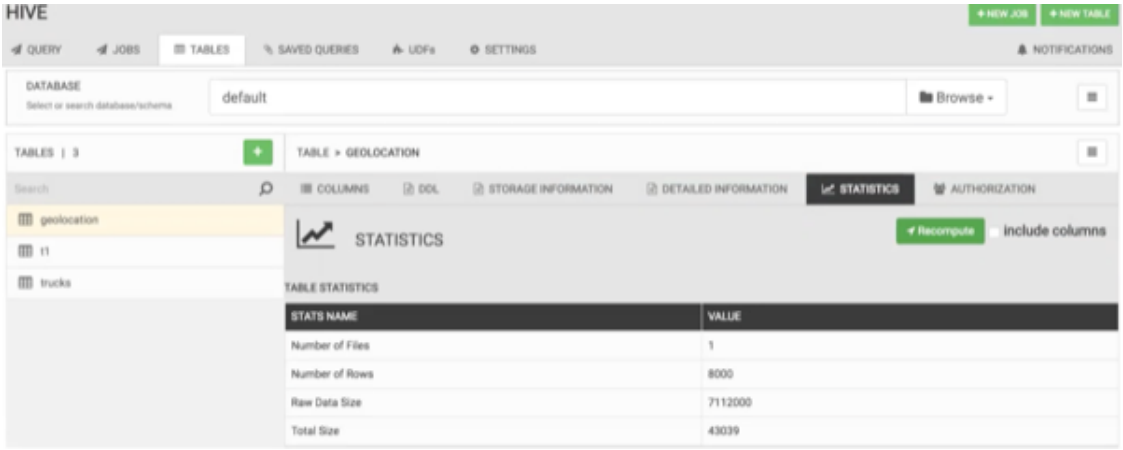

### Running and Viewing Statistics of a Table

To assess how big a table is and to estimate the execution duration of queries that run on the table, you might find the **Statistics** subtab to be helpful. The subtab also calculates statistics on each column of the table if you want to gather this information.

If statistics do not appear on this subtab, click the **Compute** button. The button changes to **Recompute** after you first calculate statistics. You might click **Recompute** if you want to change whether or not column information is included or if you change the table in some way after the initial calculation of statistics.

In the following screenshot, **include columns** must be selected and the **Recompute** button clicked to display column statistics.

**Figure 6.12. Example of Statistics Subtab**




## 6.3.1. Creating Tables

You can add a table to a database by either clicking the + symbol or the **NEW TABLE** button.

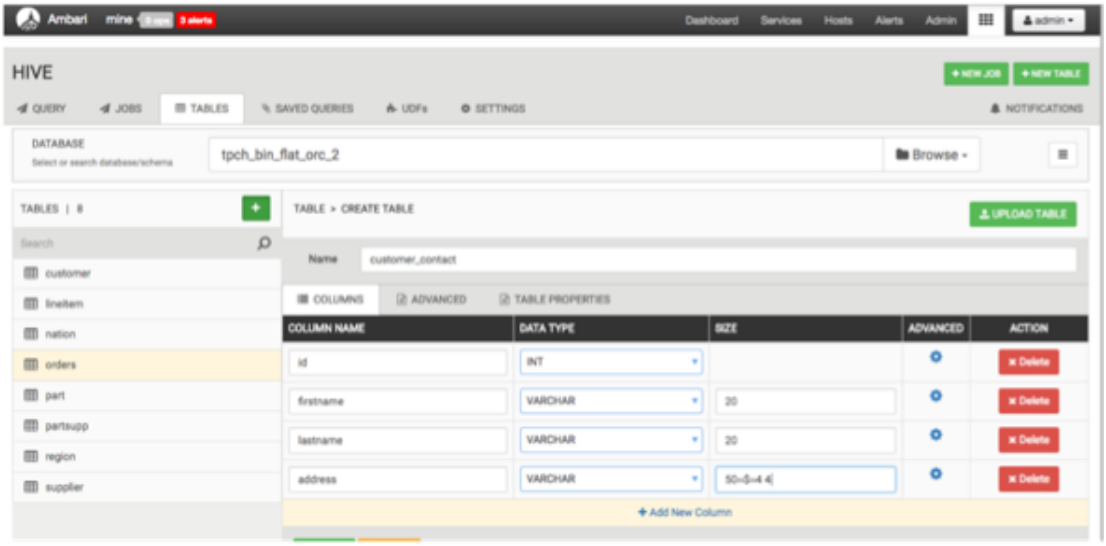
If you create a table, the Table Manager lets you build the table column by column and helps you use advanced features like ACID, partitioning, and bucketing. The form that appears is dynamic. For example, if you designate the DATA TYPE for a column as **DECIMAL**, the SIZE fields show two editable fields that are appropriate for a decimal data type (Precision and Scale).

Depending on the column data type that you choose, click the configuration wheel in the ADVANCED column of the form to select whether the column is Partitioned or Clustered. If you select Partitioned, the Table Manager adds a **Partitions** tab in the Table Manager. The **Partitions** tab helps provide an overview of all partitioned columns in the table.

### Tip

Notice that the Create Table mode also includes the **ADVANCED** and **TABLE PROPERTIES** subtabs, which enable you to further customize the new table. The **ADVANCED** subtab lets you make a table transactional (that is, enable ACID properties), change the default location of the Hive table in HDFS, change the file format from the default ORC format. If you select the TEXTFILE format, you can also configure the row format (for example, separator values, null values, escape values).

**Figure 6.13. Example of Creating a Table Form**




## 6.3.2. Uploading Tables

To access the Upload Table functionality, click the + symbol or the **NEW TABLE** button.

### Tip

With Hive View 2.0, you can change more properties of an existing table while performing the Upload Table operation. Earlier versions of Hive View only let you change column names and data types while uploading a table.

In the Upload Table window, you can upload files which contain the rows of an 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?** option 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 or 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 treated as 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 Hive View:

**Steps**

1. Select the input file format file type by specifying CSV, XML, or JSON.

2. If the File Type is CSV, you can select the **Field Delimiter**, the **Escape Character**, and the **Quote Character** values in the drop-down menus. Also, you can click the **Is first row header?** box if you want to enable this feature.

a. 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.

b. 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.

3. Expand the Select File Source section to pick the table to upload. Click **Upload from HDFS** or **Upload from Local**.

4. If you clicked **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. (Optional) Change the suggested table name, column names, column data types, and many other table DDL and properties in the **Columns**, **Advanced**, and **Table Properties** subtabs.

6. Click the **Create** table. The actual table and temporary table (stored as TEXTFILE) are created. After this the data rows from the file are inserted into the temporary table followed by insertion from temporary table to actual table.

7. 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.

# 6.4. 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:

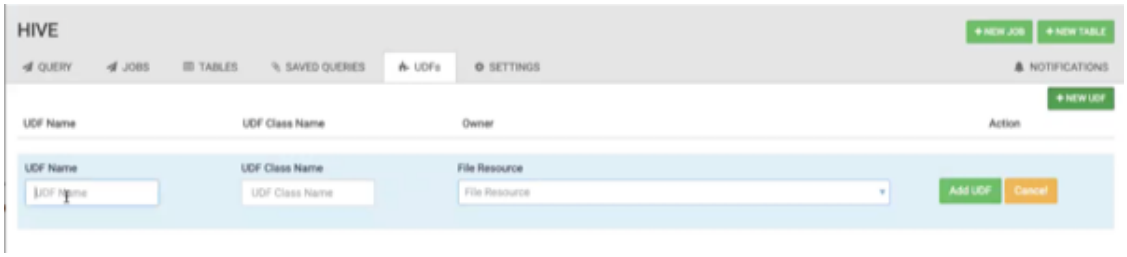
**Figure 6.14. Saved Queries Tab**


# 6.5. UDFs Tab

User-defined functions (UDFs) can be added to queries by pointing to a JAR file on HDFS and indicating the Java classpath, 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:

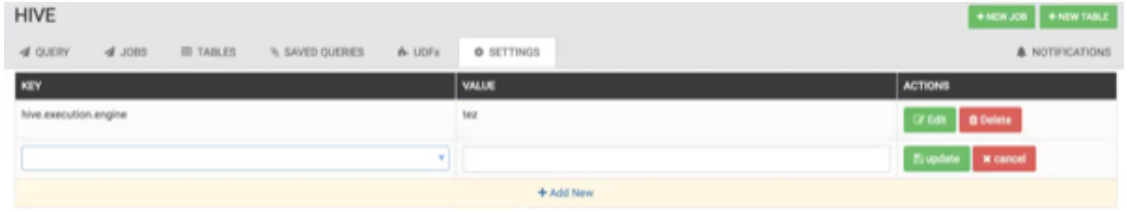
**Figure 6.15. UDF Tab**




# 6.6. Settings Tab

Use the **Settings** tab to append settings to queries that you execute in Hive View.

