Hortonworks Data Platform

Ranger User Guide

(Sep 30, 2015)

docs.hortonworks.com

Hortonworks Data Platform: Ranger User Guide

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 Tables

1. Security Introduction

Centralized security administration in a Hadoop environment has four aspects:

- Authentication

  Effected by Kerberos in native Apache Hadoop, and secured by the Apache Knox Gateway via the HTTP/REST API. (For further information, see the Apache Knox Gateway Manager Guide.)

- Authorization

  Fine-grained access control provides flexibility in defining policies...

  - on the folder and file level, via HDFS
  - on the database, table and column level, via Hive
  - on the table, column family and column level, via HBase

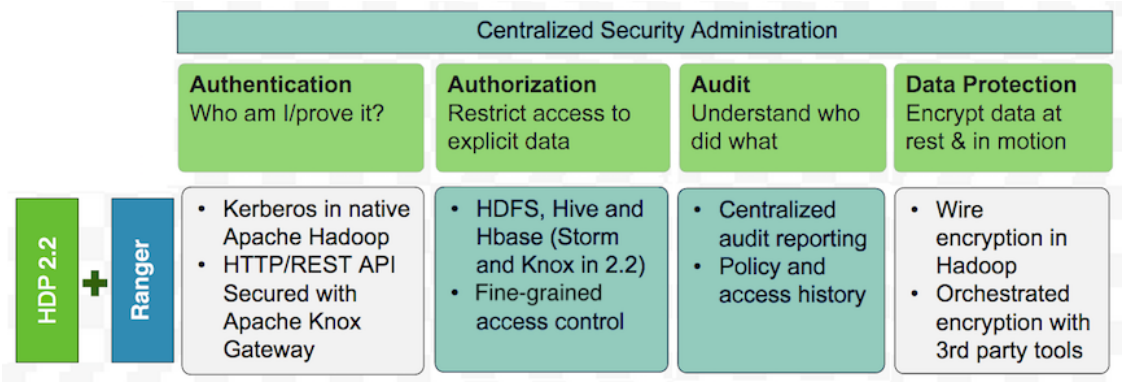
- Audit

  Controls access into the system via extensive user access auditing in HDFS, Hive and HBase at...

  - IP address
  - Resource/resource type
  - Timestamp
  - Access granted or denied

- Data Protection

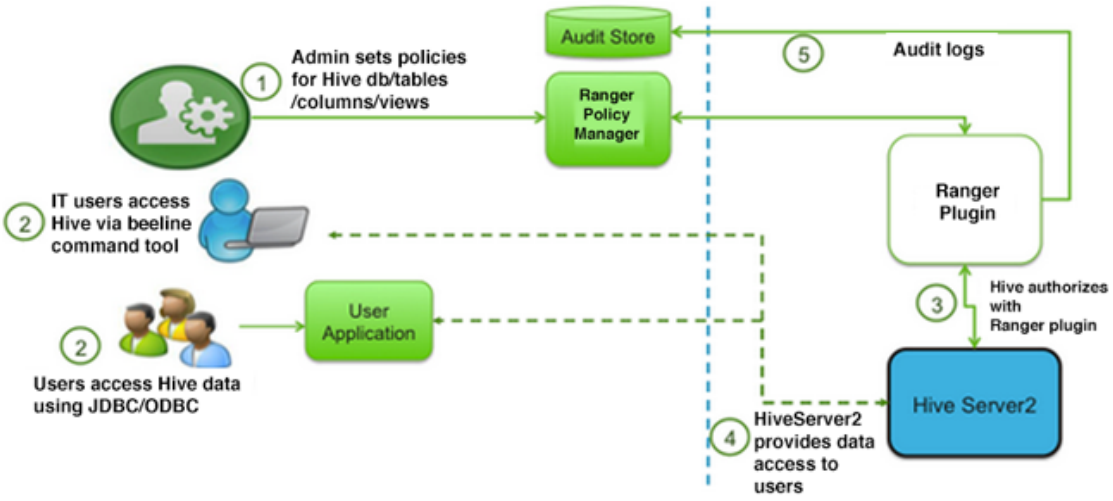
  Provided by wire encryption, volume encryption and (via HDFS TDE and Hortonworks partners) file/column encryption



Ranger Security Administration

1.1. Centralized Security Administration in Hadoop

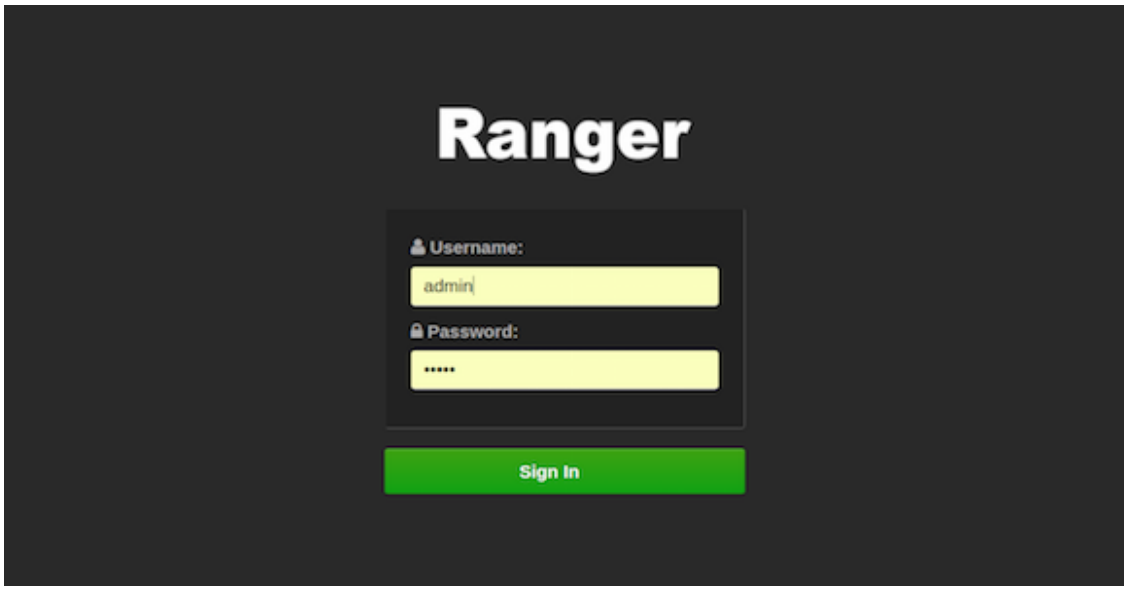
Central security administration is provided through the the Apache Ranger console, which delivers a 'single pane of glass' for the security administrator. The console ensures consistent security policy coverage across the entire Hadoop stack.



Ranger Central Security Administration

2. Opening and Closing the Console

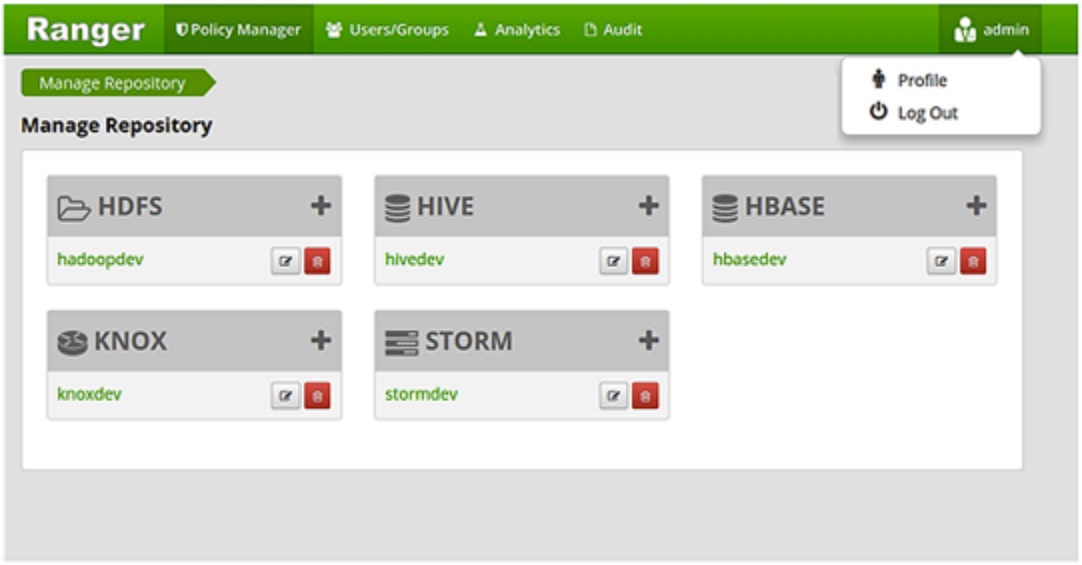
To invoke the Ranger Console, log in to the Ranger portal. To log in, use your username and password.