**Figure 6.16. Settings Tab with Example Key and Value for One Property**



# 7. Using 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. **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.

Use Pig View to:

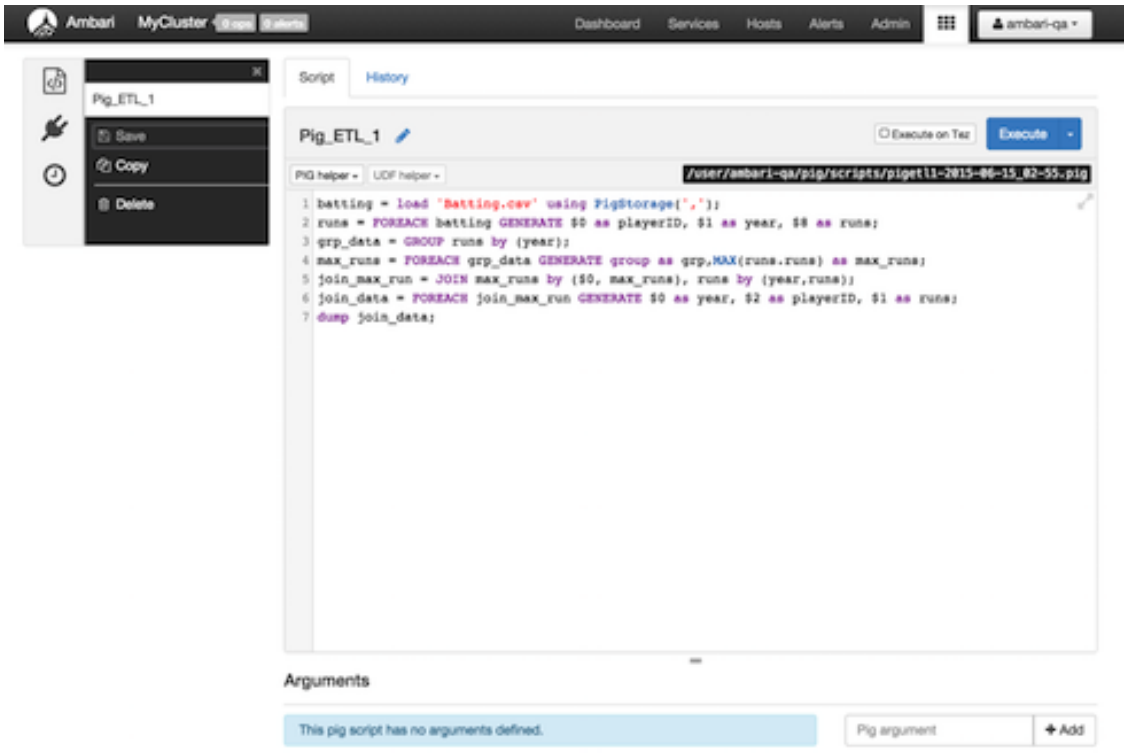
- Write Pig scripts
- Execute Pig scripts
- Add user-defined functions (UDFs) to Pig scripts
- View the history of all Pig scripts run by the current user

## 7.1. 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:

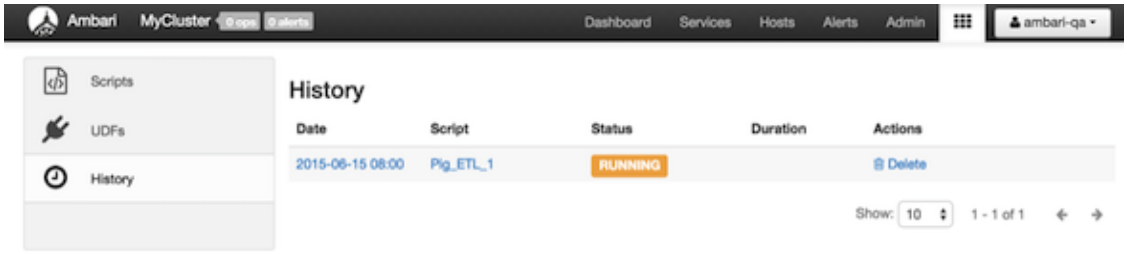
### Figure 7.1. Pig Script Running in Pig View



# 7.2. 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 7.2. Pig View Script History Tab




# 7.3. 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 7.3. Pig View UDFs Tab

# 8. Using Slider View

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

### Important

This view has been marked deprecated.

# 9. Using SmartSense View

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.

Use the SmartSense View to perform the following tasks:

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

# 10. Using 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.

Use 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.

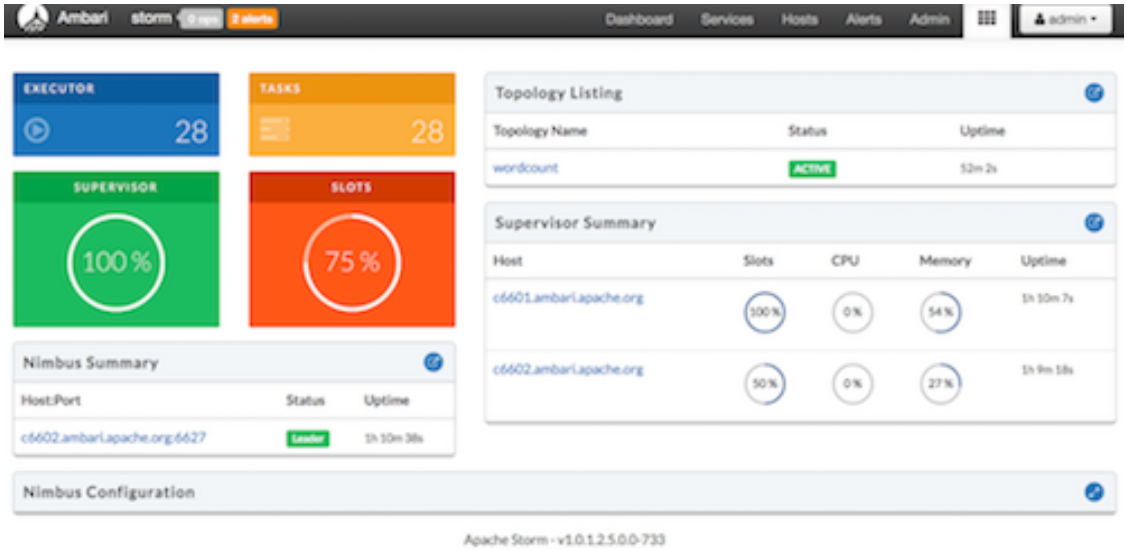
**Important**

This view has been marked deprecated.

## 10.1. Monitoring Storm Cluster Status: the Cluster Summary Page

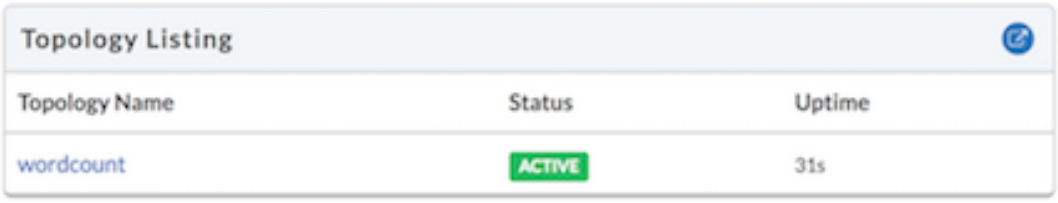
The landing page for Storm view shows current cluster status and nimbus configuration.

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:

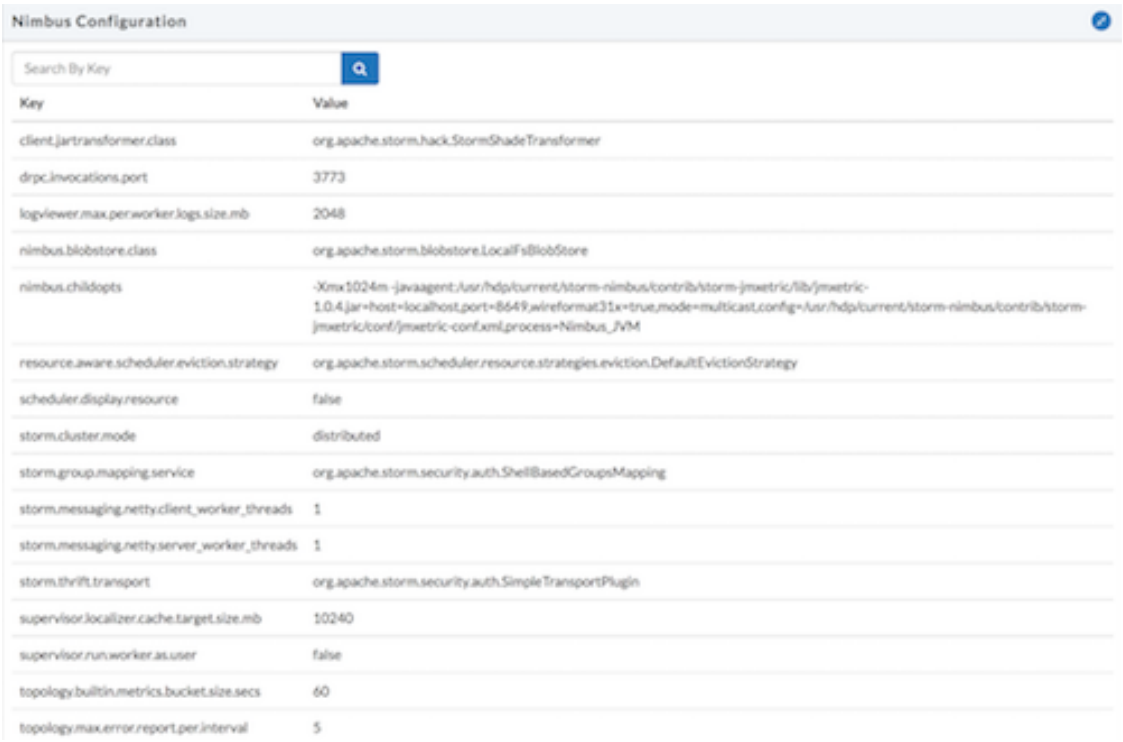


The lower left section of the summary page shows resource utilization of supervisors:


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



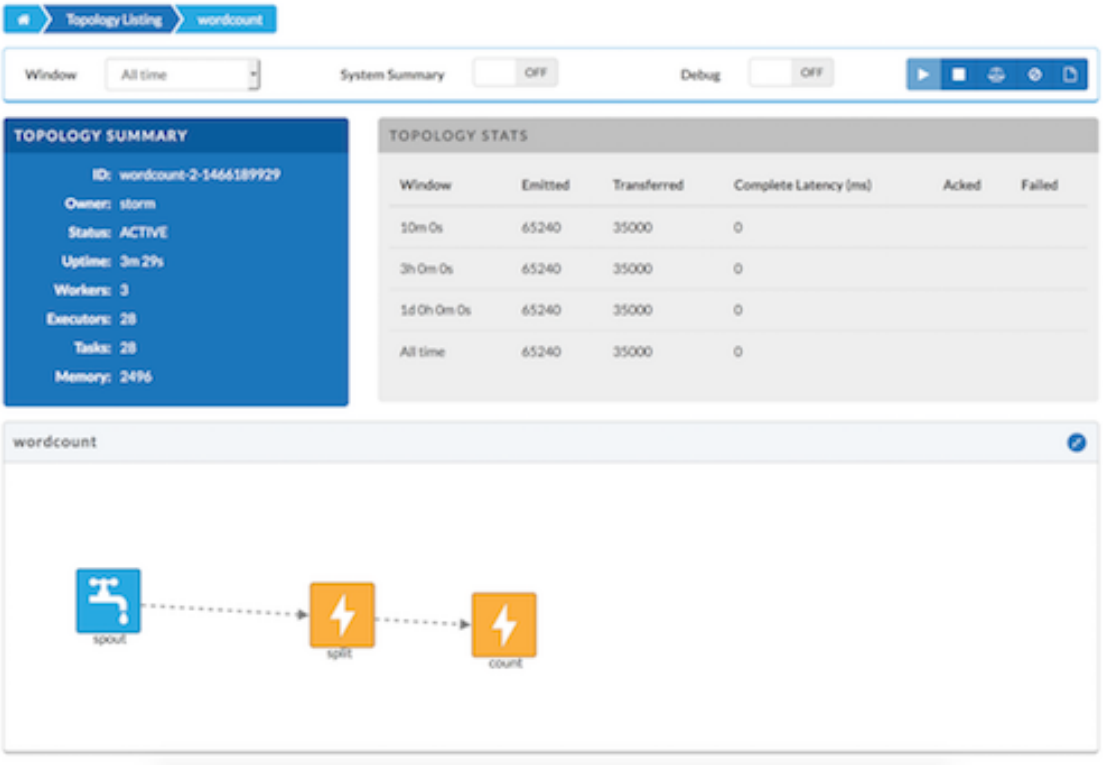

Click on the "Nimbus Configuration" section to list Storm configuration settings:



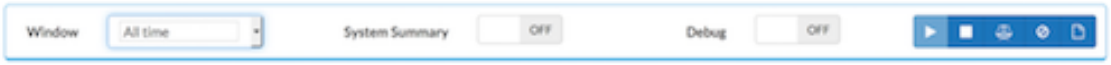
# 10.2. Monitoring Topology Status: the Topology Summary Page

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."

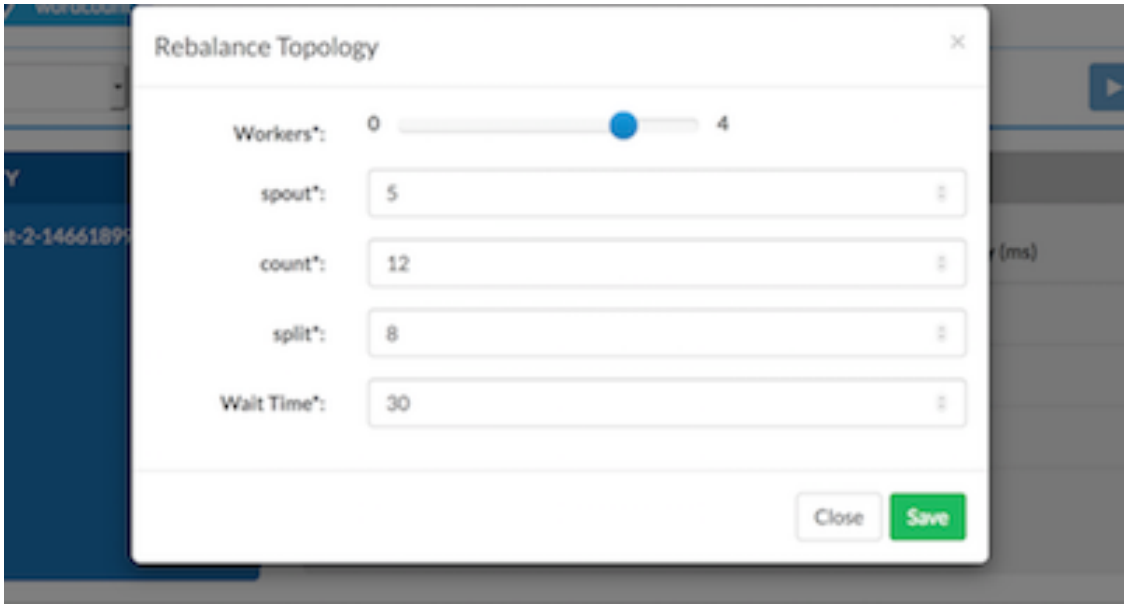



On the right side above panel, there are several topology actions buttons. These buttons allow you to perform several actions: Activate (highlights when topology status is deactivated), Deactivate, Rebalance, Kill, and Changing log level.




**Rebalancing a Topology**

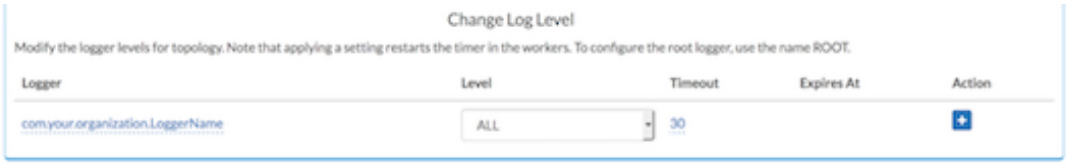
To adjust the number of workers for the topology and the parallelism of each component in the topology, use the rebalance button.