Ranger Login Console

Once you log in, your username is also displayed on the Ranger Console home page.

To log out of the Ranger Console, click your username, in the top right-hand corner of the screen. At the drop-down menu, click Logout.

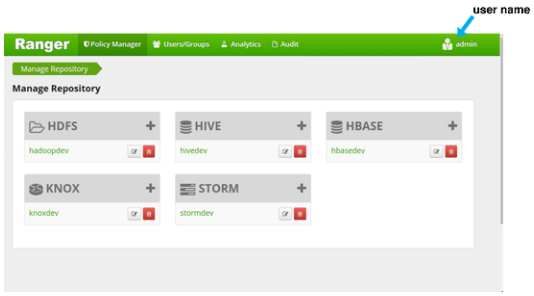


Ranger Console Home Page

3. Console Operations Summary

The Ranger console controls five types of functions:

- **The Repository Manager** - (visible upon user login, as shown above) adds and administers service repositories.
- **The Policy Manager tab** - creates and administers repository policies.
- **The Users/Groups tab** - assigns policy permissions to users and groups.
- **The Analytics tab** - is used to perform analytics on one or more HDFS, Hive, HBase, Knox or Storm policies.
- **The Audit tab** - monitors user activity at the resource level, and conditional auditing based on users, groups or time.

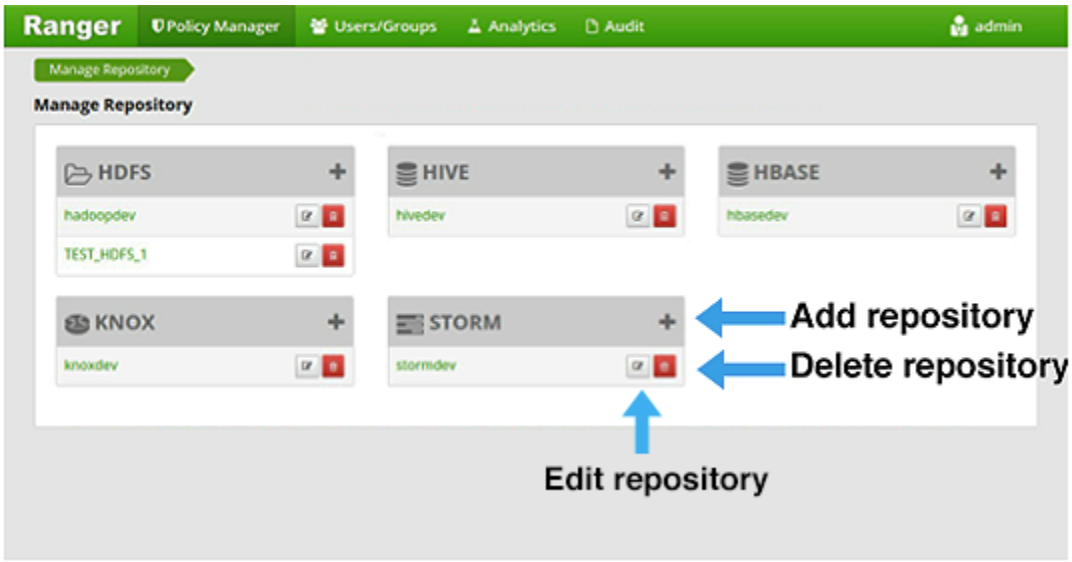


Ranger Repository Manager

4. Repository Manager

The Ranger Repository Manager is open by default in the Ranger Console. To return to the Repository Manager from any tab in the Ranger Console, go to the top left corner of the console and click `Ranger`.

- **To add a new repository** to the Policy Manager, click the + button in the appropriate box on the Ranger Policy Manager, then complete the repository screen. When you are finished filling in the screen, click the green Add button.
- **To edit a repository** in the Policy Manager, click the Edit icon to the right of the entry for that repository. The Policy Manager displays an expanded view of that repository, including a list of the policies it contains, their status, and the groups designated to administer those policies.
- **To delete a repository** from the Policy Manager, click the Delete icon to the right of the entry for that repository.



Ranger Policy Manager Console

This section describes how to configure repositories in:

- HDFS
- Hive
- HBase
- Knox
- Storm

4.1. HDFS Repository Configuration

To add a repository to HDFS, complete the HDFS Create Repository screen as follows:

Field name	Description
Repository Name	The name of the repository; required when configuring agents.
Description	A description of the repository.
Active Status	Enabled or Disabled.
Repository Type	HDFS (cannot be modified).
User Name	The end system username that can be used for connection.
Password	The password for the username entered above.
fs.defaultFS	The location of the Hadoop HDFS service, as noted in the hadoop configuration file core-site.xml OR (if this is a HA environment) the path for the primary NameNode.
hadoop.security.authorization	The type of authorization in use, as noted in the hadoop configuration file core-site.xml; either `simple` or `Kerberos`. (Required only if authorization is enabled).
hadoop.security.auth_to_local	Maps the login credential to a username with Hadoop; use the value noted in the hadoop configuration file, core-site.xml.
dfs.datanode.kerberos.principal	The principal associated with the datanode where the repository resides, as noted in the hadoop configuration file hdfs-site.xml. (Required only if Kerberos authenticaiton is enabled).
dfs.namenode.kerberos.principal	The principal associated with the NameNode where the repository resides, as noted in the hadoop configuration file hdfs-site.xml. (Required only if Kerberos authentication is enabled).
dfs.secondary.namenode.kerberos.principal	The principal associated with the secondary NameNode where the repository resides, as noted in the hadoop configuration file hdfs-site.xml. (Required only if Kerberos authentication is enabled).
Common Name For Certificate	The name of the certificate.

4.2. Hive Repository Configuration

To add a repository to Hive,complete the Hive Create Repository screen as follows:

Field name	Description
Repository Name	The name of the repository; required when configuring agents.
Description	A description of the repository.
Active Status	Enabled or Disabled.
Repository Type	Hive (cannot be modified).
User Name	The end system username that can be used for connection.
Password	The password for the username entered above.
jdbc.driver ClassName	The full classname of the driver used for Hive connections. Default: org.apache.hive.jdbc.HiveDriver
jdbc.url	The complete connection URL, including port and database name. (Default port: 10000.) For example, on the sandbox, jdbc:hive2://sandbox:10000/.
Common Name For Certificate	The name of the certificate.

4.3. HBase Repository Configuration

To add a repository to HBase, complete the HBase Create Repository screen as follows:

Field name	Description
Repository Name	The name of the repository; required when configuring agents.
Description	A description of the repository.
Active Status	Enabled or Disabled.
Repository Type	HBase (cannot be modified).
User Name	The end system username that can be used for connection.
Password	The password for the username entered above.
hadoop.security.authorization	The complete connection URL, including port and database name. (Default port: 10000.) For example, on the sandbox, jdbc:hive2://sandbox:10000/.
hbase.master.kerberos.principal	The Kerberos principal for the HBase Master. (Required only if Kerberos authenticaiton is enabled.)
hbase.security.authentication	As noted in the hadoop configuration file hbase-site.xml.
hbase.zookeeper.property.cientPort	As noted in the hadoop configuration file hbase-site.xml.
hbase.zookeeper.quorum	As noted in the hadoop configuration file hbase-site.xml.
zookeeper.znode.parent	As noted in the hadoop configuration file hbase-site.xml.

4.4. Knox Repository Configuration

To add a repository to Knox, complete the Knox Create Repository screen as follows:

Field name	Description
Repository Name	The name of the repository; required when configuring agents.
Description	A description of the repository.
Active Status	Enabled or Disabled.
Repository Type	Knox (cannot be modified).
User Name	The end system username that can be used for connection.
Password	The password for the username entered above.
knox.url	The Gateway URL for Knox.
Common Name For Certificate	The name of the certificate.

4.5. Storm Repository Configuration

To add a repository to Storm, complete the Storm Create Repository screen as follows:

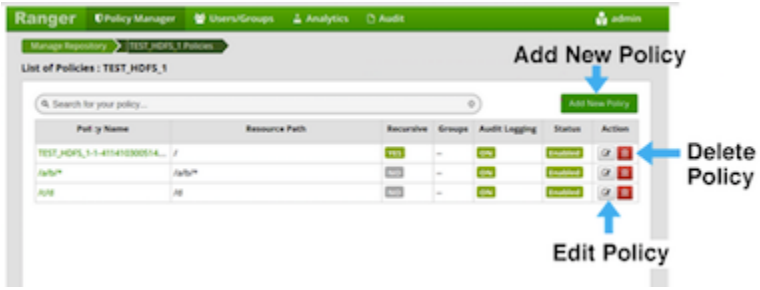
Field name	Description
Repository Name	The name of the repository; required when configuring agents.
Description	A description of the repository.
Active Status	Enabled or Disabled.