**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.

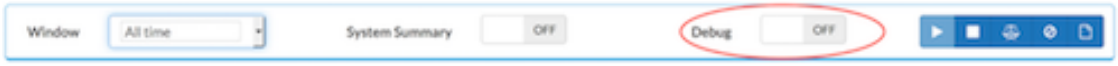


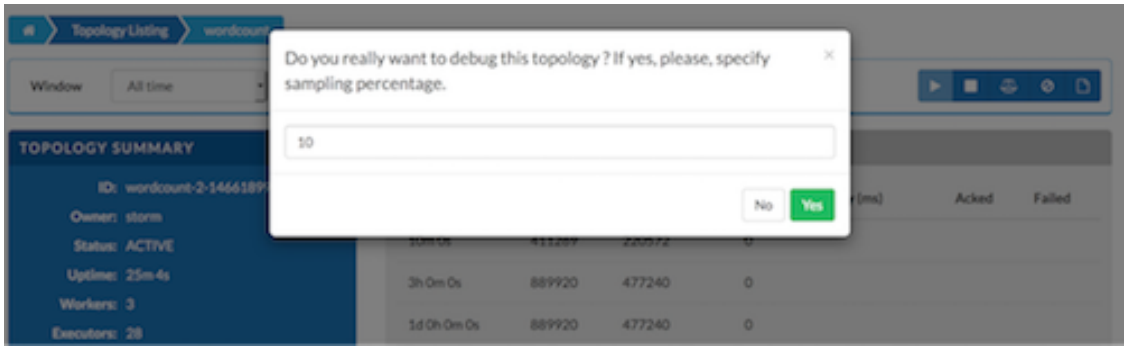

For example, if you would like to see debug logs in the count bolt of the sample word count topology supplied with Storm, add the classname as `org.apache.storm.starter.WordCount`.

**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":

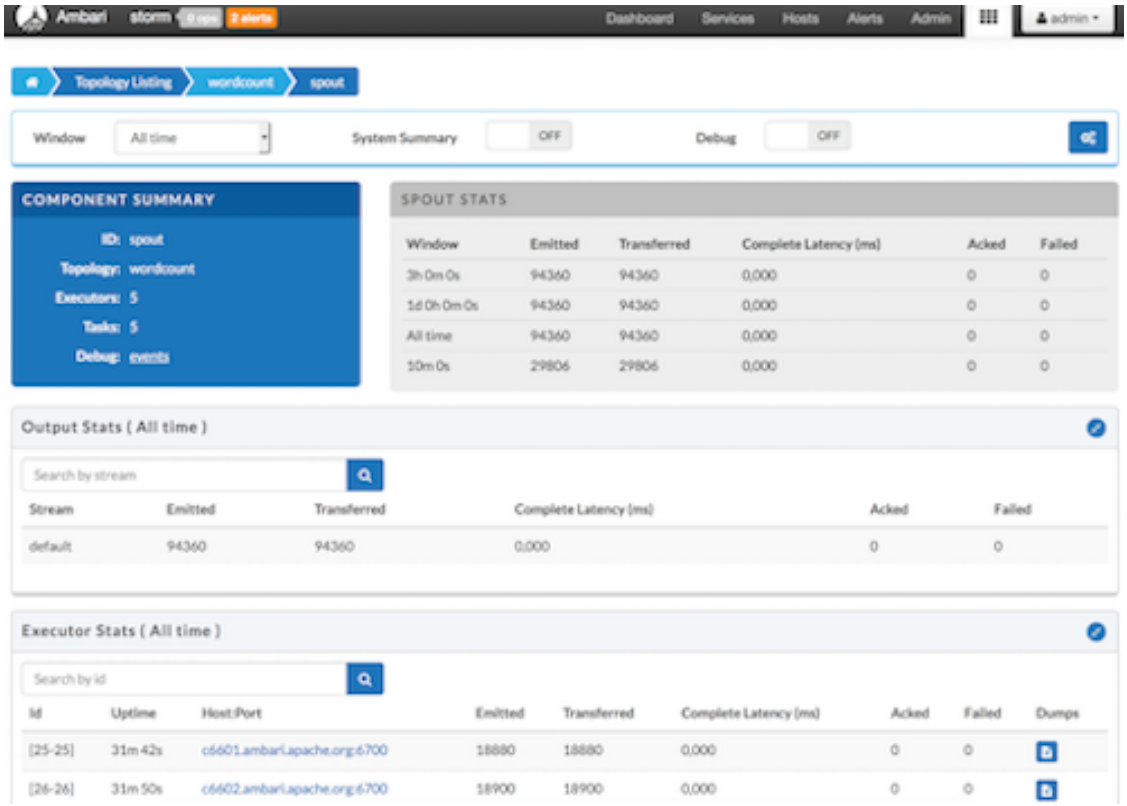




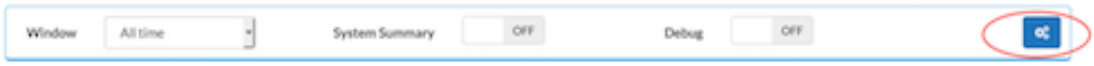

The event logger will sample the given percentage of incoming tuples and write them to the log for users to see the incoming tuples at each stage of topology. We recommend that you not set this to a higher percentage, because it can fill up the logs on disk very quickly.

# 10.3. Looking Up Configuration Values: the Component Summary Page

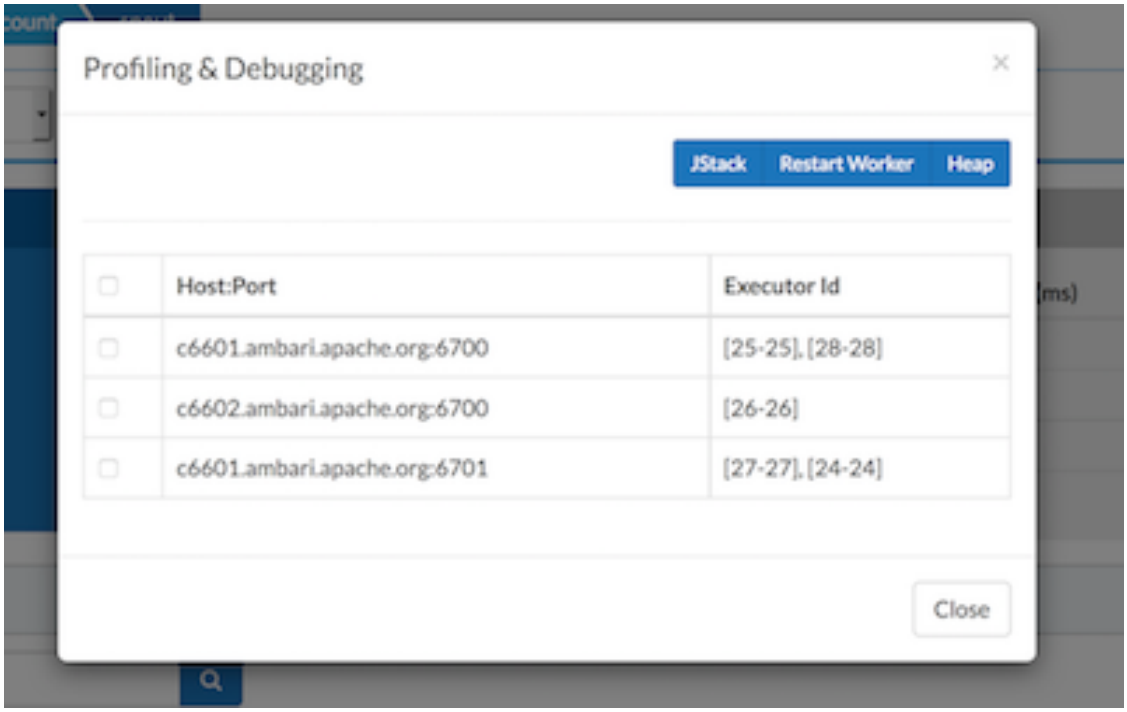
On the Component Summary page, you can drill down to a individual component in a topology to see relevant stats for the component and access debug logs and jstack outputs.




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.



# 11. Using Tez View

Tez is a 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 Tez View to track and debug the execution of that job.

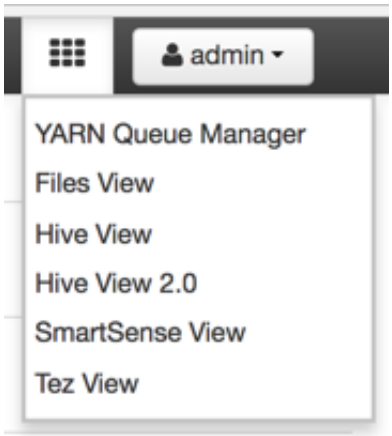
Tez provides a framework that enables human-interactive response times with Apache Hive queries and Apache Pig data transformations. Tez View helps you find specific Hive queries and their performance metrics, even in environments where there are hundreds or more queries running daily, when you need to troubleshoot or tune data analytic applications. Both sortable metrics and graphic visualizations enable you to understand and debug submitted Tez jobs, such as Hive queries or Pig scripts, that are executed using the Tez execution engine.

**To open the Tez View:**

1. Open Ambari.

2. Click the views tile in the upper right corner of the window:

**Figure 11.1. Views Menu of Ambari**




3. Select **Tez View**.

The following sections describe using Tez View to manage Hive and Pig tasks:

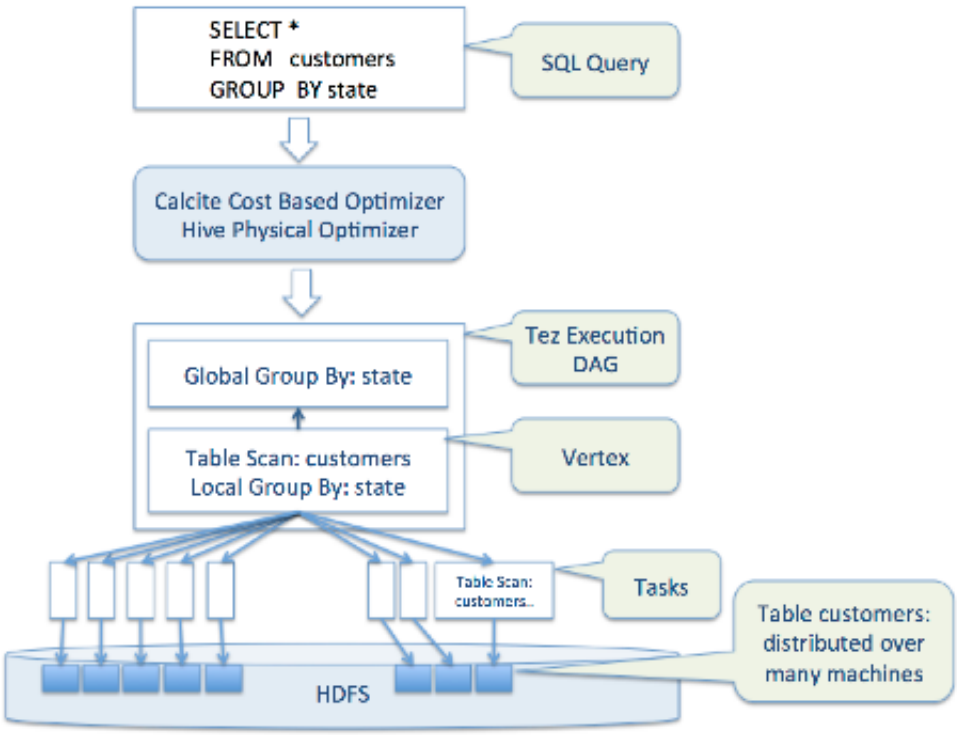
- Understanding Directed Acyclic Graphs (DAGs), Vertices, and Tasks [116]
- Searching and Identifying Hive Queries [116]
- Identifying the Tez DAG for Your Job [120]
- Understanding How Your Tez Job Is Executed [122]
- Identifying Causes of Failed Jobs [122]

# 11.1. 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 11.2. SQL Query Execution in Hive**




# 11.2. Searching and Identifying Hive Queries

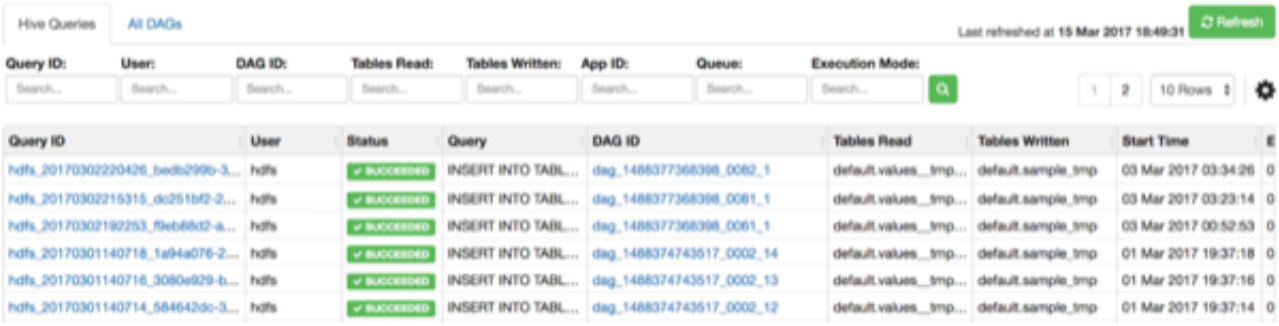
The landing page of the Tez View has a tabbed layout, with the **Hive Queries** tab as the default tab. While displaying all the query details in a tabular format with pagination support, the UI also provides options to search based on various parameters.

The search criteria of the **Hive Queries** tab are:

- Query ID
- User
- DAG ID
- Tables Read
- Tables Written
- App ID
- Queue
- Execution Mode

Search returns hits when there are exact matches with the criteria that is entered on the top of the tab. Therefore, you must enter each search string as a complete value in the search criteria fields. You can enter multiple values (table names) in the **Tables Read** filter and **Tables Written** filters.

**Figure 11.3. Hive Queries Tab Showing Unfiltered Results**




## 11.2.1. Analyzing the Details of Hive Queries

Click the relevant link in the **Query ID** column for a query that you want to investigate. A window with three tabs containing information about the query is displayed.

**Details Tab**

The **Details** tab, the first of the three tabs, is displayed after clicking a **Query ID** link.

**Figure 11.4. Details for a Successful Query with Links to Application and DAG Windows**


### Total Timeline View

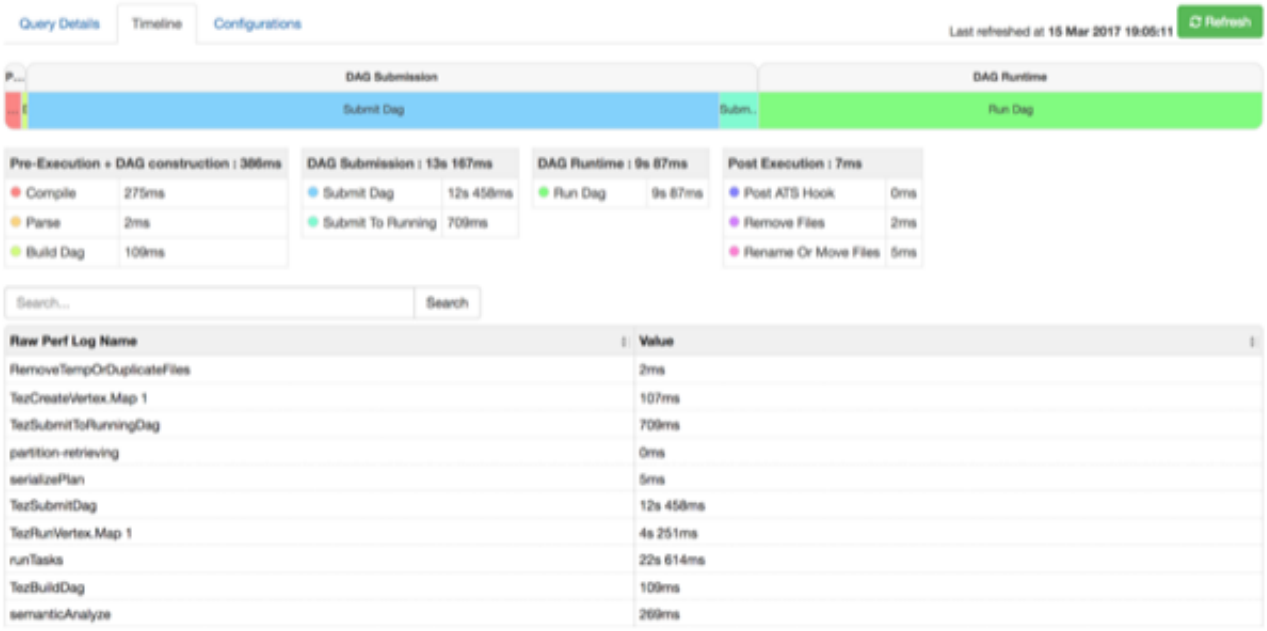
Click the **Timeline** tab to get a visual representation of Hive performance logs. The view represents the following pre-execution, runtime, and post-execution phases of a query:

- Pre-execution and DAG construction
- DAG submission
- DAG runtime
- Post-execution

Duration data about each phase are distilled into more granular metrics based on query execution logs.