Field name	Description
Repository Type	Storm (cannot be modified).
User Name	The end system username that can be used for connection.
Password	The password for the username entered above.
nimbus.url	Hostname of nimbus format, in the form: `http://ipaddress:8080`.
Common Name For Certificate	The name of the certificate.

5. Policy Manager

To examine the policies associated with each repository, go to the service where the repository resides and click the `edit` button. The Ranger Policy Manager view opens and an expanded view of that repository displays, with the policies listed beneath. The policy view includes a search window.

- **To add a new policy** to the repository, click the Add New Policy button. The form looks slightly different, depending on the type of repository to which you are adding the policy.
- **To edit a policy**, click the Edit icon to the right of the entry for that repository. The Policy Manager displays an expanded view of that policy.
- **To delete a** policy, click the Delete icon to the right of the entry for that repository.



Ranger Policy Manager

Open Repository With Policy List

This section describes the requirements for policy creation in

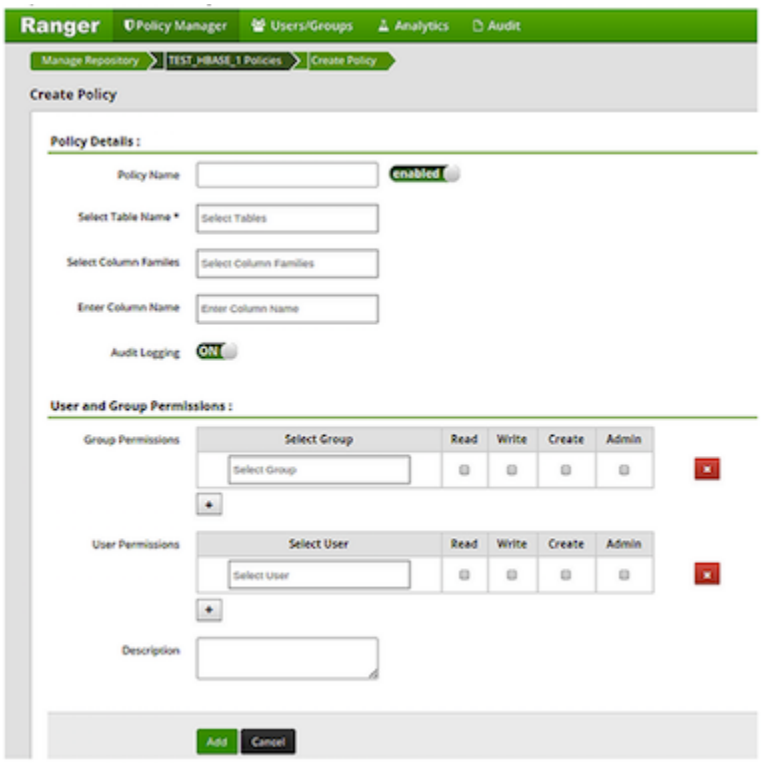
- HDFS
- Hive
- HBase
- Knox
- Storm

5.1. HDFS Policy Creation

Through configuration, Apache Ranger enables both Ranger policies and HDFS permissions to be checked for a user request. When the NameNode receives a user request, the Ranger plugin checks for policies set through the Ranger Policy Manager. If there are no policies, the Ranger plugin checks for permissions set in HDFS.

We recommend that permissions be created at the Ranger Policy Manager, and to have restrictive permissions at the HDFS level.

To add a policy to an HDFS repository,use the HDFS Add Policy form.



HDFS Policy Creation Console

5.2. HDFS Add Policy

Complete the HDFS Add Policy Form as follows:

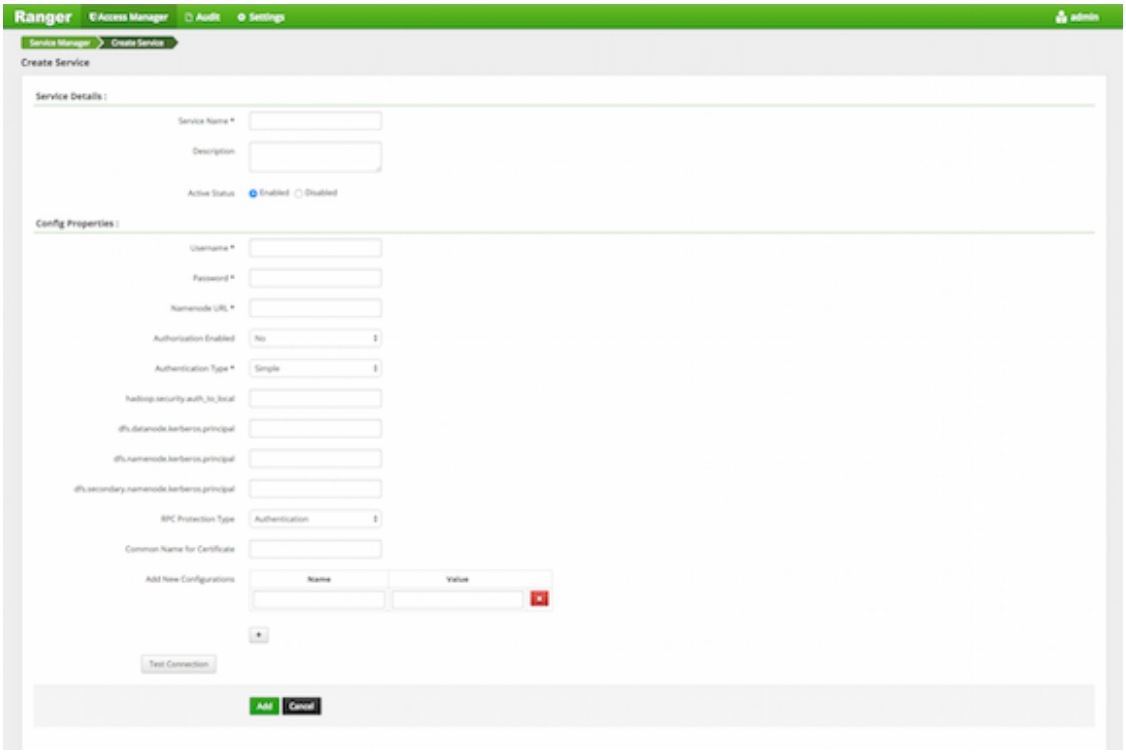


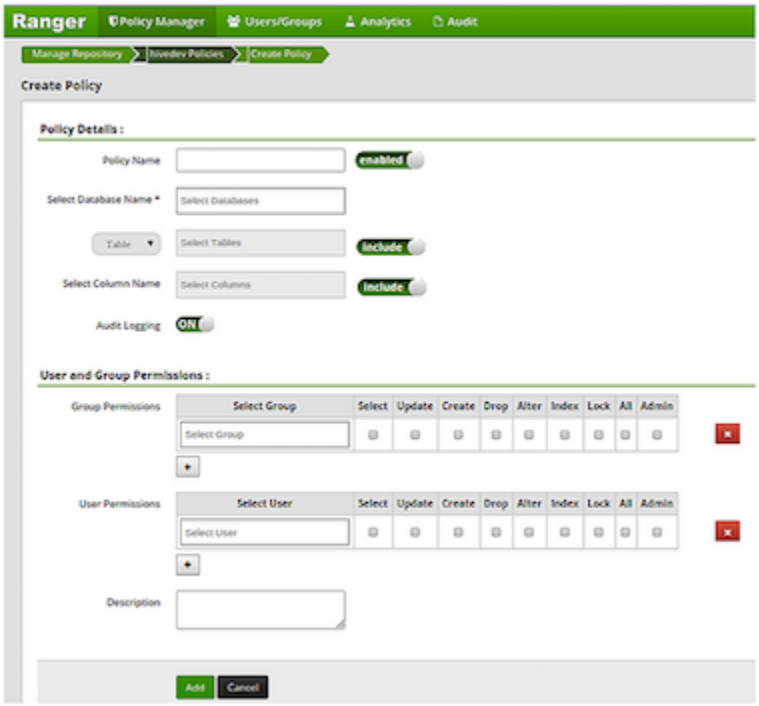
Table 5.1. HDFS Add Policy Fields

Field	Description
Enter Policy Name	Enter a unique name for this policy. The name cannot be duplicated anywhere in the system.
Resource Path	Define the resource path for the policy folder/file. To avoid the need to supply the full path OR to enable the policy for al subfolders or files, you can either complete this path using wildcards (for example, /home*) or specify that the policy should be recursive. (See below.)
Description	(Optional) Describe the purpose of the policy.
Recursive	Select if all files or subfolders within the existing folder will be included in this policy. (Use this option if you have specified a specific Resource Path to the top-level folder, but want all subfolders or files to be included).
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Group Permissions	Use the pick list to assign group permissions appropriate to this policy. If desired, assign the group Administration privileges for the chosen resource. To add users or groups to the list, click the + button. (For further information, see Users).
User Permissions	Use the pick list to assign group permissions appropriate to this policy. If desired, designate one or more users as Administrators for the chosen resource.
Enable/Disable	Policies are enabled by default. To restrict user/group access for a policy, disable the policy.