A search-enabled table with raw performance log names and their major values is displayed under the timeline visualization.

**Figure 11.5. Total Timeline and Log Details of a Submitted Query**



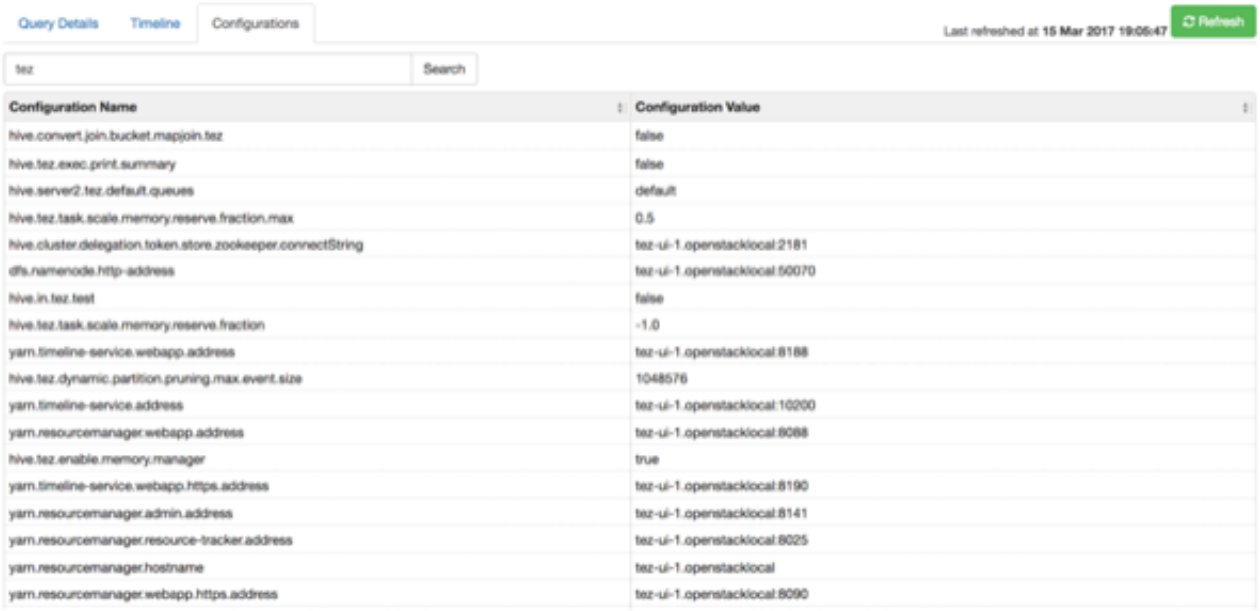

### Configurations Tab

Click the **Configurations** tab to see a list of configuration properties and settings that are used in the Hive query. You can use this tab to verify that configuration property values align with your expectations.

#### Tip

By default, only configuration property names that contain the substring `tez` are listed. Use the **Search** field to change the search criteria.

**Figure 11.6. Configurations Tab**

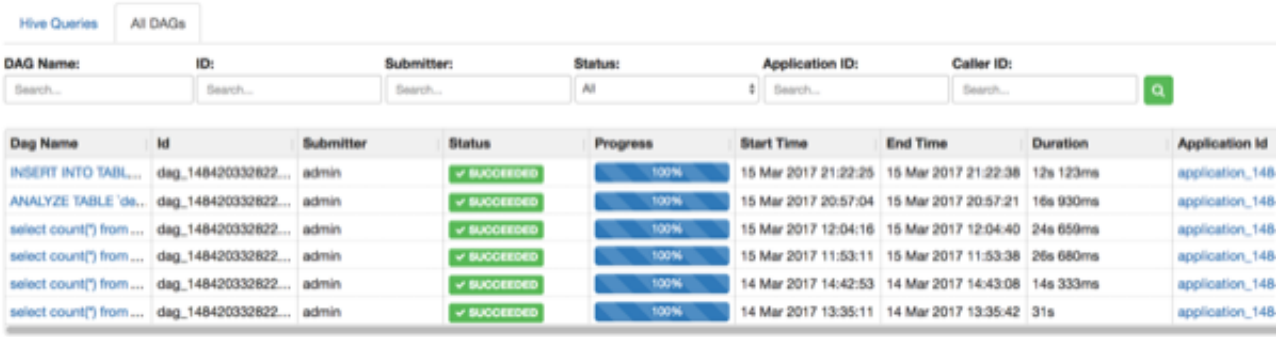



# 11.3. Identifying the Tez DAG for Your Job

Click **All DAGs** to view the a list of jobs sorted by time, listing the latest jobs first. You can search for a job using the following search criteria fields:

- **DAG Name**
- **Id** (DAG identifier)
- **Submitter** (user who submitted the job)
- **Status** (job status)
- **Application ID** (application identifier)
- **Caller ID**

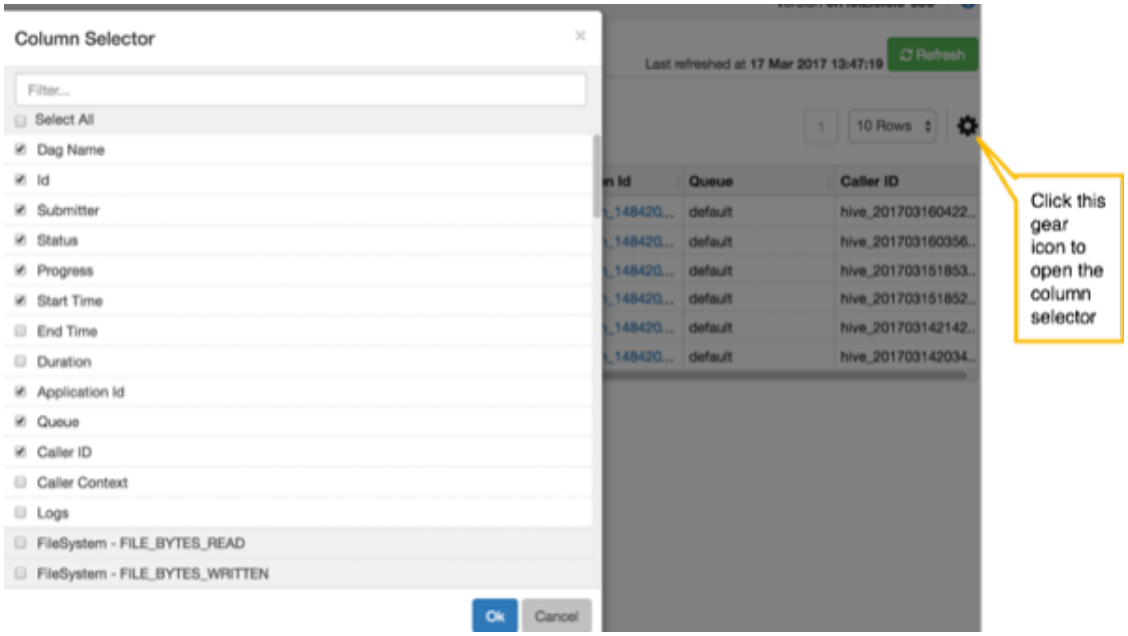
**Figure 11.7. All DAGs View (Truncated Screenshot)**



### 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 Tez View:

**Figure 11.8. Tez View Column Selector Dialog Box**




### Tip

To search for columns, use the search filter at the top of the Column Selector dialog box. Click **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 Tez View:

**Table 11.1. 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. |

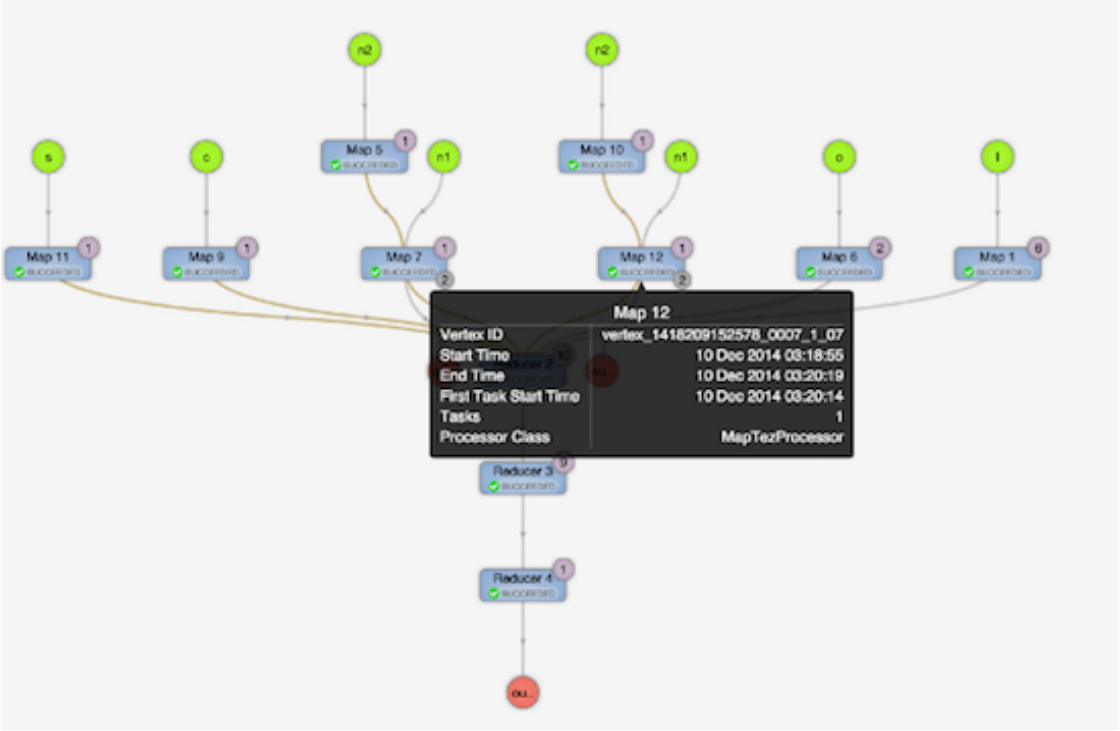
# 11.4. Understanding How Your Tez Job Is Executed

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 11.9. View Tab in Tez View




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

# 11.5. Identifying Causes of Failed Jobs

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

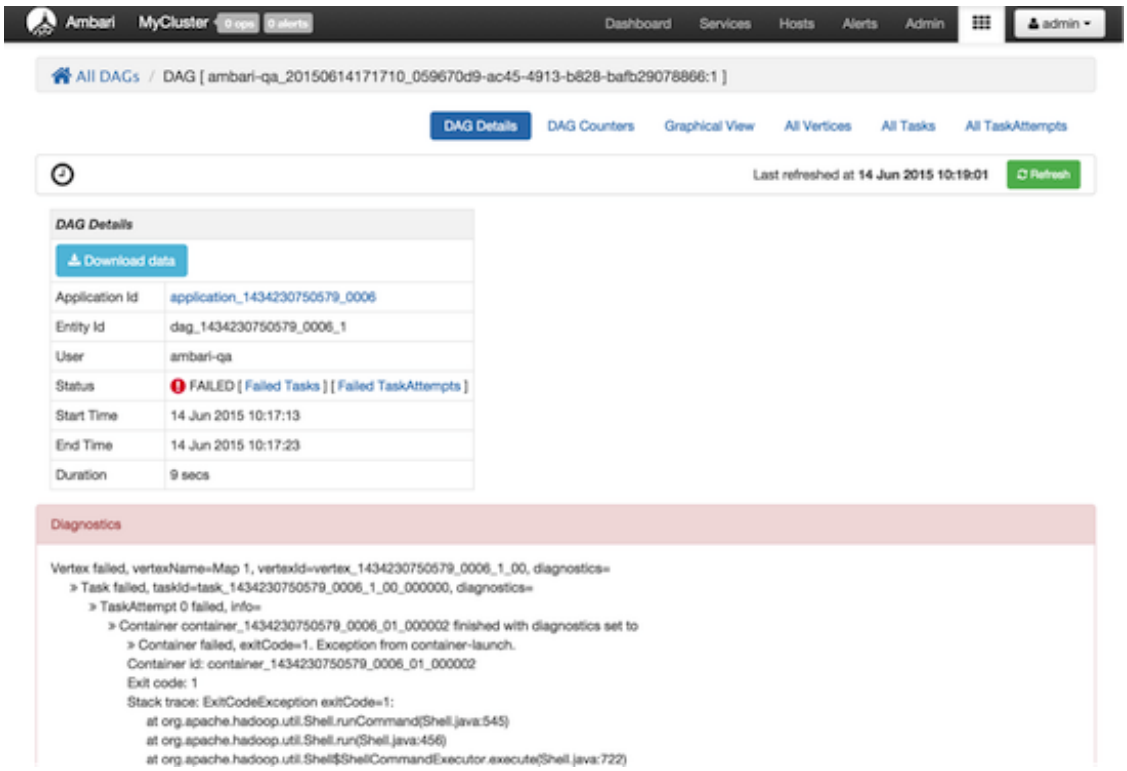
- Identify why the task failed
- Capture the reason for task failure

The DAG Details tab

When a Tez task fails, the **DAG Details**tab explains the failure. You can download the data of the DAG to further examine the cause of failure. Starting with Tez View in Ambari 2.5.0, the downloader has the following features:

- A progress bar during the download process is displayed and includes messaging about the download status
- If the data download is not completely successful, some information rather than a failure message without any DAG details is provided. The available data is stored as JSON file in the zip archive.
- If the downloader fails to completely capture any data, the downloader re-attempts to download data. The maximum number of re-attempted downloads is 3. The main reason for this feature is to address the scenario where ATS is briefly offline.

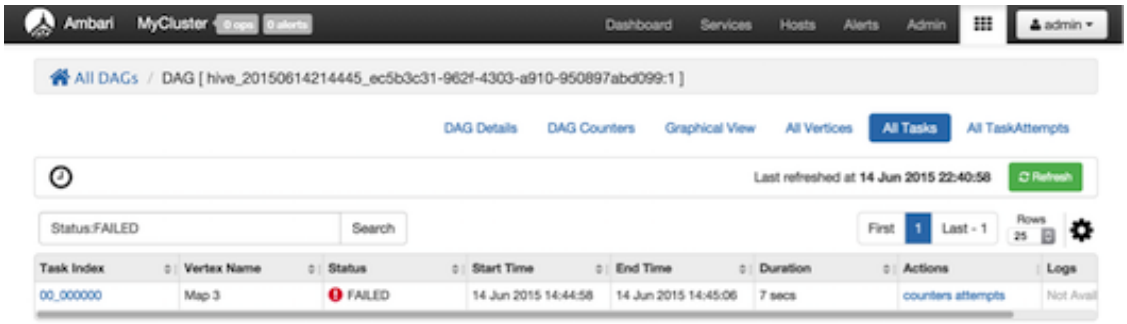
**Figure 11.10. DAG Details Window**




# 11.6. 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 11.11. Tez View All Tasks Tab**



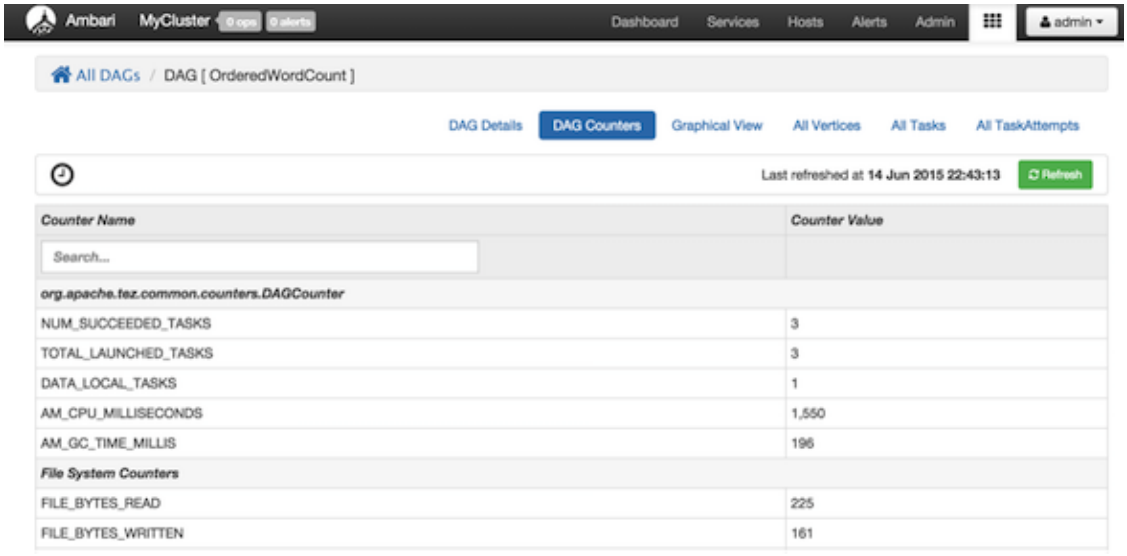

# 11.7. Using Counters to Identify the Cause of Slow-Performing Jobs

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. As of Tez View in Ambari 2.5.0, the following Hive LLAP counter information is displayed:

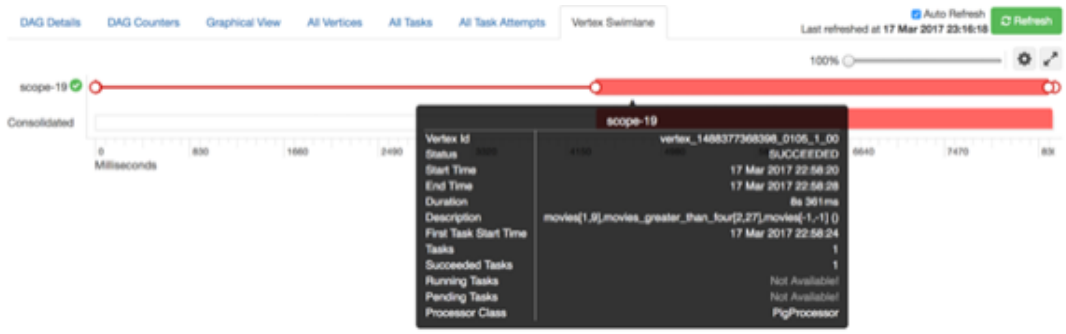
Group: `org.apache.hadoop.hive.llap.counters.LlapIOCounters`
Counters: `CACHE_HIT_BYTES, CACHE_MISS_BYTES, METADATA_CACHE_HIT, METADATA_CACHE_MISS`

**Figure 11.12. Tez View DAG-Level Counters Tab**



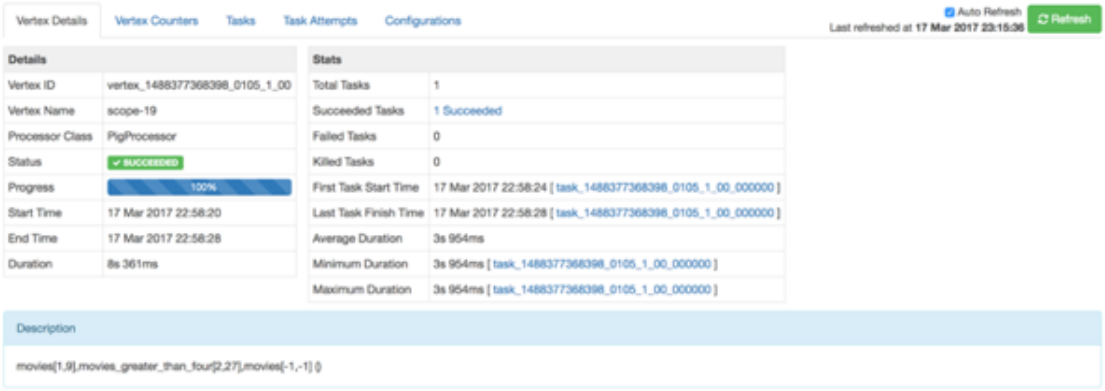

As of Tez View in Ambari 2.5.0, the **Vertex Swimlane** tab is also available. The information here is about vertex processor details for Hive and Pig.

### Figure 11.13. Tez View Vertex Swimlane Tab




If you want to view the information about the vertex processing in a different format, open a vertex to see the **Vertex Details** sutab. A **Description** pane at the bottom of the window is from the user payload of the respective vertex in the DAG plan.

### Figure 11.14. Tez View Vertex Details Subtab




### Figure 11.15. Tez View Vertex-Level Counters Tab

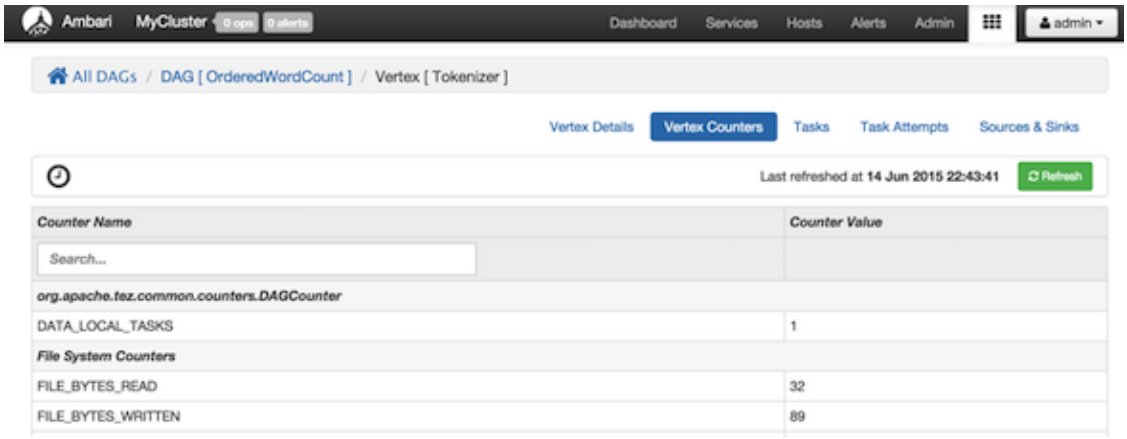
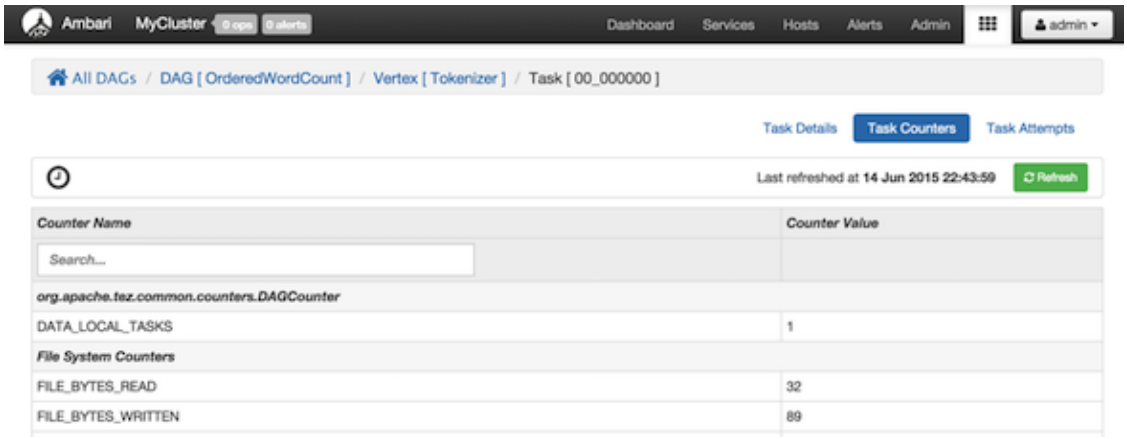


**Figure 11.16. Tez View Task-Level Counters Tab**




### 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.

# 12. Using Workflow Manager View

Ambari includes Workflow Manager View, which supports creating, scheduling, and monitoring jobs on a Hadoop cluster. Hadoop administrators can easily design and visualize workflows in the UI as flow graphs.

Workflow Manager is based on the Apache Oozie workflow engine that allows users to connect and automate the execution of big data processing tasks into a defined workflow. Workflow Manager integrates with the Hortonworks Data Platform (HDP) and supports Hadoop jobs for Hive, Sqoop, Pig, MapReduce, Spark, and more. In addition, it can be used to perform Java, Linux shell, distcp, SSH, email, and other operations.

You can access the Workflow Manager documentation in the Workflow Management guide on the Hortonworks documentation website.