5.3. Hive Policy Creation

You can create policies in Hive at the Database Name, Table Name, and Column Name level.

To add a policy to a Hive repository, use the Hive Add Policy form.



Hive Policy Creation Console

5.4. Hive Policy Creation

Complete the Hive Add Policy Form as follows:

Table 5.2. Hive Add Policy Fields

Field	Description
Enter Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
Select DataBase Name	Select the appropriate database. Multiple databases can be selected for a particular policy. This field is mandatory.
Table/UDF Drop-down	To continue adding a table-based policy, keep Table selected. To add a User Defined Function (UDF), select UDF.
Select Table Name	For the selected database, select table(s) for which the policy will be applicable.
Select Column Name	For the selected database and table(s), select columns for which the policy will be applicable.
Enter UDF Name	When UDF is selected, this field displays in place of Select Table Name and Select Column Name. Enter the name of

Field	Description
	the User Defined Function that should be the subject of the new policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Group Permissions	Specify the group to which this policy applies. To designate the group as an Administrator for the chosen research, specify Admin permission. (Administrators can create child policies based on existing policies).
User Permissions	Specify a particular user to which this policy applies (outside of an already-specified group) Or designate a particular user as Admin for this policy (Administrators can create child policies based on existing policies).
Include/Exclude	Flags particular fields (table names or column names) as being included or excluded from consideration in the policy.
Enable/Disable	Policies are enabled by default. To restrict user or group access for the policy, select Disable.

Wild cards can be included in the resource path, in the database name, the table name, or column name:

- ***** indicates zero or more occurrences of characters
- **?** indicates a single character

5.5. Providing User Access to Hive Database Tables from the Command Line

Hive provides the means to manage user access to Hive database tables directly from the command line. The most commonly-used commands are:

- GRANT

  Syntax: `grant` <permissions> `on table` <table> `to user` <user or group>`;`

  For example, to create a policy that grants user1 SELECT permission on the table default-hivesmoke22074, the command is `grant select on table default.hivesmoke22074 to user user1;`

  The syntax is the same for granting UPDATE, CREATE, DROP, ALTER, INDEX, LOCK, ALL and ADMIN rights.

- REVOKE

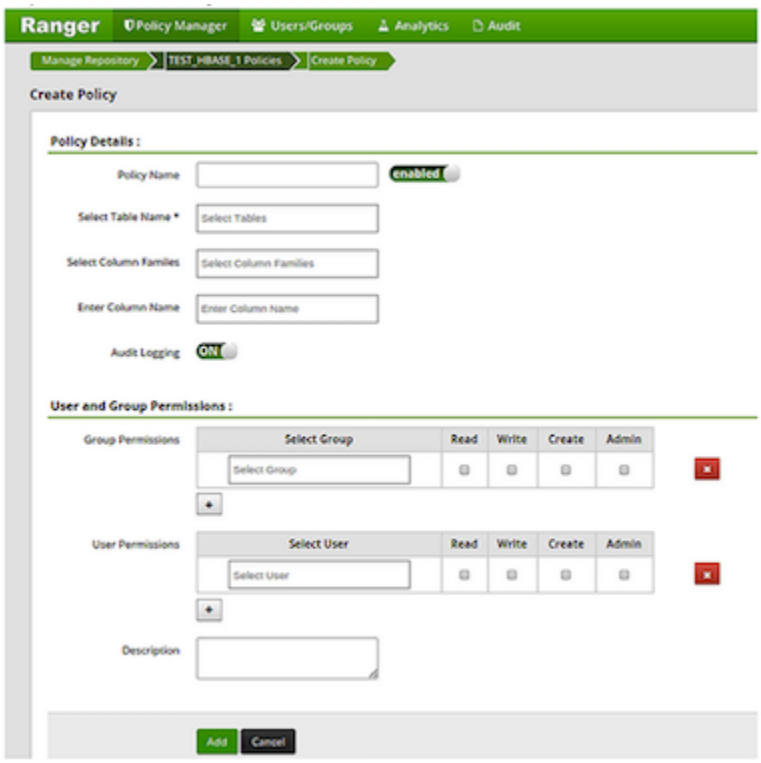
  Syntax: `revoke` <permissions> `on table` <table> `from user` <user or group>`;`

  For example, to revoke the SELECT rights of user1 to the table default.hivesmoke22074, the command is `revoke select on table default.hivesmoke22074 from user user1;`

  The syntax is the same for revoking UPDATE, CREATE, DROP, ALTER, INDEX, LOCK, ALL and ADMIN rights.

5.6. HBase Policy Creation

To add a policy to an HBase repository, use the HBase Create Policy form.



HBase Policy Creation Console

5.7. HBase Policy Creation

Complete the HBase Create Repository screen as follows:

Label	Description
Enter Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
Select Table Name	Select the appropriate database. Multiple databases can be selected for a particular policy. This field is mandatory.
Select Column Families	For the selected table, specify the column families to which the policy applies.
Select Column Name	For the selected table and column families, specify the columns to which the policy applies.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Group Permissions	Specify the group to which this policy applies. To designate the group as an Administrator for the chosen resource, specify Admin permissions. (Administrators can create child policies based on existing policies).
User Permissions	Specify a particular user to which this policy applies (outside of an already-specified group) OR designate a

Label	Description
	particular user as Admin for this policy. (Administrators can create child policies based on existing policies).
Enable/Disable	Policies are enabled by default. To restrict user or group access to the policy, select Disable.

Wild cards can be included in the resource path, in the database name, the table name, or column name:

- ***** indicates zero or more occurrences of characters
- **?** indicates a single character

5.8. Providing User Access to HBase Database Tables from the Command Line

HBase provides the means to manage user access to Hive database tables directly from the command line. The most commonly-used commands are:

- GRANT

  Syntax: `grant '<user-or-group>','<permissions>','<table>`

  For example, to create a policy that grants user1 read/write permission on the table usertable, the command is `grant 'user1','RW','usertable'`

  The syntax is the same for granting CREATE and ADMIN rights.

- REVOKE

  Syntax: `revoke '<user-or-group>','<usertable>'`

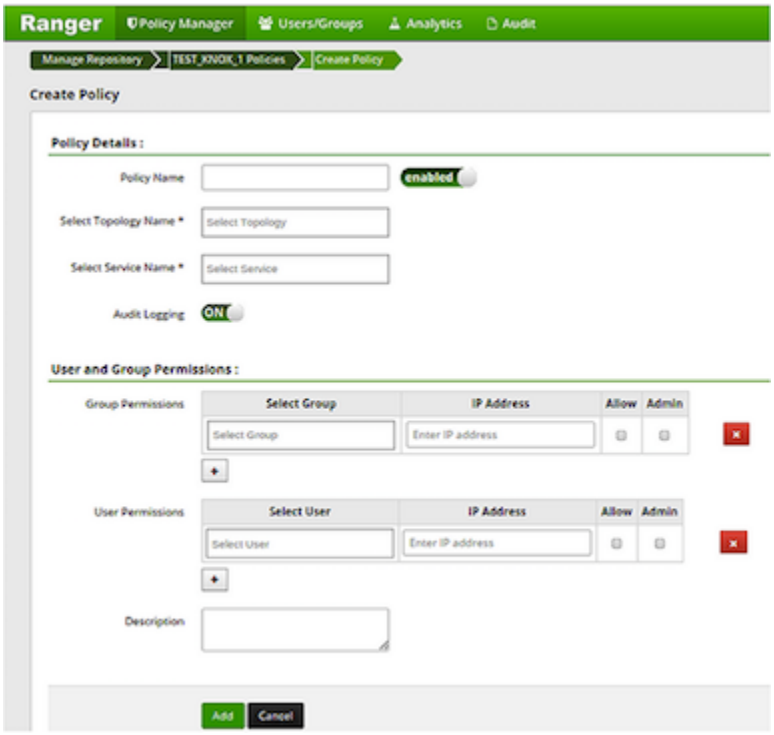
  For example, to revoke the read/write access of user1 to the table usertable, the command is `revoke 'user1','usertable'`

Note

Unlike Hive, HBase has no specific revoke commands for each user privilege.

5.9. Knox Policy Creation

To add a policy to a Knox repository, use the Knox Add Policy Form.



Knox Policy Creation Console

5.10. Knox Policy Creation

Complete the Knox Create Policy screen as follows:

Table 5.3. Knox Policy Labels

Label	Description
Enter Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system.
Select Topology Name	A topology is a graph of computation. Each node in a topology contains processing logic, and links between nodes indicate how data should be passed around between nodes. Enter an appropriate topology name.
Select Service Name	Service Name: Binds a Hadoop service with an internal URL that the Knox gateway uses to proxy requests from external clients to the internal cluster services. Enter an appropriate Service Name.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Group Permissions	Specify the group to which this policy applies. To designate the group as an Administrator for the chosen resource, specify the Admin permissions. (Administrators can create child policies based on existing policies).
User Permissions	Specify a particular user to which this policy applies (outside of an already-specified group) OR designate a particular user as Admin for this policy. (Administrators can create child policies based on existing policies).
Enable/Disable	Policies are enabled by default. To restrict user or group access to the policy, select Disable.

Wild cards can be included in the resource path, in the database name, the table name, or column name:

- * indicates zero or more occurrences of characters
- ? indicates a single character

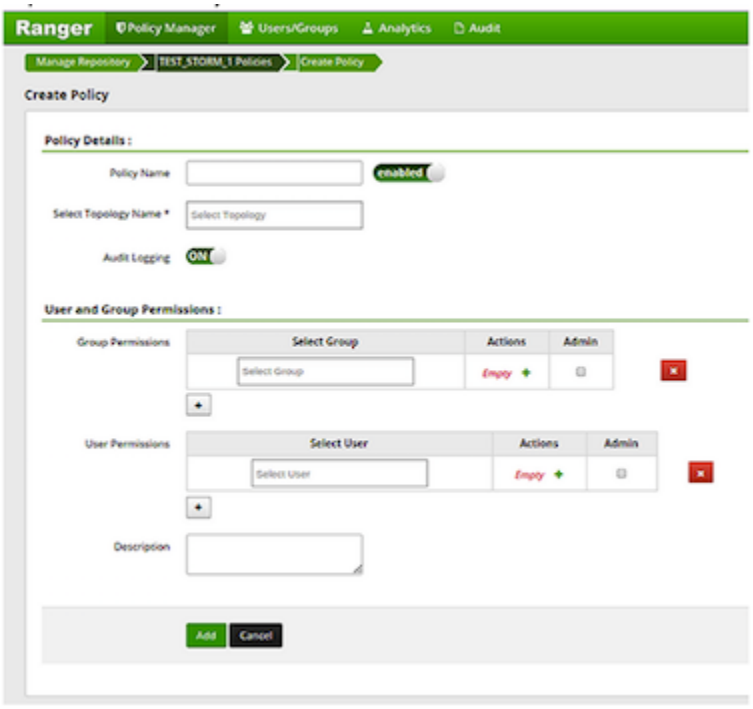
Since Knox does not provide a command line methodology for assigning privileges or roles to users, the User and Group Permissions portion of the Knox Create Policy form is especially important.

Table 5.4. Knox Permissions

Permission	Description
IP Address	The IP address from which the user logs in.
Allow	Permits user to access topology that is specified in the topology name.
Admin	Gives the user Admin privileges.

5.11. Storm Policy Creation

To add a policy to a Storm repository, use the Storm Create Policy Form.



Storm Policy Creation Console

Label	Description
Enter Policy Name	Enter an appropriate policy name. This name is cannot be duplicated across the system.
Select Service Name	Service Name:Binds a Hadoop service with an internal URL that the gateway uses to proxy requests from external

Label	Description
	clients to the internal cluster services. Enter an appropriate Service Name.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Group Permissions	Specify the group to which this policy applies. To designate the group as an Administrator for the chosen resource, specify Admin privileges. (Administrators can create child policies based on existing policies).
User Permissions	Specify a particular user to which this policy applies (outside of an already-specified group) OR designate a particular user as Admin for this policy. (Administrators can create child policies based on existing policies).
Enable/Disable	Policies are enabled by default. To restrict user or group access to the policy, select Disable.

Wild cards can be included in the resource path, in the database name, the table name, or column name:

- ***** indicates zero or more occurrences of characters
- **?** indicates a single character

Since Storm does not provide a command line methodology for assigning privileges or roles to users, the User and Group Permissions portion of the Storm Create Policy form is especially important.

To assign user privileges or roles, complete the User and Group Permissions portion of the Storm Create Policy form.

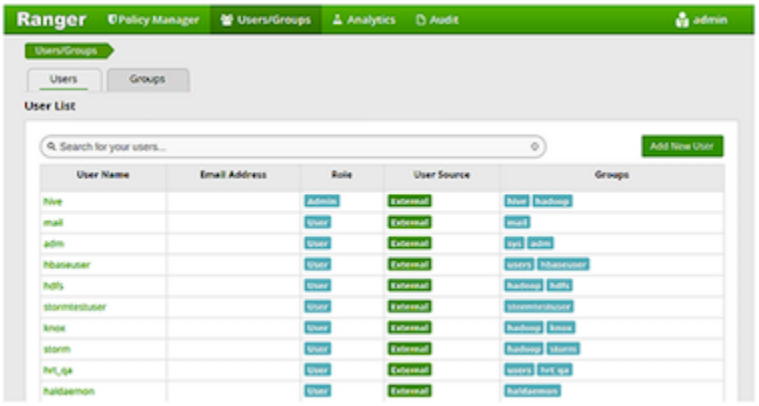
Table 5.5. Knox User and Group Permissions

Actions	Description
File upload	Allows a user to upload files.
Get Nimbus Conf	Allows a user to access Nimbus configurations.
Get Cluster Info	Allows a user to get cluster information.
File Download	Allows a user to download files.
Kill Topology	Allows a user to kill the topology.
Rebalance	Allows a user to rebalance topologies.
Activate	Allows a user to activate a topology.
Deactivate	Allows a user to deactivate a topology.
Get Topology Conf	Allows a user to access a topology configuration.
Get Topology	Allows a user to access a topology.
Get User Topology	Allows a user to access a user topology.
Get Topology Info	Allows a user to access topology information.
Upload New Credential	Allows a user to upload a new credential.
Admin	Provides a user with delegated admin access.

6. Users/Groups Administration

To examine the list of users and groups who can access the Ranger portal or its repositories, click the **User/Groups** tab at the top of the Ranger Console. The User/sGroup view displays:

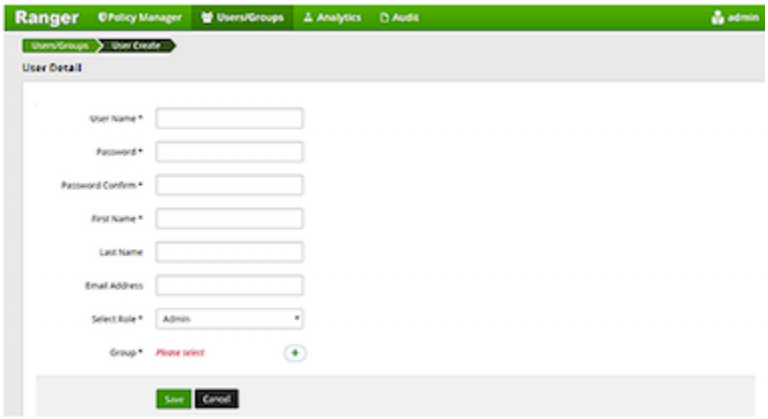
- Internal users who can log in to the Ranger portal; created by the Ranger console Policy Manager
- External users who can access repositories controlled by the Ranger portal; created at other systems like Active Directory, LDAP or UNIX, and synched with those systems
- Admins who are the only users with permission to create users and create repositories, run reports, and perform other administrative tasks. Admins can also create child policies based on the original policy (base policy).



Users/Group List with Users tab active

6.1. Add User

To add a new user to the user list, click the **Users** sub-tab, then click **Add New User**. Add the appropriate user details, then click **Save**. The user is immediately added to the list.

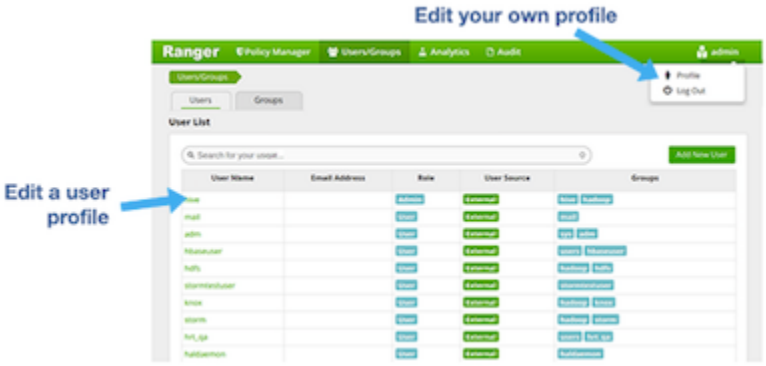


Ranger User Detail

6.2. Edit User

To edit a user, click the **Users** sub-tab, then click the user name and edit the appropriate details. If the user is internal, you can edit any detail. If the user is external, you can only edit their Role.

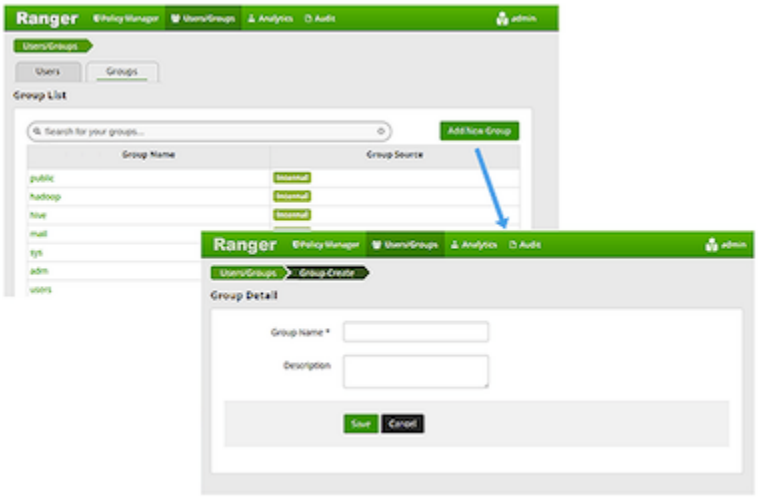
To edit your own user details, click your name in the upper-right hand corner of the Ranger Console, then click **Profile**.



Edit User

6.3. Add Group

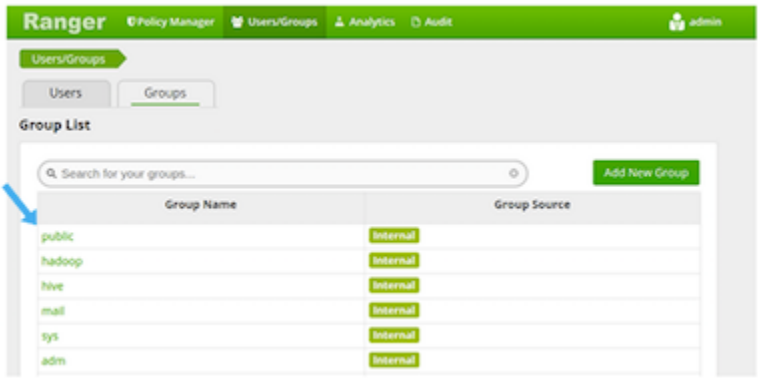
To add a group, click the **Groups** sub-tab, then click **Add a Group**. Enter a unique name for the group, and an optional description, then click **Save**.



Add Group Detail

6.4. Edit Group

To edit a group, click the **Groups** sub-tab, then click the group name and edit the appropriate details. Click **Save**.


Ranger
Policy Manager
Users/Groups
Analytics
Audit
admin
Users/Groups
Users
Groups
Group List
Search for your groups...
Add New Group
Group Name
Group Source
public
hadoop
hive
mail
sys
adm
Internal

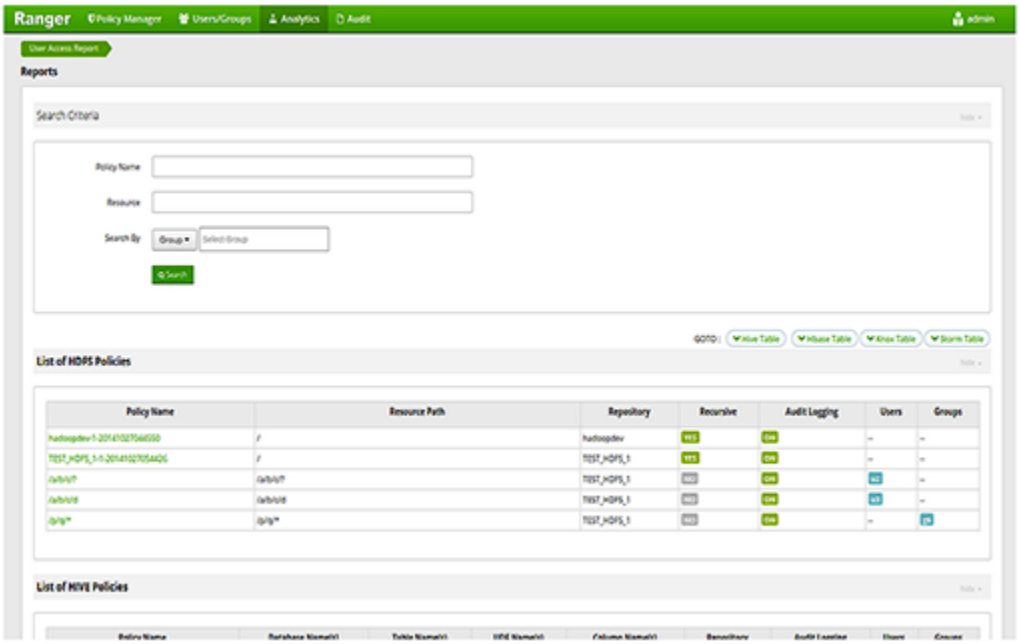
Edit Group

7. Analytics Administration

To perform analytics on one or more policies, click the **Analytics** tab at the top of the Ranger Console. A list of all HDFS, Hive, HBase, Knox and Storm policies displays, and a search window.

You can search:

- **Policy Name** - The policy name assigned to the policy while creating it.
- **Resource Path** - The resource path used while creating the policy.
- **Group, User Name** - The group and the users to which the policy is assigned



Ranger Analytics Tab

8. Auditing

To explore options for auditing policies in Ranger, click the **Audit** tab.

The Audit view contains four sub-tabs:

- Access
- Admin
- Login
- Agents

Note

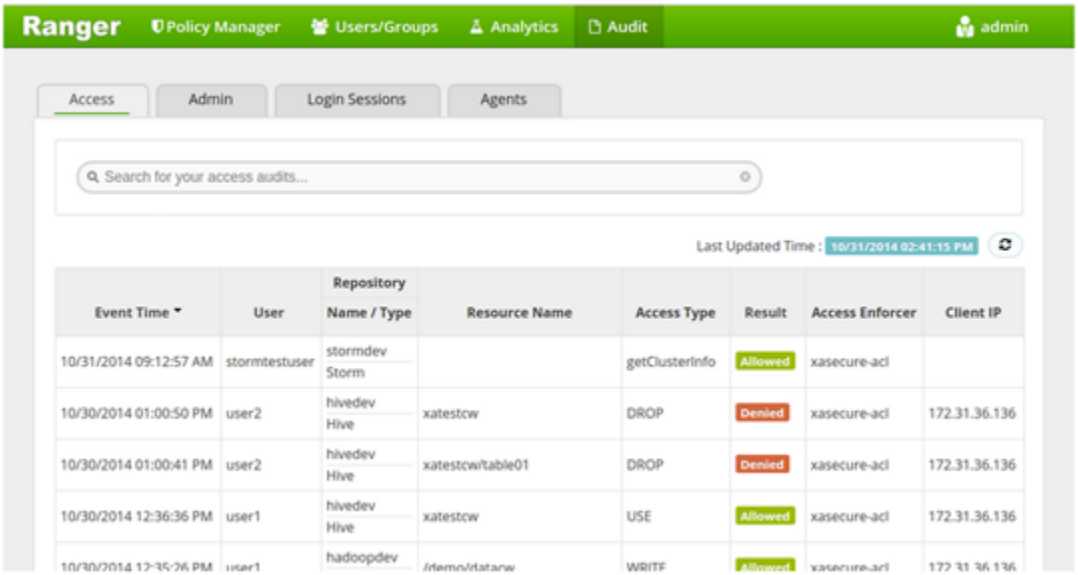
To access audit logs from the Ranger Audit UI, enable XAAUDIT.DB.IS_ENABLED in your Ranger plug-in configuration.

8.1. Access Sub-tab

The Access sub-tab provides Repository activity data for all policies that have Audit set to **On**. The default repository Policy is configured to log all user activity within the Repository. This default policy does not contain user and group access rules.

You can filter the data based on the following criteria:

- **Actions** – Operations performed on resources, such as CREATE, UPDATE, DELETE, and password change
- **Audit Type** – Resource (for operations performed on resources), Assets (for operations performed on Policy) or Users
- **Session ID** – The session count increments each time you try to log into the system
- **User** – The mode through which the user tries to log in (username through which the operation was performed)
- **Start Date, End Date** – Filters results for a particular date range.



Ranger Console Audit tab, Access sub-tab

8.2. Different view when we click on an operation (Update operation in this case)

This module logs the information related to the sessions for each login. You can filter the data based on the search criteria listed in the table below.

Table 8.1. Auditing Search Criteria

Search Criteria	Description
Login ID	The user name through which you login to the system.
Sessionid	The session count increments each time you try to login to the system.
Start Date, End Date	Login time and date is stored for each session. A date range is used to filter the results for that particular date range.
Login Type	The mode through which the user tries to login. By entering username and password.
IP	The IP of the system through which we log in.
User Agent	Login time and date is stored for each session.
Login time	Result indicates whether the Login was successful or not.It can be one of the following: 'Success','Wrong Password', 'Account Disabled', 'Locked', 'Password Expired', 'User Not Found'.

8.3. Agents

This module shows the upload history of the Security Agents.This module displays all the repositories exported from the system.You can filter the data based on the following:

Table 8.2. Agents Search Criteria

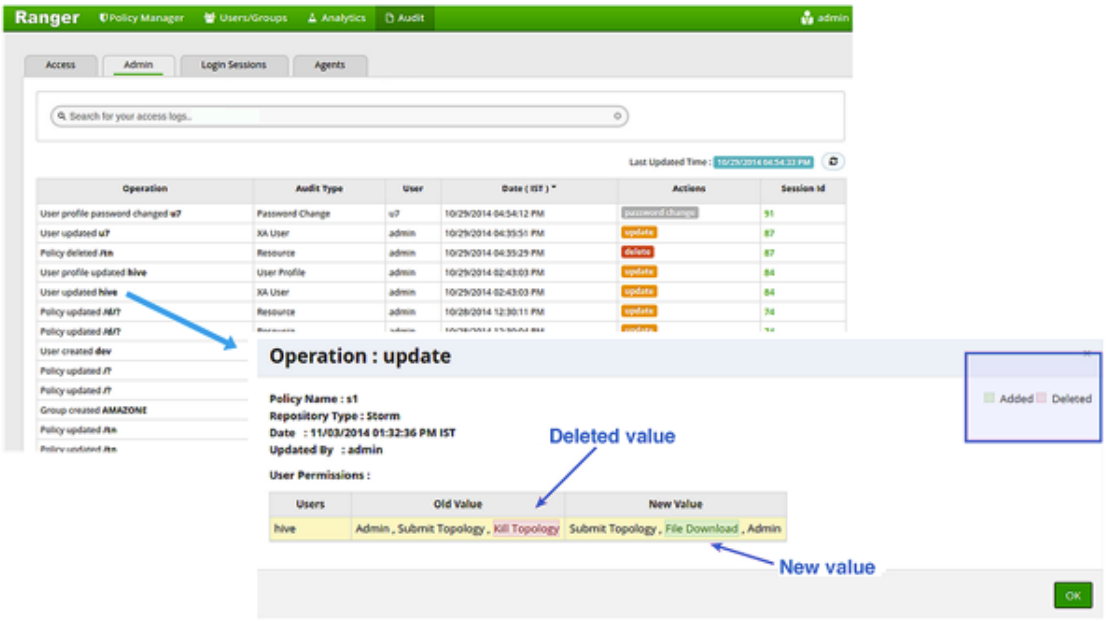
Search Criteria	Description
Agent IP	IP address of the agent that tries to export the repository.
Agent ID	Name of the agent that tries to export the repository.
HTTP Response Code	The HTTP code returned when trying to export the repository.
Start Date, End Date	Export time and date is stored for each agent. A date range is used to filter the results for that particular date range.
Repository Name	The repository name we are trying to export.

8.4. Admin Sub-tab

The Admin sub-tab module contains all events for the HDP Security Administration Web UI, including Repository, Policy Manager, Log in, etc. (actions like create,update,delete,password change).

You can filter the data based on the following criteria:

- **Actions** – Operations performed on resources, such as CREATE, UPDATE, DELETE, and password change
- **Audit Type** – Resource (for operations performed on resources), Assets (for operations performed on Policy) or Users
- **Session ID** – The session count increments each time you try to log into the system
- **User** – The mode through which the user tries to log in (username through which the operation was performed)
- **Start Date, End Date** – Filters results for a particular date range.

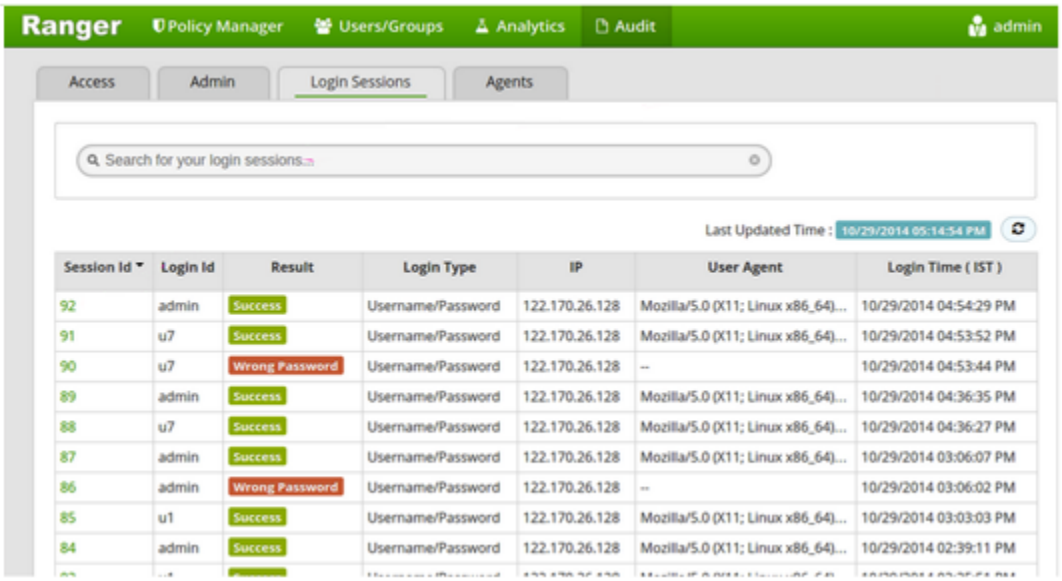


Ranger Console Audit tab, Access sub-tab

8.5. Login Sessions Sub-tab

The Login Sessions sub-tab logs the information related to the sessions for each login. You can filter the data based on the following criteria:

- **Login ID** – The username through which someone logs in to the system
- **Session-id** – The session count increments each time the user tries to log into the system
- **Start Date, End Date** – Specifies that results should be filtered based on a particular start date and end date
- **Login Type** – The mode through which the user tries to login (by entering username and password)
- **IP** – The IP address of the system through which the user logged in
- **User Agent** – The login time and date for each session
- **Login time** – Logs whether or not the login was successful. Possible results can be Success, Wrong Password, Account Disabled, Locked, Password Expired or User Not Found



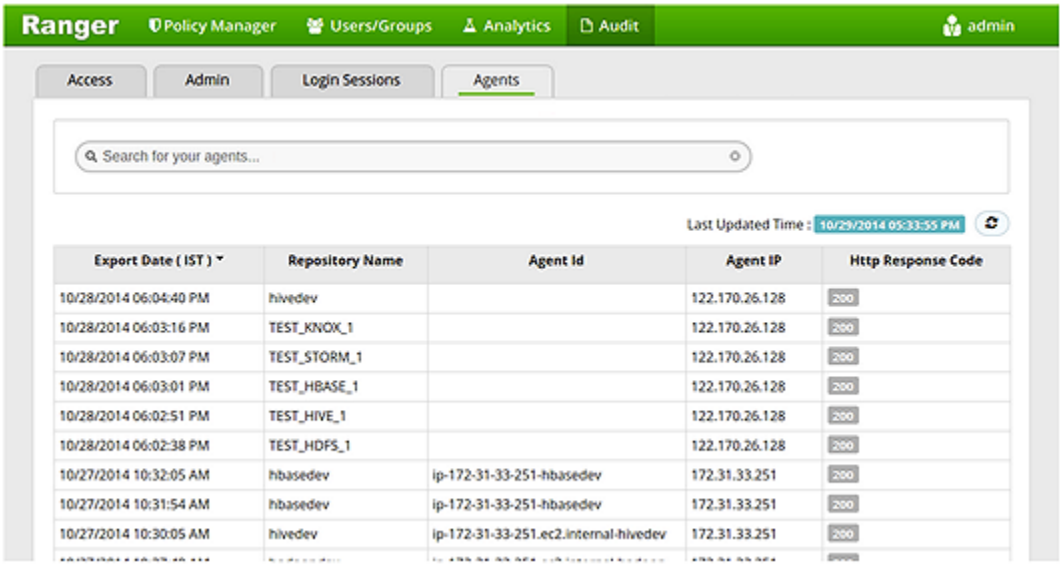
Ranger Console Audit tab, Login Sessions sub-tab

8.6. Agents Sub-tab

The Agents sub-tab shows the upload history of the Security Agents. This module lists all the repositories exported from the system.

You can filter the data based on the following criteria:

- **Agent IP** – the IP address of the agent that tried to export the repository
- **Agent ID** – the name of the agent that tried to export the repository
- **HTTP Response Code** – the HTTP code obtained when the export was attempted
- **Start Date, End Date** – filters results for a particular start date and end date
- **Repository Name** – the name of the exported repository



Ranger Console Audit tab, Agents sub-tab

9. Special Requirements for High Availability Environments

Special Requirements for High Availability Environments In a HA environment, primary and secondary NameNodes must be configured as described in the HDP System Administration Guide.

To enable Ranger in the HDFS HA environment, the HDFS plugin must be set up in each NameNode, and then pointed to the same HDFS repository set up in the Security Manager. Any policies created within that HDFS repository are automatically synchronized to the primary and secondary NameNodes through the installed Apache Ranger plugin. That way, if the primary NameNode fails, the secondary namenode takes over and the Ranger plugin at that NameNode begins to enforce the same policies for access control.

When creating the repository, you must include the `fs.default.name` property must be set to the full hostname of the primary NameNode. If the primary NameNode fails during policy creation, you can then temporarily use the `fs.default.name` of the secondary NameNode in the repository details to enable directory lookup for policy creation.

If, while the primary node is down, you wish to create new policies, there is a slight difference in user experience when specifying the resource path. If everything is normal, this is a drop-down menu with selectable paths; however, if your cluster is running from the familover node, there will be no drop-down menu, and you will need to manually enter the path.

Primary NameNode failure does not affect the actual policy enforcement. In this setup for HA, access control is enforced during primary NameNode failure, by the Ranger plugs at the secondary NameNodes.

10. Adding a New Component to Apache Ranger

This document provides a general description of how to add a new component to Apache Ranger.

Apache Ranger has three main components:

- **Admin Tool** – Provides web interface & REST API for managing security policies
- **Custom Authorization Module for components** – Provides custom authorization within the (Hadoop) component to enforce the policies defined in Admin Tool
- **UserGroup synchronizer** – Enables the user/group information in Apache Ranger to synchronize with the Enterprise user/group information stored in LDAP or Active directory.

For supporting new component authorization using Apache Ranger, the component details needs to be added to Apache Ranger as follows:

- Add component details to the Admin Tool
- Develop a custom authorization module for the new component

Adding Component Details to the Admin Tool

The Apache Ranger Admin tool supports policy management via both a web interface (UI) and support for (public) REST API. In order to support a new component in both the UI and the Server, the Admin Tool needs to be modified.

Required UI changes to support the new component:

1. Add a new component template to the Policy Manager page (console home page):

   Show new component on the policy manager page i.e home page[#!/policymanager]. Apache Ranger needs to add table template to policy manager page and make changes in corresponding JS files. Ranger also needs to create a new repository type enum to distinguish the component for which the repository/policy is created/updated.

   For example: Add a table template to PolicyManagerLayout_tmpl.html file to view the new component on the Policy Manager page and make changes in the PolicyManagerLayout.js file related to the new componen, such as passing knox repository collection data to the PolicyManagerLayout_tmpl template. Also create a new repository type enum (for example, ASSET_KNOX) in the XAEnums.js file.

2. Add new configuration information to the Repository Form:

   Add new configuration fields to Repository Form [AssetForm.js] as per new component configuration information. This will cause the display of new configuration fields in the corresponding repository Create/Update page. Please note that the AssetForm.js is a common file for every component to create/update the repository.

For example: Add new field(configuration) information to AssetForm.js and AssetForm_tmpl.js.

3. Add a new Policy Listing page:

   Add a new policy listing page for the new component in the View Policy list. For example: Create a new KnoxTableLayout.js file and add JS-related changes as per the old component[HiveTableLayout.js] to the View Policy listing. Also create a template page, KnoxTableLayout_tmpl.html.

4. Add a new Policy Create/Update page:

   Add a Policy Create/Update page for the new component. Also add a policy form JS file and its template to handle all policy form-related actions for the new component. For example: Create a new KnoxPolicyCreate.js file for Create/Update Knox Policy. Create a KnoxPolicyForm.js file to add knox policy fields information. Also create a corresponding KnoxPolicyForm_tmpl.html template.

5. Other file changes, as needed:

   Make changes in existing common files as per our new component like Router.js, Controller.js, XAUtils.js, FormInputList.js, UserPermissionList.js, XAEnums.js, etc.

Required server changes for the new component:

Let's assume that Apache Ranger has three components supported in their portal and we want to introduce one new component, Knox:

1. Create New Repository Type

   If Apache Ranger is introducing new component i.e Knox, then they will add one new repository type for knox. i.e repositoryType = “Knox”. On the basis of repository type, while creating/updating repository/policy, Apache Ranger will distinguish for which component this repository/policy is created/updated.

2. Add new required parameters in existig objects and populate objects

   For Policy Creation/Update of any component (i.e HDFS, Hive, Hbase), Apache Ranger uses only one common object, `VXPolicy.` The same goes for the Repository Creation/Update of any component: Apache Ranger uses only one common object `VXRepository.` As Apache Ranger has three components, it will have all the required parameters of all of those three components in `VXPolicy/VXRepository.` But for Knox, Apache Ranger requires some different parameters which are not there in previous components. Thus, it will add only required parameters into `VXPolicy/VXRepository` object. When a user sends a request to the Knox create/update policy, they will only send the parameters that are required for Knox to create/update the VXPolicy object.

   After adding new parameters into VXPolixy/VXRepository, Apache Ranger populates the newly-added parameters in corresponding services, so that it can map those objects with Entity Object.

3. Add newly-added fields (into database table) related parameters into entity object and populate them

As Apache Ranger is using JPA-EclipseLink for database mapping into java, it is necessary to update the Entity object. For example, if for Knox policy Apache Ranger has added two new fields (`topology` and `service`) into db table `x_resource`, it will also have to update the entity object of table (i.e `XXResource`), since it is altering table structure.

After updating the entity object Apache Ranger will populate newly-added parameters in corresponding services (i.e XResourceService), so that it can communicate with the client using the updated entity object.

4. Change middleware code business logic

   After adding and populating newly required parameters for new component, Apache Ranger will have to write business logic into file `AssetMgr`, where it may also need to do some minor changes. For example, if it wants to create a default policy while creating the Repository, then on the basis of repositoryType, Apache Ranger will create one default policy for the given repository. Everything else will work fine, as it is common for all components.

Required database changes for the new component:

For repository and policy management, Apache Ranger includes the following tables:

- x_asset (for repository)
- x_resource (for repository)

As written above, if Apache Ranger is introducing new component then it is not required to create individual table in database for each component. Apache Ranger has common tables for all components.

If Apache Ranger has three components and wants to introduce a fourth one, then it will add required fields into these two tables and will map accordingly with java object. For example, for Knox, Apache Ranger will add two fields (`topology`, `service`) into `x_resource`. After this, it will be able to perform CRUD operation of policy and repository for our new component, and also for previous components.

11. Developing a Custom Authorization Module

In the Hadoop ecosystem, each component (i.e., Hive, HBase) has its own authorization implementation and ability to plug in a custom authorization module. To implement the centralized authorization and audit feature for a component, the component should support a customizable (or pluggable) authorization module.

The custom component Authorization Plugin should do the following:

- Provide authorization based on Policies defined in Policy Admin Tool
- Provide audit information based on the authorization decisions

Implementing Custom Component Authorization

To implement the custom component authorization plugin, the Ranger common agent framework provides the following functionalities:

- Ability to read all policies from Policy Manager for a given repository-id
- Ability to log audit information

When the custom authorization module is initialized, the module should do the following:

1. Initiate a REST API call to the "Policy Admin Tool" to retrieve all policies associated with the specific component.
2. Once the policies are available, it should:
    - be built into a custom data structure for enabling the authorization module.
    - kick off the policy updater thread to refresh policies from "Policy Admin Tool" at a regular interval.

When the custom authorization module is called to perform authorization of a component action (such as READ action) on a specific component resource (such as /app folder), the authorization module will:

- **Identify authorization decision** - For each policy:policyList:
    - If (resource in policy <match> auth-requested-resource)
    - If (action-in-policy <match>action-requested
    - If (current-user or current-user-groups or public-group <allowed> for the policy), Return access-allowed
- **Identify auditing needs** - For each policy:policyList
    - If (resource in policy <match> auth-requested-resource), return policy.isAuditEnabled()

12. About HDP

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.**

For more information on Hortonworks technology, Please visit the Hortonworks Data Platform page. 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.